

Second Generation Sparse Models

A DISSERTATION

**SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA**

BY

Ignacio Ramírez

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Doctor of Philosophy

Prof. Guillermo Sapiro

December, 2011

© Ignacio Ramírez 2011

ALL RIGHTS RESERVED

Acknowledgements

To Professor Guillermo Sapiro for being, not only a great advisor, but also a great friend who gave me confidence in myself and my work throughout these years. I feel that I have learned a lot from him, and this is not limited to academic matters.

To Gadiel Seroussi and Gregory Randall, my former Master's research and academic advisors at Universidad de la República back in Uruguay, who have continued providing me invaluable guidance and support during my doctorate research.

To Professors Tom Luo, Nikos Papanikolopoulos and Yousef Saad for accepting to be part of my examination committee, and to the Director of Graduate Studies, Prof. Jiali Gao, who helped me a lot in sorting out the administrative intricacies of the graduate program, enormously simplifying my efforts in this aspect.

To the colleagues and friends with whom I have shared good moments as well as enlightening discussions during these years: Juan Bazerque, Alexey "Pana" Castrodad, Federico Lecumberry, Gonzalo Mateos, Pablo Sprechmann, Iman Aganj, Xue Bai, Leah Bar, Mona Mahmoudi and Adarsh Chandran.

To my friends Julia Demasi, Ana Inés Torres, Heng Chooi, Tal & Nili Makovsky, Ilan & Yasmin Sagiv, who helped me and my family stay warm during the harsh Minnesota winters by being close to us and sharing unforgettable moments.

To my beloved wife Patricia Vaz, who has always supported me, even in the face of the extremely difficult times that we had to endure lately, even while working on her own doctorate, which is still ongoing; to my little kids Joaquin and Violeta who are my ultimate

driving force; to the rest of my family, specially my parents Beatriz and Mario, and to my late mother-in-law, Amalia Jauri, who gave us enormous support and love to the end.

To all of you, thank you very much!

Abstract

Sparse data models, where data is assumed to be well represented as a linear combination of a few elements from a learned dictionary, have gained considerable attention in recent years, and their use has led to state-of-the-art results in many applications.

The success of these models is largely attributed to two critical features: the use of sparsity as a robust mechanism for regularizing the linear coefficients that represent the data, and the flexibility provided by overcomplete dictionaries that are learned from the data. These features are controlled by two critical hyper-parameters: the desired sparsity of the coefficients, and the size of the dictionaries to be learned. However, lacking theoretical guidelines for selecting these critical parameters, applications based on sparse models often require hand-tuning and cross-validation to select them, for each application, and each data set. This can be both inefficient and ineffective. On the other hand, there are multiple scenarios in which imposing additional constraints to the produced representations, including the sparse codes and the dictionary itself, can result in further improvements.

This thesis is about improving and/or extending current sparse models by addressing the two issues discussed above, providing the elements for a new generation of more powerful and flexible sparse models. First, we seek to gain a better understanding of sparse models as data modeling tools, so that critical parameters can be selected automatically, efficiently, and in a principled way. Secondly, we explore new sparse modeling formulations for effectively exploiting the prior information present in different scenarios. In order to achieve these goals, we combine ideas and tools from information theory, statistics, machine learning, and optimization theory. The theoretical contributions are complemented with applications in audio, image and video processing.

Contents

Acknowledgements	i
Abstract	iii
List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Sparsity and sparse coding	4
1.2 Learning overcomplete dictionaries	6
2 Adding structure to sparse models	10
2.1 Structured sparse coding	10
2.1.1 Hierarchical sparsity	11
2.1.2 Collaborative hierarchical sparsity	13
2.2 Structured dictionary learning	16
2.3 Imposing atom smoothness	18
3 An information-theoretic view of learned overcomplete sparse models	21
3.1 Issues with traditional sparse models: a motivating example	21
3.2 Universal models for priors with unknown parameters	23
3.3 Sparse models, model selection, and MDL	28

3.3.1	The class of learned overcomplete sparse models	29
3.3.2	The description length of learned sparse models	29
3.3.3	MDL-based sparse coding and dictionary learning algorithms	30
3.4	Extension to other types of problems	32
4	Conclusions	38
	References	41
Appendix A. Classification and Clustering via Dictionary Learning with Structured		
Incoherence and Shared Features 48		
1	Introduction and Basic Formulation	49
2	Sparse Coding and Dictionary Learning	53
3	The Sparse Representation Quality $\hat{\mathcal{R}}$ and Supervised Classification	54
3.1	Sharing Atoms	56
3.2	Experimental Results	56
4	Dictionary Learning for Clustering	59
4.1	Initialization: Spectral Clustering Meets Dictionary Learning	60
4.2	Theoretical Guarantees	62
4.3	Clustering Results	64
5	Concluding Remarks	66
	References	68
Appendix B. C-HiLasso: A Collaborative Hierarchical Sparse Modeling Framework 71		
1	Introduction and Motivation	72
2	Collaborative Hierarchical Sparse Coding	74
2.1	Background: Lasso and Group Lasso	74
2.2	The Hierarchical Lasso	76

2.3	Collaborative Hierarchical Lasso	78
2.4	Relationship to Recent Literature	80
3	Optimization	82
3.1	Single-Signal Problem: HiLasso	82
3.2	Optimization of the Collaborative HiLasso	85
4	Theoretical guarantees	87
4.1	Block-Sparse Coherence Measures	88
4.2	Recovery Proof	91
5	Experimental results	102
6	Discussion	111
	References	112
Appendix C. Universal Regularizers For Robust Sparse Coding and Modeling		116
1	Introduction	116
2	Sparse modeling and the need for better models	118
2.1	Interpretations of the sparse coding problem	120
2.2	The need for a better model	125
3	Universal models for sparse coding	126
3.1	The conjugate prior	131
3.2	The Jeffreys prior	133
3.3	The conditional Jeffreys	136
4	Optimization and implementation details	138
5	Experimental results	141
5.1	Dictionary learning	141
5.2	MOE as a prior for sparse coding coefficients	142
5.3	Recovery of noisy sparse signals	143
5.4	Recovery of real signals with simulated noise	145

5.5	Zooming	146
5.6	Classification with universal sparse models	148
6	Concluding remarks	150
7	Appendix: Derivation of the MOE model	151
8	Supplementary material	156
	References	157
Appendix D. An MDL framework for sparse coding and dictionary learning		163
1	Introduction	163
2	Background on sparse modeling	168
2.1	Sparse coding	168
2.2	Dictionary learning	169
2.3	Issues with traditional sparse models: a motivating example	170
3	Sparse model selection and MDL	171
3.1	A brief introduction to MDL	173
3.2	Modern MDL and universal coding	175
4	Encoding scheme	176
4.1	Encoding the sparse coefficients	177
4.2	Encoding the error	179
4.3	Model for the dictionary	182
4.4	Extension to sequential (collaborative) coding	184
5	MDL based sparse coding	187
6	MDL based dictionary learning	188
6.1	Optimizing the dictionary for fixed p	191
7	Experimental results	193
7.1	Coding performance	193
7.2	Learning performance	193

7.3	Denoising of natural images	194
7.4	Texture mosaic segmentation	195
7.5	Low-rank matrix approximation	196
8	Concluding remarks	200
9	Supplementary material	200
9.1	MDL-based sparse coding via convex relaxation	200
9.2	ℓ_1 regularized dictionary update	205
	References	206

Appendix E. Low-rank data modeling via the Minimum Description Length principle **211**

1	Introduction	211
2	Low-rank matrix estimation/approximation	213
2.1	Low-rank approximation as dimensionality reduction	214
3	MDL-based low-rank model selection	215
3.1	Encoding Σ	216
3.2	Encoding \mathbf{U} and \mathbf{V} , general case	216
3.3	Encoding \mathbf{U} predictively	218
3.4	Encoding \mathbf{V} predictively	219
3.5	Encoding \mathbf{E}	220
3.6	Model selection algorithm	220
4	Results and conclusion	221
4.1	Conclusion	221
	References	222

List of Tables

3.1	Denoising results summary	32
A.1	Supervised classification results	55
A.2	Unsupervised classification	63
B.1	Simulated signal results	103
B.2	Noisy digit mixtures results	104
B.3	Texture separation results	108
C.1	Image denoising results summary	147

List of Figures

1.1	Sparse data model	2
1.2	Sparse representation example	4
1.3	Example dictionaries	7
2.1	Sparse data model	11
2.2	Sparsity patterns induced by HiLasso and C-HiLasso	13
2.3	Texture separation results	14
2.4	Speaker identification results	15
2.5	Source separation with missing observation	16
3.1	Distribution of DCT coefficients	27
3.2	Denoising results for $\sigma = 10$	33
3.3	Denoising results for $\sigma = 20$	34
3.4	Texture classification results	34
3.5	Low-rank decomposition scheme	36
3.6	Low-rank matrix approximation results	37
A.1	Atoms discarded due to excessive coherence	59
A.2	Bike detection results	60
A.3	Effect of incoherence in clustering performance	65
A.4	Iterative clustering	65
A.5	Texture segmentation results	67
A.6	Source separation example	68

B.1	Sparsity patterns induced by HiLasso and C-HiLasso	76
B.2	Effect of parameters on the HiLasso solution	79
B.3	Indexing conventions	92
B.4	Separation of digit mixtures	105
B.5	Source separation with missing observation	107
B.6	Texture separation results	109
B.7	Speaker identification results	110
C.1	Statistical features of sparse coefficients	124
C.2	Universal sparse regularizers	136
C.3	Empirical distribution of sparse coefficients for encoding image patches . . .	144
C.4	Sample image denoising results	146
C.5	Image zooming results	147
C.6	Textures used in the texture classification example.	149
C.7	Classification results	150
D.1	Sparse code probability model and encoding scheme	178
D.2	Residual probability model	181
D.3	Atom prediction scheme	183
D.4	Collaborative encoding	185
D.5	COMPA evolution example	189
D.6	Denoising results	196
D.7	Texture classification results	197
D.8	Low-rank matrix approximation results	199
D.9	Huber approximation of the LG codelength function.	203
E.1	Encoding scheme	218
E.2	Low-rank approximation of the Lobby scene	224
E.3	Low-rank approximation of the ShoppingMall scene	225

1 Introduction

A *sparse model* is one in which signals of a given type $\mathbf{y} \in \mathbb{R}^m$ can be represented accurately as sparse linear combinations of the columns \mathbf{d}_k (atoms) of a *dictionary* $\mathbf{D} = [\mathbf{d}_1 | \mathbf{d}_2 | \dots | \mathbf{d}_p] \in \mathbb{R}^{m \times p}$,

$$\mathbf{y} = \mathbf{D}\mathbf{a} + \mathbf{e},$$

where by accurate we mean that the model approximation error is small, i.e., $\|\mathbf{e}\| \ll \|\mathbf{y}\|$ (in some norm), and by sparse we mean that the number of non-zero elements in the linear coefficients vector $\mathbf{a} \in \mathbb{R}^p$, denoted by $\|\mathbf{a}\|_0$, is small compared to its dimension p . This concept is depicted in Figure 1.1.

Such models have been used for decades in a variety of applications. Early examples include dimensionality reduction techniques such as Principal Component Analysis (PCA) (Figure 1.3b), or transform-based signal coding (see Figure 1.2). However, it was only recently, with the introduction of *learned overcomplete dictionaries* [1, 2, 3, 4], a mathematical formalization of sparse decompositions as representations of data [5, 6, 7], and the advent of powerful special-purpose optimization algorithms for computing such decompositions [8, 9, 10, 11, 12], that the full power of sparse models seems to have been harnessed, leading to state-of-the-art results in many signal and image processing tasks, e.g., [13, 14, 15, 16]. We refer the reader to [17, 18, 19] for recent reviews on the subject.

These recent results and experimentation, where sparsity and overcompleteness are the main driving concepts behind the studied models, comprise what we call the “first generation sparse models.”

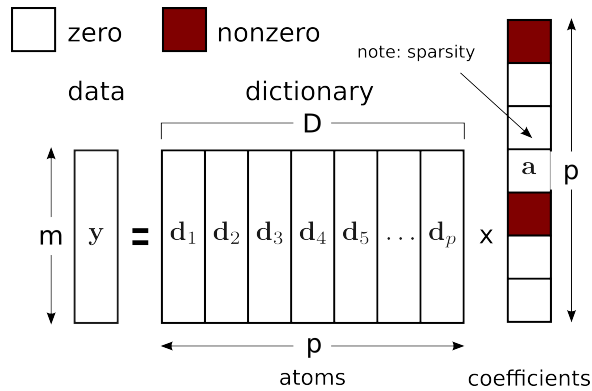


Figure 1.1: Sparse data model for samples $y \in \mathbb{R}^m$ based on a dictionary with p atoms.

When sparsity is a modeling device and not an hypothesis about the nature of the analyzed signals, parameters such as the *desired* sparsity in the solutions, or the size p of the dictionaries to be learned, play a critical role in the effectiveness of sparse models for the data and tasks at hand. However, lacking theoretical guidelines for such parameters, published applications based on learned sparse models often rely on either cross-validation or ad-hoc methods for determining such critical parameters (an exception for example being the Bayesian approach, e.g., [20]). Clearly, such techniques can be impractical and/or ineffective in many cases. This in turn hinders the further application of such models to new types of data and applications, or their evolution into different, possibly more sophisticated, models.

This thesis is about gaining a better theoretical understanding of sparse models based on learned dictionaries of arbitrary size, and extending and/or improving these models by incorporating more prior information in the specific problems to be solved. More specifically, the main questions driving this work are, in order of relevance:

- Q1** If the size p of the dictionary and the *sparsity level* γ (upper bound on $\|a\|_0$) are both unknowns to be learned from the data, how does one define an objective criterion for selecting such critical parameters?

- Q2** How do we add robustness to the process of fitting or learning a sparse model to data?
- Q3** How does one incorporate more prior information besides sparsity to enhance the representations? Examples of this are observed empirical distributions of the coefficients \mathbf{a} , frequency of different atoms in the dictionary \mathbf{D} , dependencies between atoms representing the same sample or between different data samples.
- Q4** How good are sparse representation coefficients as features or patterns representing observed data, for example, in machine learning or pattern recognition/classification tasks?

In order to answer such questions, this work focuses on the use of tools from information theory, and in particular, from *universal coding* [21] and Rissanen's *Minimum Description Length (MDL) principle* [22, 23]. We complement this with other tools and techniques such as compressive sensing and machine learning, to aid in the exploration of structural aspects of sparse models, and their applications for example to classification tasks.

The rest of this chapter introduces traditional sparse models and the associated notation used throughout this document (with the exception of the appendixes, which are self-contained). The exposition of the work itself is described in the two chapters following this introduction. Chapter 2 explores questions Q3-Q4 (and, to some extent, Q2) from a “traditional point of view”, that is, via modifications to the sparse coding and dictionary learning formulations, assessing their effect in various applications. The chapter gives a summary of the work done on this subject, leaving technical details, as well as the complete experimental results, to appendixes A and B. Chapter 2 also serves to further highlight some of the issues related to this traditional approach. In particular, the critical effect of choosing the right coding and learning parameters is aggravated as new terms are added to the cost functions, as each one adds one (or more) new parameter(s) that need(s) to be

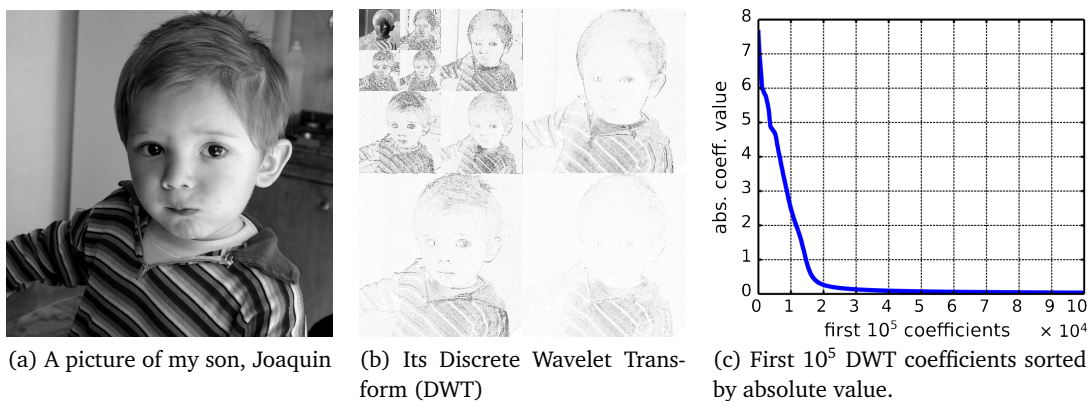


Figure 1.2: Sparse representation example.

tuned. Nevertheless, with properly chosen parameters, we show how such carefully chosen reformulations of the sparse modeling problem can lead to significant improvements in their application to challenging problems such as source separation (see Appendix B).

Chapter 3 deals with questions Q1-Q2 using the information-theoretic tools mentioned. As such, a brief introduction to such tools is provided, with focus on their practical implementation. Then, the main results obtained towards answering such questions are summarized, again leaving the technical details to the associated appendixes C through E.

The document is concluded in Chapter 4, summarizing the main results, the open questions that arose along the way, and the future lines of work suggested by such questions.

1.1 Sparsity and sparse coding

Depending on the application, the sparsity assumption can be justified from many points of view. For example, it is known that DCT or wavelet representations of natural images concentrate most of the energy of the signals in very few coefficients (see Figure 1.2). Those examples were an inspiration for the recent development of compressive sensing theory [7]. In statistics, sparsity is an important tool in model selection (determining the relevant factors affecting given observed phenomena, for example the incidence of a given

disease in a population), and also in model fitting, where ℓ_1 regularization is often preferred over the traditional ℓ_2 (Ridge) regression [24]. In machine learning applications, sparse models have also proved useful, for example in classification tasks (see for example [14, 25, 26]).

We define the support, or active set, of a vector $\mathbf{a} \in \mathbb{R}^p$ as $\text{supp}(\mathbf{a}) = \{k : \mathbf{a}_k \neq 0\}$. Let $\Gamma = \text{supp}(\mathbf{a})$. We use the pseudo-norm $\|\mathbf{a}\|_0 := |\Gamma|$ to denote the number of non-zero elements of \mathbf{a} . The sparse coding problem can be formulated in different ways, depending on the application. For example, one may request the sparsest possible representation of a data sample $\mathbf{y} \in \mathbb{R}^m$ up to a pre-specified distortion,

$$\hat{\mathbf{a}} = \arg \min_{\mathbf{u} \in \mathbb{R}^p} \|\mathbf{u}\|_0 \quad \text{s.t.} \quad \|\mathbf{y} - \mathbf{D}\mathbf{u}\|_2 \leq \epsilon, \quad (1.1)$$

This type of formulation is usually seen when sparsity is an *assumption* about the data, for example, in compressive sensing [7]. When sparsity is an explicit model constraint, a formulation such as the following one may be used,

$$\hat{\mathbf{a}} = \arg \min_{\mathbf{u} \in \mathbb{R}^p} \|\mathbf{y} - \mathbf{D}\mathbf{u}\|_2 \quad \text{s.t.} \quad \|\mathbf{u}\|_0 \leq \gamma, \quad (1.2)$$

where $\gamma \ll p$ indicates the desired *sparsity level* of the solution. In the statistics field of model selection [22, 27, 28], a Lagrangian form is used which provides a compound cost function that can be used to compare different competing models,

$$\hat{\mathbf{a}} = \arg \min_{\mathbf{u} \in \mathbb{R}^p} \|\mathbf{y} - \mathbf{D}\mathbf{u}\|_2^2 + \lambda \|\mathbf{u}\|_0. \quad (1.3)$$

Since problems (1.1)–(1.3) are non-convex and NP-hard, approximate solutions are sought. This is done either by using greedy methods such as Matching Pursuit (MP) [29], or by solving convex approximations to (1.1)–(1.3). For example, the convex approximation

to (1.2), commonly known as the *lasso* [24], is given by

$$\hat{\mathbf{a}} = \arg \min_{\mathbf{u} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{y} - \mathbf{D}\mathbf{u}\|_2 \quad \text{s.t.} \quad \|\mathbf{u}\|_1 \leq \tau, \quad (1.4)$$

which is also usually found in unconstrained (Lagrangian) form

$$\hat{\mathbf{a}} = \arg \min_{\mathbf{u} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{y} - \mathbf{D}\mathbf{u}\|_2^2 + \lambda \|\mathbf{u}\|_1, \quad (1.5)$$

which in turn is a convex approximation to (1.3). There exists a body of results showing that, under certain conditions on γ and \mathbf{D} , problems (1.1)–(1.3) can be solved exactly via their convex approximations or by using greedy algorithms such as MP (see for example [6, 7, 17]). In other cases, the objective is not to solve (1.2)–(1.3), but to guarantee some property of the estimated coefficients $\hat{\mathbf{a}}$. An example of this is the SURE estimator [30]. A recent generalization of this idea, the GSURE, is developed in [31]. However, if \mathbf{D} is arbitrary, no such choice exists. Also, if \mathbf{D} is orthonormal, the problem (1.4) admits a closed form solution obtained via the so-called *soft thresholding* operator [30]. However, again, for general \mathbf{D} , no such solution exists, and the search for efficient algorithms has been a hot topic recently, e.g., [9, 10, 11, 12].

1.2 Learning overcomplete dictionaries

Another key component in modern sparse modeling applications is the use of learned, possibly *overcomplete* dictionaries. By overcomplete we mean that the atoms in the dictionary form a redundant basis (i.e., linearly dependent, spanning the whole space). The use of non-overcomplete, learned dictionaries can be traced back to PCA [32] (Figure 1.3b). On the other hand, the introduction of overcomplete, pre-designed, dictionaries for image processing represented a breakthrough in the field, for example in image restoration, e.g. [33].

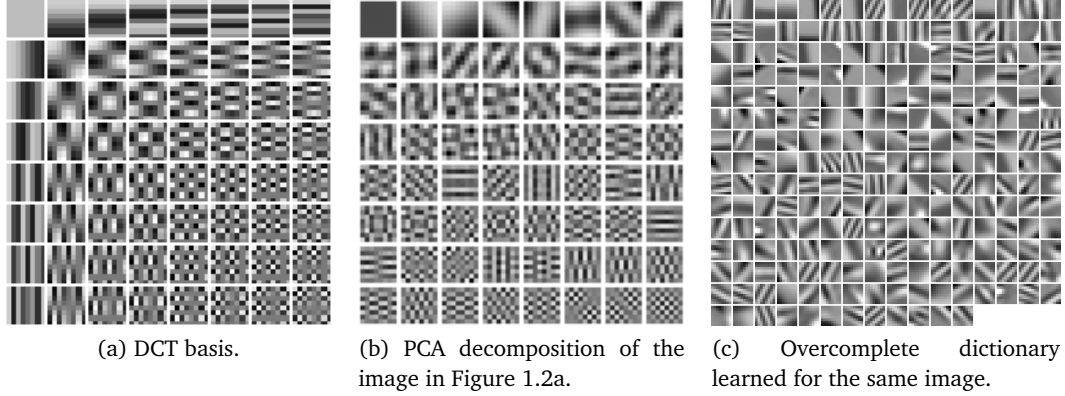


Figure 1.3: Example dictionaries for sparsely representing 8×8 image patches.

Combining these two concepts, learned overcomplete dictionaries have provided further significant improvements in these applications, e.g. [4]. Figure 1.3c shows an overcomplete dictionary which was learned for representing the image in Figure 1.2a using the algorithms described in Chapter 3.

Assuming that the parameter p is fixed, the problem of learning a dictionary from given data samples (ordered as columns of a matrix $\mathbf{Y} = [\mathbf{y}_1 | \mathbf{y}_2 | \dots | \mathbf{y}_n] \in \mathbb{R}^{m \times n}$) can be formulated as follows,

$$(\hat{\mathbf{A}}, \hat{\mathbf{D}}) = \arg \min_{\mathbf{A}, \mathbf{D}} \sum_{j=1}^n \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{a}_j\|_2^2 \quad \text{s.t.} \quad \|\mathbf{a}_j\|_r \leq \tau, \forall j, \quad \|\mathbf{d}_k\|_2 \leq 1, \forall k, \quad (1.6)$$

with $0 \leq r \leq 1$. The constraint $\|\mathbf{d}_k\|_2 \leq 1$, $k = 1, \dots, p$, is necessary to avoid an arbitrary decrease of the cost function by setting $\mathbf{D} \leftarrow \alpha \mathbf{D}$, $\mathbf{A} \leftarrow \frac{1}{\alpha} \mathbf{A}$, for any $\alpha > 1$. The cost function in (1.6) is non-convex in (\mathbf{A}, \mathbf{D}) , so that only local convergence can be guaranteed. This is usually achieved using alternate optimization in \mathbf{D} and \mathbf{A} . Algorithm 1 shows an example of this technique. We refer the reader to [4, 18, 19, 34] for other variants of this traditional approach. In any case the dictionary size p is a critical parameter that needs to be tuned to the particular data and application for which the model is learned.

Algorithm 1: Traditional dictionary learning.

Input: Data \mathbf{Y} , initial dictionary \mathbf{D}^0 , multiplier λ , tolerance ε , regularizer ϵ

Output: Local-optimum $(\hat{\mathbf{A}}, \hat{\mathbf{D}})$

initialize $\mathbf{D}^{(0)} = \mathbf{D}^0$, $t = 0$;

repeat

for $j = 1, \dots, n$ **do**

$t \leftarrow t + 1$;

$\mathbf{a}_j^{(t)} \leftarrow \arg \min_{\mathbf{u}} \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}^{(t-1)} \mathbf{u}\|_2^2 + \lambda \|\mathbf{u}\|_1$;

end

for $k = 1, \dots, p$ **do**

$\mathbf{u} \leftarrow \frac{1}{(\mathbf{A}\mathbf{A}^\top)_{kk} + \epsilon} (\mathbf{D}\mathbf{A}^{(t)}((\mathbf{A}^{(t)})^\top)_k - \mathbf{Y}((\mathbf{A}^{(t)})^\top)_k)$;

$\mathbf{d}_j^{(t)} \leftarrow \frac{1}{\min\{1, \|\mathbf{u}\|_2\}} \mathbf{u}$;

end

until $\frac{\|\mathbf{D}^{(t)} - \mathbf{D}^{(t-1)}\|_F}{\|\mathbf{D}^{(t)}\|_F} \leq \varepsilon$;

Set $\hat{\mathbf{A}} \leftarrow \mathbf{A}^{(t)}$;

Set $\hat{\mathbf{D}} \leftarrow \mathbf{D}^{(t)}$;

In Algorithm 1, the descent step on \mathbf{A} for fixed \mathbf{D} is solved for each j -th column of \mathbf{Y} using any of the aforementioned ℓ_1 sparse coding algorithms. The descent on \mathbf{D} for fixed \mathbf{A} , called *dictionary update* step, corresponds to a single iteration of a block-coordinate descent (where each block corresponds to an atom) using a scaled projected gradient method, where the scaling is the diagonalized Hessian of the fitting term w.r.t. to \mathbf{D} . Imposing sparsity on \mathbf{A} has as the side effect that $\mathbf{A}\mathbf{A}^\top$ can be very ill-conditioned, often with zeroes along the diagonal (note that $(\mathbf{A}\mathbf{A}^\top)_{kk} = 0$ if an atom is not used by any sample). Therefore, the scaling must be regularized as well by adding a regularization term $\epsilon > 0$ to the diagonalized Hessian, which is formally equivalent to adding the penalty term $\frac{\epsilon}{2} \|\mathbf{D}\|_F^2$ to the cost function in (1.6), so that the actual cost function being minimized is

$$(\hat{\mathbf{D}}, \hat{\mathbf{A}}) = \arg \min_{\mathbf{D}, \mathbf{A}} \sum_{j=1}^n \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{a}_j\|_2^2 + \lambda \|\mathbf{a}_j\|_1 + \frac{\epsilon}{2} \|\mathbf{D}\|_F^2, \quad \text{s.t.} \quad \|\mathbf{d}_k\|_2 \leq 1, \quad k = 1, \dots, p, \quad (1.7)$$

As with the other parameters, the choice of ϵ is critical to the success of the learning algorithm in producing a good model for the given data. As mentioned, dealing with such parameters in a principled and automatic way, as well as defining alternative ways of regularizing and robustifying the dictionary learning process, is the main subject of this thesis, which will be developed in the following chapters.

2 Adding structure to sparse models

The sparsity constraint has proved to be a powerful device for robustifying the representation of signals. This, together, with the flexibility of learned dictionaries, has provided breakthroughs in many signal processing applications. However, there are multiple scenarios in which imposing additional constraints to the produced representations, including the sparse codes and the dictionary itself, can result in further improvements.

This chapter summarizes our results and developments in this line of work, which span the publications [35, 36, 37, 38], reproduced here in appendixes A–D respectively. While the work described in Appendix B deals exclusively with adding structure to the sparse codes, appendixes A, C and D approach the issue of adding structure to the dictionaries in different ways. This division is reflected in the structure of the remainder of this chapter.

2.1 Structured sparse coding

Besides requiring sparse solutions, the need to add further constraints to the sparse codes arises in several scenarios. We refer to *structured sparse coding* when additional constraints are imposed on the possible supports of the sparse codes [39, 40]. When prior information about the problem allows one to impose such constraints, the resulting sparse representations are often more stable and robust than the ones provided by traditional formulations such as (1.4). One example is the Group Lasso [39], which generalizes the Lagrangian formulation of Lasso [24] by imposing sparsity to groups of coefficients (meaning that only a few groups of coefficients may contain non-zero coefficients— within the groups, no sparsity

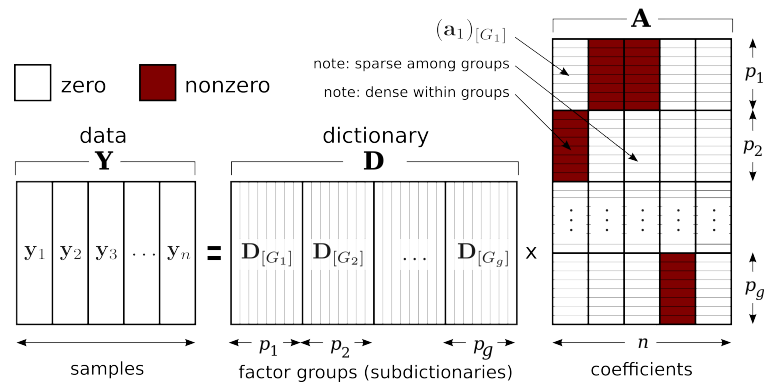


Figure 2.1: Group-sparse data model. Several data samples (columns of the matrix \mathbf{Y}) are here encoded using group-sparse codes (columns of the matrix \mathbf{A}).

is assumed – see Figure 2.1). Define $\mathcal{G} = \{G_1, \dots, G_g\}$ to be a partition of the atom index set $\{1, \dots, p\}$. The problem of encoding a data sample \mathbf{y}_j from \mathbf{Y} via the Group Lasso model can be stated as follows,

$$\hat{\mathbf{a}}_j = \arg \min_{\mathbf{u} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{u}\|_2^2 + \sum_{r=1}^g \|\mathbf{u}_{[G_r]}\|_2 \leq \tau. \quad (2.1)$$

Here the notation $\mathbf{u}_{[G_r]}$ indicates the sub-vector whose indexes belong to the index set G_r . As before, the data samples \mathbf{y}_j and their corresponding sparse codes $\mathbf{a}_j, j = 1, \dots, n$, are assembled as columns of the data and coefficients matrices \mathbf{Y} and \mathbf{A} respectively (see Figure 2.1). For example, in the context of Compressive Sensing, if the sparse vector \mathbf{a}_j to be recovered from \mathbf{y}_j exhibits this kind of group pattern, the original recovery conditions [5, 6, 7, 17] can be significantly relaxed [41, 42, 43, 44].

2.1.1 Hierarchical sparsity

One particular situation where group sparsity is a natural assumption is when one wants to represent signals \mathbf{y} which are a linear mixture of several other signals, $\mathbf{y} = \sum_{r=1}^g \mathbf{x}^r$,

each of which can be efficiently represented in terms of a corresponding dictionary \mathbf{D}^r . Using this prior information, we expect \mathbf{y} to be efficiently encoded in terms of the dictionary $\mathbf{D} = [\mathbf{D}_{[G_1]} | \mathbf{D}_{[G_2]} | \dots | \mathbf{D}_{[G_g]}]$ where $\mathbf{D}_{[G_r]} = \mathbf{D}^r$. In the setting of Appendix B, the focus is on the source separation problem, where one wants to estimate each unmixed signal \mathbf{x}^r from \mathbf{y} (in particular, we were interested in the problem of separating music instruments playing simultaneously during a recording). However, a straightforward application of Group Lasso here presents us with an inconsistency problem, since the Group Lasso produces dense representations within each group, (as seen in Figure 2.1), while our dictionaries are trained for representing each \mathbf{x}^r sparsely. In order to enforce sparsity within the groups, the regularizers from the Group Lasso and the Lasso are combined, resulting in the *Hierarchical Lasso*,

$$\hat{\mathbf{a}}_j = \arg \min_{\mathbf{u} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{u}\|_2^2 + \lambda_2 \sum_{r=1}^g \|\mathbf{u}_{[G_r]}\|_2 + \lambda_1 \|\mathbf{u}\|_1. \quad (2.2)$$

This is complemented with the development of an efficient special-purpose optimization algorithm for solving problem (2.2), as well as a generalization of the exact recovery conditions for traditional [5, 6, 7, 17] and group sparse coding [41, 42, 43, 44] to the hierarchical case. These developments, as well as experiments that show the practical benefits of adding this new formulation, are detailed in Appendix B.

Of course, as with models such as Lasso and Group Lasso, the optimal parameters λ_1 and λ_2 are application and data dependent. As mentioned in the introduction, we defer the treatment of this problem (albeit not for the particular case of the Hierarchical Lasso) to the next chapter.

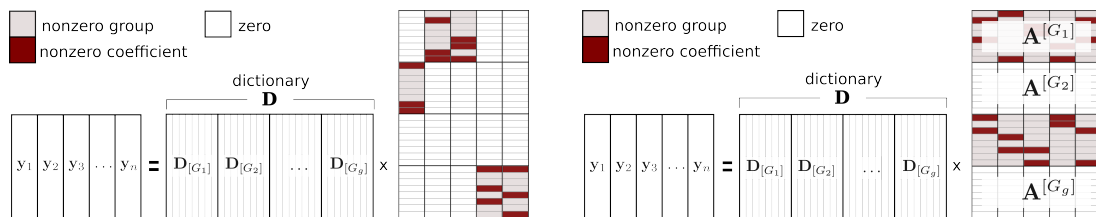


Figure 2.2: Sparsity patterns induced by HiLasso (left) and C-HiLasso (right) model selection programs. Notice that the C-HiLasso imposes the same group-sparsity pattern in all the samples (same class), whereas the in-group sparsity patterns can vary between samples (samples themselves are different).

2.1.2 Collaborative hierarchical sparsity

In numerous applications, one expects that the samples \mathbf{y}_j in \mathbf{Y} all share the same active components from the dictionary, that is, that the indexes of the nonzero coefficients in \mathbf{a}_j are the same for all $j = 1, \dots, n$. Imposing such dependency in the ℓ_1 regularized regression problem gives rise to the so called collaborative (also called “multitask” or “simultaneous”) sparse coding problem [41, 44, 45, 46]. This model is given by

$$\min_{\mathbf{A} \in \mathbb{R}^{p \times n}} \frac{1}{2} \|\mathbf{Y} - \mathbf{DA}\|_F^2 + \lambda \sum_{k=1}^p \|\mathbf{a}^k\|_2, \quad (2.3)$$

where $\mathbf{a}^k \in \mathbb{R}^n$ is the k -th’s row of \mathbf{A} , that is, the vector of the n different values that the coefficient associated to the k -th atom takes for each sample $j = 1, \dots, n$. If we now extend this idea to the Group Lasso, we obtain a *collaborative Group Lasso* (C-GLasso) formulation,

$$\min_{\mathbf{A} \in \mathbb{R}^{p \times n}} \frac{1}{2} \|\mathbf{Y} - \mathbf{DA}\|_F^2 + \lambda \sum_{r=1}^g \left\| \mathbf{A}^{[G_r]} \right\|_F \quad (2.4)$$

where $\mathbf{A}^{[G]}$ is the sub-matrix formed by all the rows belonging to group G . This regularizer is the natural collaborative extension of the regularizer in (2.1).

Again, motivated by the case of source (instrument) separation in mixed signals (audio

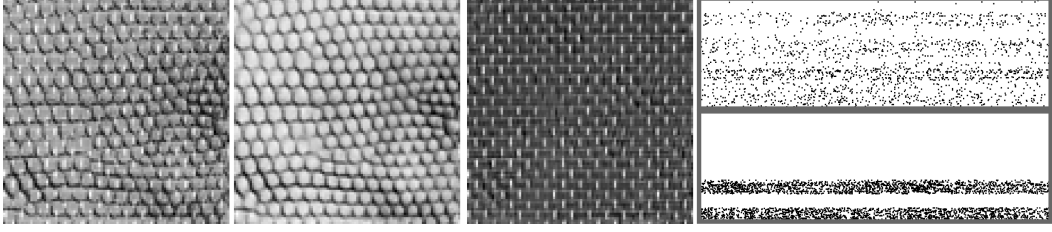


Figure 2.3: Texture separation results. Left to right: sample mixture, corresponding C-HiLasso separated textures, and comparison of the active set diagrams obtained by the Lasso (as in Figure 2.5). The one for Lasso is shown on top, where all groups are wrongly active, and the one for C-HiLasso on bottom, showing that only the two correct groups are selected.

recordings), the work described in Appendix B takes an additional step, adding collaborative coding to the hierarchical model. The combined model that we propose, *C-HiLasso*, is given by

$$\min_{\mathbf{A} \in \mathbb{R}^{P \times n}} \frac{1}{2} \|\mathbf{Y} - \mathbf{DA}\|_F^2 + \lambda_2 \sum_{r=1}^g \left\| \mathbf{A}^{[G_r]} \right\|_F + \lambda_1 \sum_{j=1}^n \|\mathbf{a}_j\|_1. \quad (2.5)$$

The sparsity pattern obtained using (2.5) is shown in Figure 2.2(right). The C-GLasso is a particular case of our model when $\lambda_1 = 0$. On the other hand, one can obtain independent Lasso solutions for each \mathbf{y}_j , $j = 1, \dots, n$ by setting $\lambda_2 = 0$. We see that (2.5) encourages all the signals to share the same groups (classes), while the active set inside each group is signal dependent. We thereby obtain a collaborative hierarchical sparse model, with collaboration at the class level (all signals collaborate to identify the classes), and freedom at the individual levels inside the class to adapt to each particular signal. A graphical example of the effectiveness of this modeling approach is shown in Figure 2.3, where we applied the C-HiLasso model to separate mixtures of two textures. Here \mathbf{Y} contains all 12×12 overlapping patches of the mixture texture image.

This new model is also particularly well suited, for example, when the data vectors have missing components. In this case combining the information from all the samples is very important in order to obtain a correct representation and model (group) selection. This

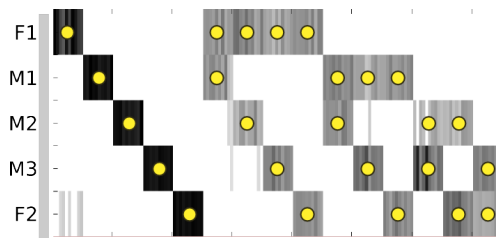


Figure 2.4: Speaker identification results. Each column corresponds to the sources identified for a specific time frame, the true ones marked by yellow dots. The vertical axis indicates the estimated activity of the different sources, where darker colors indicate higher energy. For each possible combination of speakers, 10 frames (15 seconds of audio) were evaluated.

can be done by slightly changing the data term in (2.5) so that only observed elements are taken into account in the error term,

$$\min_{\mathbf{A} \in \mathbb{R}^{p \times n}} \frac{1}{2} \|\mathbf{H} \odot (\mathbf{Y} - \mathbf{DA})\|_F^2 + \lambda_2 \sum_{r=1}^g \|\mathbf{A}^{[G_r]}\|_F + \lambda_1 \sum_{j=1}^n \|\mathbf{a}_j\|_1. \quad (2.6)$$

where \odot denotes Schur (element-wise) product, and \mathbf{H} is a binary mask which contains zeroes at the locations of the missing elements. Figure 2.5 shows the results of applying this model to recover a set of digits from their observed mixtures with 60% of the observed mixture samples missing, providing evidence on the power of this new modeling approach.

We also applied the C-HiLasso model to the source identification problem, where the goal is to detect the sources that are active in the observed mixture. In this case the observed signal is an audio recording, and the sources correspond to different human speakers. The results are summarized in Figure 2.4. See Appendix B for more details on this and the other experiments.

As for the HiLasso, we developed efficient algorithms for solving problems (2.5) and (2.6). Again, details of this are given in Appendix B.

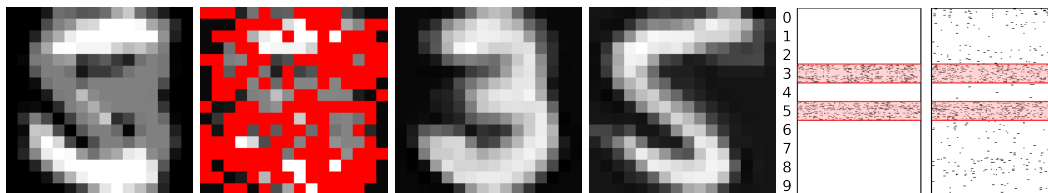


Figure 2.5: Example of recovered digits (3 and 5) from a mixture with 60% of missing components. From left to right: noiseless mixture, observed mixture with missing pixels highlighted in red, recovered digits 3 and 5, and active set recovered for all samples using the C-HiLasso and Lasso respectively. In the last two figures, the active sets are represented as in Figure 2.5. The coefficients blocks for digits “3” and “5” are marked as pink bands. Notice that the C-HiLasso exploits efficiently the hypothesis of collaborative group-sparsity, succeeding in recovering the correct active groups in all the samples. The Lasso, which lacks this prior knowledge, is clearly not capable of doing so, and active sets are spread over all the groups.

2.2 Structured dictionary learning

The only constraint in traditional dictionary learning applications is that their atoms should have unit norm. As mentioned in Chapter 1.2, this is imposed to avoid an arbitrary decrease of the cost function by a trivial rescaling of the atoms.

There are however various possible reasons to impose further constraints on the learned dictionaries. For example, all the sparse recovery conditions developed so far (see [17] for a review) depend on geometrical properties of the dictionaries. In particular, one such property is the *cumulative mutual coherence*,

$$\bar{\mu}_L(\mathbf{D}) := \max \left\{ \left\{ \max_{i \notin J} \sum_{j \in J} |\mathbf{d}_i^T \mathbf{d}_j| : J \subseteq \{1, \dots, p\} \right\} : |J| = L \right\}, \quad (2.7)$$

that is, the maximum absolute correlation between a fixed atom and L other atoms of \mathbf{D} (we use the shorthand $\bar{\mu}(\mathbf{D})$ for the particular case of $L = p - 1$). Loosely speaking, the more distinguishable the atoms in a dictionary are (smaller mutual coherence), the easier it is to recover the sparsest representation (1.2) via approximate methods such as ℓ_1 convex

relaxations, or greedy pursuit. For example, in [6, Theorem B] it is established that, to recover an γ -sparse “signal” \mathbf{a} (sparse coefficients in our notation) by either approximate method, the following condition must hold,

$$\bar{\mu}_{(\gamma-1)}(\mathbf{D}) + \bar{\mu}_{(\gamma)}(\mathbf{D}) < 1.$$

Such recoverability properties are important in sparse modeling when \mathbf{a} and, in particular, its support $\text{supp}(\mathbf{a})$, are features used for example in classification.

Another geometrical property of the dictionary \mathbf{D} which is relevant to sparse modeling applications is its spectral norm $\rho(\mathbf{D})$, that is, its largest singular value $\rho(\mathbf{D}) = \sqrt{\lambda_{\max}(\mathbf{D}^T \mathbf{D})}$. It is well known that the speed of convergence of ℓ_1 sparse coding algorithms from the family of Iterative Shrinkage/Thresholding (IST) [9, 12] depends on this value (see [47] for a review).

Both the incoherence $\bar{\mu}(\mathbf{D})$ and $\rho(\mathbf{D})$ are related via the following inequality (see Supplementary Material at the end of Appendix C for a proof)

$$\rho(\mathbf{D}) \leq \sqrt{1 + \bar{\mu}(\mathbf{D})}.$$

Controlling the incoherence has been investigated in the compressive sensing literature [47]. In our case, we propose to add a new term to the traditional dictionary learning formulation (1.6) which imposes a small Frobenius norm of the Gram matrix $\mathbf{D}^T \mathbf{D}$, thus indirectly reducing both $\rho(\mathbf{D})$ and $\bar{\mu}(\mathbf{D})$,

$$\min_{(\mathbf{A}, \mathbf{D})} \frac{1}{n} \sum_{j=1}^n \left[\|\mathbf{y}_j - \mathbf{D} \mathbf{a}_j\|_2^2 + \lambda \psi(\mathbf{a}_j) \right] + \eta \|\mathbf{D}^T \mathbf{D}\|_F^2, \quad (2.8)$$

where $\psi(\cdot)$ is a sparsity inducing regularizer such as the ℓ_1 norm.

For proper choices of the parameter $\eta > 0$, the resulting dictionaries exhibit a cumulative coherence $\bar{\mu}$ usually around 5 times smaller than those observed in dictionaries learned without the additional term, while retaining similar representative power (in the sense that the sum of the fitting and regularization terms remains approximately the same), leading to a significant improved ability to recover sparse signals, and an important acceleration of the IST family of methods. See Appendix C for details.

The recent extension of sparse recovery results to group-sparse signals [42, 43, 44] motivates a natural extension of the incoherence concept, the block-coherence, which can be defined as follows,

$$\mu_B(\mathbf{D}) := \frac{1}{m} \max \left\{ \rho \left(\mathbf{D}_{[G_r]}^\top \mathbf{D}_{[G_l]} \right) \right\}.$$

In the work described in Appendix A, we promote block-incoherence while learning dictionaries for different classes in order to improve the classification performance of a sparse model based classification algorithm. In this case we have g classes, each one with an associated dictionary \mathbf{D}^r , $r = 1, \dots, g$ which is learned from a corresponding training dataset \mathbf{Y}^r . We adapt the dictionaries to the classes by minimizing the standard ℓ_1 regularized squared error energy. Block-incoherence is simultaneously promoted on all dictionaries in a similar way to (2.8), by means of minimizing the Frobenius norm of $(\mathbf{D}^r)^\top \mathbf{D}^l$, $r \neq l$. This results in the following problem,

$$\min_{\{(\mathbf{A}^r, \mathbf{D}^r): r=1 \dots g\}} \sum_{r=1}^g \left\{ \sum_{j=1}^{n_r} \left[\frac{1}{2} \left\| \mathbf{y}_j^r - \mathbf{D}^r \mathbf{a}_j^r \right\|_2^2 + \lambda \|\mathbf{a}_j^r\|_1 \right] + \eta \sum_{l \neq r} \left\| (\mathbf{D}^r)^\top \mathbf{D}^l \right\|_F^2 \right\}. \quad (2.9)$$

2.3 Imposing atom smoothness

The previously described work adds structure to the dictionaries by imposing restrictions between the atoms. A complimentary idea is to impose structure to the atoms themselves.

This is a natural result of applying the MDL-based framework described in Appendix D, which is the subject of the next chapter. However, we can consider this idea aside from the MDL interpretation as a way to robustify the process of learning dictionaries using a few highly corrupted samples. We start by recalling the effective cost function (1.7) that is approximately minimized in ℓ_1 traditional dictionary learning frameworks,

$$(\hat{\mathbf{D}}, \hat{\mathbf{A}}) = \arg \min_{\mathbf{D}, \mathbf{A}} \sum_{j=1}^n \left[\frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{a}_j\|_2^2 + \lambda \|\mathbf{a}_j\|_1 \right] + \frac{\epsilon}{2} \|\mathbf{D}\|_F^2, \quad \text{s.t.} \quad \|\mathbf{d}_k\|_2 \leq 1, \quad k = 1, \dots, p.$$

Here the dictionary regularization term $\frac{\epsilon}{2} \|\mathbf{D}\|_F^2$ is a simple regularization that further stabilizes the norms of the atoms. This formulation, however, neglects the fact that learned atoms often present features that are similar to those of the original data. For example, the piecewise smoothness of small image patches is to be expected in the atoms of learned dictionaries for such data (see Figure 1.3c).

Applying the ideas developed in Appendix D, Section 4.3, an alternative dictionary regularization term can be formulated which takes such smoothness into account. For example,

$$\begin{aligned} (\hat{\mathbf{D}}, \hat{\mathbf{A}}) = \arg \min_{\mathbf{D}, \mathbf{A}} \sum_{j=1}^n \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{a}_j\|_2^2 + \lambda \|\mathbf{a}_j\|_1 + \epsilon \sum_{k=1}^p \|\mathbf{W}\mathbf{d}_k\|_1 \\ \text{s.t.} \quad \|\mathbf{d}_k\|_2 \leq 1, \quad k = 1, \dots, p. \end{aligned} \quad (2.10)$$

where \mathbf{W} is a matrix that maps atoms \mathbf{d}_k to prediction residual vectors which are modeled as Laplacian IID sequences (thus resulting in a ℓ_1 regularization term, see Appendix D for details). For example, in the case of imposing piecewise-smoothness prior on the atoms described in Appendix D, the matrix \mathbf{W} produces the residuals of applying a bilinear predictor to each atom \mathbf{d}_k . Another alternative is to replace the ℓ_1 term by an ℓ_2 norm in the dictionary regularization term of (2.10), which in practice is easier to optimize than the ℓ_1 and

already results in significant improvements over (1.7). See the Supplementary Material at the end of Appendix D for details on this.

3 An information-theoretic view of learned overcomplete sparse models

As mentioned in the introduction, one of the main goals of this thesis is to gain a better understanding of sparse models as tools for describing given data. This chapter summarizes the work done in this thesis along this line of research [37, 38, 48], the details of which are given in appendixes C–E. We begin by reproducing here the motivating example given in Appendix D.

3.1 Issues with traditional sparse models: a motivating example

Consider the K-SVD-based [4] sparse image restoration framework [49]. This is an ℓ_0 -based dictionary learning framework, which approximates (1.6) for the case $r = 0$ by alternate minimization. In the case of image denoising, the general procedure can be summarized as follows:

1. An initial, *global* dictionary \mathbf{D}_0 is learned using training samples for the class of data to be processed (in this case small patches of natural images), for example using Algorithm 1. The user must supply a patch width w , a dictionary size p and a value for the sparsity-inducing penalty λ .
2. The noisy image \mathbf{J} is decomposed into overlapping $w \times w$ patches (one patch per pixel of the image), and assembled as the columns of the data matrix \mathbf{Y} from which \mathbf{D} is

further adapted using the following denoising variant of (1.6),

$$(\hat{\mathbf{D}}, \hat{\mathbf{A}}) = \arg \min_{\mathbf{D}, \mathbf{A}} \sum_{j=1}^n \|\mathbf{a}_j\|_0 \quad \text{s.t.} \quad \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{a}_j\|_2^2 \leq C\sigma^2, j = 1, \dots, n, \\ \|\mathbf{d}_k\|_2 = 1, k = 1, \dots, p. \quad (3.1)$$

Here the user must further supply a constant C (in [49], it is 1.32), the noise variance σ^2 , and the number of iterations J of the optimization algorithm, which is usually kept small to avoid over-fitting (the algorithm is *not* allowed to converge).

3. An intermediate result $\tilde{\mathbf{I}}$ is constructed by assembling the patches in $\hat{\mathbf{Y}} = \hat{\mathbf{D}}\hat{\mathbf{A}}$ into the corresponding original positions of the image.
4. The final denoised image $\hat{\mathbf{I}}$ is computed as $\hat{\mathbf{I}} = (\eta + 1)^{-1}(\eta\tilde{\mathbf{I}} + \mathbf{J})$, where $\eta > 0$ controls the amount of original noisy image that is added back to produce the result.

Despite the good results obtained for natural images, some aspects of this method are not satisfactory:

- Several parameters ($w, p, \lambda, C, J, \eta$) need to be tuned. *There is no interpretation, and therefore no justifiable choice for these parameters, other than maximizing the empirical performance of the algorithm (according to some metric, in this case PSNR) for the data at hand.*
- The effect of such parameters on the result is shadowed by the effects of later stages of the algorithm and their associated parameters (e.g. overlapping patch averaging, amount η of noisy image in the final image). *There is no fundamental way to optimize each stage separately.*

As a partial remedy to the first problem, Bayesian sparse models were developed (e.g., [20]) where some of these parameters are assigned prior distributions which are then

learned from the data. However, this approach still does not provide objective means to compare different models (with different priors, for example). Further, the Bayesian technique implies having to repeatedly solve possibly costly optimization problems, increasing the computational burden of the application.

This chapter summarizes the work done towards defining a unified framework for solving *all* of the above issues in a principled way.

3.2 Universal models for priors with unknown parameters

The above example is actually the one that motivated this thesis in the first place, and the results summarized in this section, the full details of which are provided in Appendix C, are among the first ones obtained in this thesis towards the objectives stated at the beginning of this chapter. The focus of this work is on the role of the regularization term that controls the sparsity level in (1.5), which we recognized as one of the most difficult to tune in order to obtain good results, for example in the work described in Appendix A.

The idea here is to re-interpret the traditional sparse coding and dictionary learning problems as ones of codelength minimization, and then apply tools from universal coding theory [21] to replace traditional regularizers such as ℓ_1 with one that can be guaranteed to be better suited to the prior information about sparse coefficients arising from the decomposition of any given data sample within the class of signals of interest.

For this, suppose that we have a fixed dictionary \mathbf{D} and that we want to use it to compress an image which is first decomposed into non-overlapping blocks which are then arranged as the columns of a data matrix \mathbf{Y} . The image is then encoded either losslessly by encoding the reconstruction coefficients \mathbf{A} and the residual $\mathbf{E} = \mathbf{Y} - \mathbf{DA}$, or in a lossy manner, by obtaining a good approximation $\mathbf{Y} \approx \mathbf{DA}$ and encoding only \mathbf{A} . Consider for example the

latter case. Most modern compression schemes consist of two parts: a *probability assignment stage* where the data, in this case \mathbf{A} , is assigned a probability $P(\mathbf{A})$, and an *encoding stage* where a code $C(\mathbf{A})$ of length $L(\mathbf{A})$ bits is assigned to the data given its probability, so that $L(\mathbf{A})$ is as short as possible and \mathbf{A} can be uniquely recovered from $C(\mathbf{A})$. The techniques known as Arithmetic and Huffman coding provide the best possible solution for the encoding step, which is to approximate the Shannon ideal codelength $L(\mathbf{A}) = -\log P(\mathbf{A})$ [50, Chapter 5]. Therefore, modern compression theory deals with finding the coefficients \mathbf{A} that maximize $P(\mathbf{A})$, or, equivalently, that minimize $-\log P(\mathbf{A})$.¹ Now, to encode \mathbf{Y} lossily, we obtain coefficients \mathbf{A} such that each data sample \mathbf{y}_j is approximated up to a certain ℓ_2 distortion ϵ , $\|\mathbf{y}_j - \mathbf{D}\mathbf{a}_j\|_2^2 \leq \epsilon$. Therefore, given a model $P(\mathbf{a})$ for a vector of reconstruction coefficients, and assuming that we encode each sample independently, the optimum vector of coefficients \mathbf{a}_j for each sample \mathbf{y}_j will be the solution to the optimization problem

$$\mathbf{a}_j = \arg \min_{\mathbf{u}} -\log P(\mathbf{u}) \quad \text{s.t.} \quad \|\mathbf{y}_j - \mathbf{D}\mathbf{u}\|_2^2 \leq \epsilon. \quad (3.2)$$

For example, for the choice $P(\mathbf{a}) \propto e^{-\theta\|\mathbf{a}\|_1}$ (Laplacian IID coefficients model), (3.2) coincides with the Basis Pursuit Denoising (BPDN) [8] sparse coding formulation. Suppose now that we want to perform lossless compression. In this case we also need to encode the reconstruction residual $\mathbf{e}_j = \mathbf{y}_j - \mathbf{D}\mathbf{a}_j$, which provides a conditional description of \mathbf{y}_j given \mathbf{a}_j , that is, $P(\mathbf{e}_j) = P(\mathbf{y}_j|\mathbf{a}_j)$. Since $P(\mathbf{y}_j, \mathbf{a}_j) = P(\mathbf{y}_j|\mathbf{a}_j)P(\mathbf{a}_j)$, the combined codelength will be

$$L(\mathbf{y}_j, \mathbf{a}_j) = -\log P(\mathbf{y}_j, \mathbf{a}_j) = -\log P(\mathbf{y}_j|\mathbf{a}_j) - \log P(\mathbf{a}_j). \quad (3.3)$$

¹Note that, in general, the reconstruction coefficients are considered real numbers, so that the probability of \mathbf{A} under any continuous probability distribution will be $P(\mathbf{A}) = 0$. In order to use such distributions as our models for the data, we assume that the coefficients in \mathbf{A} are quantized to a precision Δ , small enough for the associated density function $f(a)$ to be approximately constant in any interval $[a - \Delta/2, a + \Delta/2]$, $a \in \mathbb{R}$, so that we can approximate $P(a) \approx \Delta f(a)$, $a \in \mathbb{R}$. Under these assumptions, $-\log P(a) \approx -\log f(a) - \log \Delta$, and the effect of Δ on the codelength produced by any model is the same. Therefore, we will omit Δ in the sequel, and treat density functions and probability distributions interchangeably as $P(\cdot)$. Of course, in real compression applications, Δ needs to be tuned.

Therefore, obtaining the best coefficients \mathbf{a}_j amounts to solving $\min_{\mathbf{a}} L(y_j, \mathbf{a}_j)$, which for the choices $P(\mathbf{y}|\mathbf{a}) \propto e^{-\frac{1}{2\sigma^2}\|\mathbf{y}-\mathbf{D}\mathbf{a}\|_2^2}$ (Gaussian error model) and $P(\mathbf{a}) \propto e^{-\theta\|\mathbf{a}\|_1}$, (Laplacian coefficients model) leads to the Lagrangian form of the Lasso sparse coding (1.5) with $\lambda = 2\sigma^2\theta$.

In the context of codelength minimization, these priors (Gaussian, Laplacian) have unknown parameters (σ^2, θ) that need to be defined in order to have a complete description of the problem, or either they have to be part of the optimization as well. The same happens with the maximum allowed distortion ϵ in (3.2). In particular, in many applications, σ^2 is usually assumed to be known, or easily estimated from the data. The distortion ϵ is also usually set to a small multiple of σ^2 , for example $1.32m\sigma^2$ in [49]. Thus, the critical parameter to define in this scenario is θ . The approach in this work is to bypass this choice and replace $P(\mathbf{a}; \theta)$ (here the dependency on θ made explicit) by a model that is capable of fitting \mathbf{a} almost as well as *any* instance of $P(\mathbf{a}; \theta)$ that can be obtained for any θ . Such models are known as *universal models*, and constitute a major tool in modern compression theory (see for example [51]). There are several ways to construct such universal models. In our case, we build an universal model for the family of Laplacian distributions $P(\mathbf{a}; \theta)$ by means of a *convex mixture model*,

$$Q(\mathbf{a}) = \int_{\Theta} P(\mathbf{a}|\theta)w(\theta)d\theta,$$

where the mixing function $w(\theta)$ specifies the weight of each model in the mixture. Being a convex mixture implies that $w(\theta) \geq 0$ and $\int_{\Theta} w(\theta)d\theta = 1$, thus $w(\theta)$ is itself a probability measure over Θ .

Based on such universal models, we then re-define the sparse coding (and dictionary

learning) problems in terms of the resulting regularizers, which we call *universal regularizers*, $\psi(\mathbf{a}) := -\log Q(\mathbf{a})$. These regularizers share features with other well known non-convex regularizers such as those based on the $\ell_r, 0 < r < 1$ family of pseudo-norms and their associated probability distributions, the Generalized Gaussian models (see e.g.[52]). For example, one of the models developed in Appendix C, the MOE (Mixture of Exponentials), is given by

$$Q_{\text{MOE}}(a|\beta, \kappa) = \frac{1}{2}\kappa\beta^\kappa(|a| + \beta)^{-(\kappa+1)}, \quad a \in \mathbb{R}, \quad (3.4)$$

with κ, β being *non-informative* hyper-parameters. In the context of universal modeling, this means that $Q_{\text{MOE}}(\mathbf{a}|\beta, \kappa)$ will be a universal model regardless of the choice of such hyper-parameters, which in turn means that it will be a “good model” for encoding any instance of \mathbf{a} . The associated regularizer is given by,

$$\psi_{\text{MOE}}(\mathbf{a}) = -\log Q_{\text{MOE}}(\mathbf{a}|\kappa, \beta) = 2(\kappa + 1) \sum_{k=1}^p \log(|a_k| + \beta).$$

A regularizer with the same form was proposed in [53] as an alternative view of the re-weighted ℓ_1 sparse coding method developed there. In our case, the path was exactly the opposite. We arrived at such regularizer using universal modeling arguments, and then derived an optimization technique to approximately solve the resulting non-convex MOE-based sparse coding problem,

$$\hat{\mathbf{a}}_j = \arg \min_{\mathbf{u}} \frac{1}{2\sigma^2} \|\mathbf{y}_j - \mathbf{D}\mathbf{u}\|_2^2 + 2(\kappa + 1) \sum_{k=1}^p \log(|u_k| + \beta), \quad (3.5)$$

which is based on iterative local linear approximations (LLA) of the cost function (see Appendix C for details). This approximation method was proposed independently in [54]. The LLA technique, when applied to (3.5), results in the exact same re-weighted ℓ_1 formulation

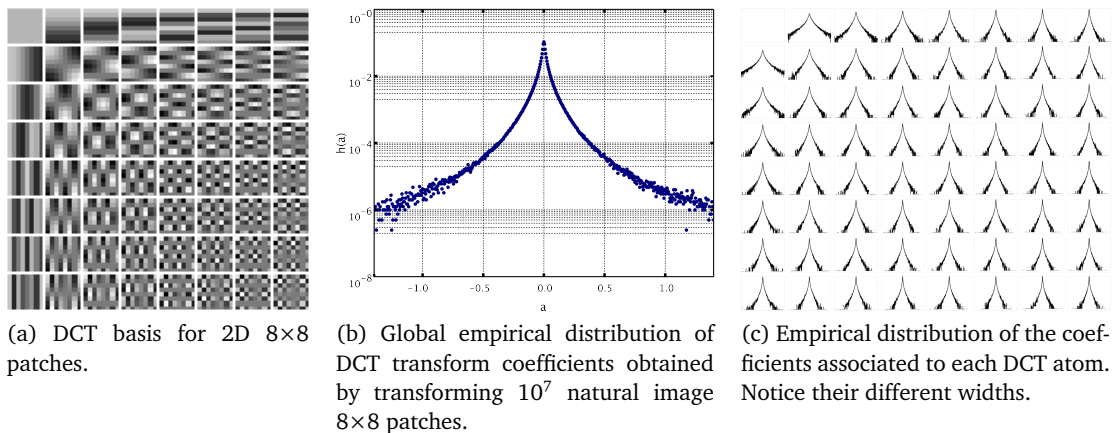


Figure 3.1: Distribution of DCT coefficients.

proposed in [53]. In our case, however, the parameters (κ, β) can be interpreted as hyper-parameters of the distribution underlying the regularizer and, as such, they can be fitted by Maximum Likelihood (for example) to the data. In contrast, in [53] these parameters have no associated interpretation, and thus need to be hand-tuned.

An additional benefit of the use of a universal regularizer is that it also works well when the underlying distribution parameter θ has a different value θ_k for each k -th coefficient in \mathbf{a} (that is, in an independent, non-identically distributed model for the coefficients). This is of particular interest to the case of image coding since it is empirically known that coefficients associated to different atoms have different distributions. This is well known for the case of the DCT transform [55] (see Figure 3.1), and, more importantly, has also been observed to hold for the case of learned dictionaries in the work described in Appendix C, where empirical results on this are reported.

Due to their universality, the proposed regularizers are at the same time robust, in the sense that they perform well for a wide range of signals (within the class of signals for which they were designed), and flexible, since they also cope well with the extended model consisting of a different Laplacian parameter θ_k for each coefficient $a_k, k = 1, \dots, p$, without adding extra parameters to the actual regularizer. With these two features combined, the

proposed universal regularizers yield improved empirical sparse recovery rates (confirmed analytically for example in [56]), which in turn improves the performance of sparse models in tasks such as image denoising or patch-based image classification. Again, full details on this are given in Appendix C.

3.3 Sparse models, model selection, and MDL

Using the universal models developed in the previous work (which are specifically designed for sparse modeling of images) as the main building block, the next step towards solving the issues described in Section 3.1 is the development of a framework for automatically choosing the best sparse model, along with *all* the associated parameters, for a given data or data class. This is the subject of the work described in Appendix D.

In general, the problem of selecting the best model \hat{M} , out of a set of candidate models \mathcal{M} , for representing a given dataset \mathbf{Y} , is known as the *model selection problem*. In this sense, the work described in Appendix D provides practical realizations of the three main ingredients for performing model selection for the specific case of learned sparse models, namely:

1. Define the class \mathcal{M} of learned sparse models, using as much prior information about it so that searches within the class can be performed efficiently.
2. Define an objective criterion for selecting the model $\hat{M} \in \mathcal{M}$ which better fits the given data \mathbf{Y} .
3. Define and implement efficient algorithms for performing such selection.

3.3.1 The class of learned overcomplete sparse models

The first ingredient, the model class \mathcal{M} , is defined to be the union of nested family of model classes $\mathcal{M} = \bigcup_{p=0}^{+\infty} \mathcal{M}(p)$, with

$$\mathcal{M}(p) := \{(\mathbf{E}, \mathbf{A}, \mathbf{D}) : \mathbf{Y} = \mathbf{D}\mathbf{A} + \mathbf{E}, \mathbf{D} \in \mathbb{R}^{m \times p}\} \quad (3.6)$$

where p is the size (number of atoms) of the dictionaries in \mathcal{M} . Such models are nested, in the sense that $\mathcal{M}(p-1) \subset \mathcal{M}(p)$.² This is an important feature of \mathcal{M} which allows for efficient search methods.

3.3.2 The description length of learned sparse models

The second ingredient, that is, the objective function used for selecting the best model within \mathcal{M} , is provided by the Minimum Description Length (MDL) principle for model selection.

In MDL, given a family or model class \mathcal{M} of candidate models indexed by a parameter M , and a data \mathbf{Y} , the best model $\hat{M} \in \mathcal{M}$ is the one that can be used to describe \mathbf{Y} completely (including the parameters M themselves) with the fewest number of bits,

$$\hat{M} = \arg \min_{M \in \mathcal{M}} L(\mathbf{y}, M), \quad (3.7)$$

where $L(\mathbf{y}, M)$ is a *codelength assignment function* which defines the theoretical codelength required to describe (\mathbf{y}, M) *uniquely*, and which is a key component of any MDL-based framework. The underlying idea of MDL is that *compressibility is a good indirect way of measuring the ability of a model to capture regularity from the data*. Common practice in MDL uses the *Ideal Shannon Codelength Assignment* [50, Chapter 5] to define $L(\mathbf{Y}, M)$ in terms of a *probability assignment* $P(\mathbf{Y}, M)$ as $L(\mathbf{Y}, M) = -\log P(\mathbf{Y}, M)$ (all logarithms will be

²In order to be formally correct, this statement requires some conventions which are detailed in Appendix D.

assumed on base 2 hereafter). In this way, the problem of choosing $L(\cdot)$ becomes one of choosing a suitable probability model for (\mathbf{Y}, M) .

As efficient probability assignment functions are needed, MDL relies heavily on the use of universal models such as the ones developed in Appendix C. It is in this sense that the work described there and summarized at the beginning of this chapter is used here as one of the main building blocks, in particular, for describing non-zero sparse coefficients efficiently. In addition to these, Appendix D introduces universal models for describing the other parts of the sparse model as well, including the approximation error \mathbf{E} , the dictionary \mathbf{D} , and the support of the non-zero coefficients (which was not considered in Appendix C as part of the coefficients model). Adding these components, an overall codelength for describing \mathbf{Y} in terms of a given model M , $L(\mathbf{Y}|M)$ is completely defined. In particular, the model chosen for the error term \mathbf{E} , together with the universal coding scheme developed for it, leads naturally to robust fitting terms that belong to the family of ψ -type M-estimators characterized by Huber [57].

3.3.3 MDL-based sparse coding and dictionary learning algorithms

The third component comprises the algorithms for performing sparse model selection. In this case, three main algorithms are provided:

MDL-based sparse coding: Given a data matrix \mathbf{Y} , this algorithm selects the best sparse code \mathbf{a}_j for each data sample \mathbf{y}_j for a fixed dictionary \mathbf{D} (thus, performing an efficient search in a subset of $\mathcal{M}(p)$). This search is performed sequentially on a sample-by-sample basis. The concatenation of such sparse codes, and the associated errors \mathbf{e}_j , constitute the model parameters (\mathbf{E}, \mathbf{A}) for the given dictionary \mathbf{D} . Here we introduce a collaborative variant in which statistical information from previously encoded samples, including spatial/temporal (Markov) dependencies, are used to improve

the coding efficiency of new samples. These are therefore MDL-based sparse coding algorithms. By virtue of the automatic model selection procedure involved, the sparsity level of each coefficients vector \mathbf{a}_j is chosen automatically according to the codelengths produced, and the resulting algorithm is free of any critical parameter to be tuned. Details on these algorithm are given in Appendix D, Section 3, and the corresponding Supplementary Material.

Fixed-size regularized dictionary learning: The second algorithm searches for the best dictionary $\hat{\mathbf{D}}$ within the sub-class $\mathcal{M}(p)$, thus constituting an MDL analog to the traditional dictionary learning algorithm described in Algorithm 1 on page 8. The fact that \mathbf{D} needs to be encoded as well to produce a complete description of the data results in a cost function (D.17) which includes a dictionary regularization term. This novel regularized dictionary learning formulation aids in the robustness of the dictionary learning process in cases where training samples are few and/or significantly corrupted by noise.

Parameter-free MDL-based dictionary learning: The third algorithm compares the best model in each subclass $\mathcal{M}(p)$, $\hat{M} = (\mathbf{A}(p), \mathbf{D}(p))$ in terms of the associated codelength cost for describing \mathbf{Y} in terms of it, $L(\mathbf{Y}|\mathbf{A}(p), \mathbf{D}(p))$, and chooses the one for which such codelength is the smallest. Together with the second algorithm, this one relieves the user of selecting the best dictionary size p , thus resulting in a parameter-free dictionary learning algorithm.

Combining the power of learned sparse models with the parameter-free model selection algorithms developed in this framework, we obtained results comparable to, and sometimes surpassing, the state-of-the-art in applications such as grayscale image denoising (figures 3.2, 3.3 and Table 3.1) and texture segmentation (Figure 3.4). In the case of image denoising, in addition to the traditional distortion-constrained (RD) denoising formulation

Table 3.1: Denoising results, in PSNR, compared for K-SVD [4], MDL denoising [58], and the Post-Thresholding (PT) and Rate-Distortion (RD) denoising variants. Notice the significant improvement over the previous MDL-based denoising algorithm [58].

noise → image ↓	$\sigma_e = 10$				$\sigma_e = 20$			
	PT	RD	[58]	[4]	PT	RD	[58]	[4]
lena	34.9	35.2	32.4	35.5	32.0	32.2	29.4	32.4
barbara	33.0	33.8	29.4	34.4	29.7	30.6	25.7	30.8
boat	33.1	33.2	30.5	33.6	29.5	30.3	27.5	30.3
peppers	34.1	34.4	32.2	34.3	31.7	31.6	29.4	30.8

found in other sparse model-based image denoising works such as [4, 49, 58], which depends on a hand tuned parameter (the required distortion ϵ), we propose a novel method called post-thresholding (PT), which is truly parameter free, is more consistent with the model assumptions, is faster, and produces less artifacts than the RD one (however, its performance in terms of PSNR is consistently worse than RD, which can be explained by a tendency to over-smoothing, see Figure 3.3).

3.4 Extension to other types of problems

As an example of the flexibility of the framework proposed in Appendix D for tackling the sparse model selection problem in a variety of scenarios, we introduce in appendixes D and E an extension of this framework for performing MDL-based model selection on the set of low-rank matrix approximations of a given data matrix \mathbf{Y} . Here we summarize the concepts and results from these works, leaving the details to the respective appendixes.

The low-rank matrix approximation family of problems (see [59] for a review) can be seen as an extension to the problem of sparse coding where sparsity is substituted by matrix rank. Concretely, the task is to recover a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ from an incomplete and/or corrupted observation \mathbf{Y} , under the assumption that the rank of \mathbf{A} , $\text{rank}(\mathbf{A})$, is small. As with sparse coding, $\text{rank}(\mathbf{A})$ is relaxed using the ℓ_1 equivalent for matrix rank, which is



Figure 3.2: Denoising results for $\sigma = 10$. For each image we show, from left to right: detail of clean image, noisy image, PT result, RD result.



Figure 3.3: Denoising results for $\sigma = 20$. For each image we show, from left to right: detail of clean image, noisy image, PT result, RD result. Note the artifacts produced by the RD algorithm. The PT algorithm, on the other hand, has a tendency to over-smooth.

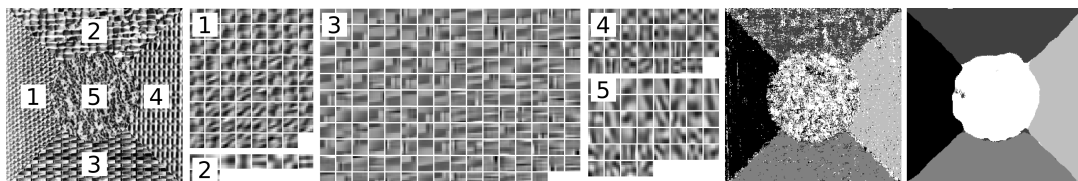


Figure 3.4: Left to right: Texture mosaic, dictionaries learned for each class (note the automatically learned different sizes), patch-wise codelength-based classification map –each shade of gray corresponds to a texture class – (77.0% success rate), classification map obtained by averaging the codelength over a neighborhood of patches (95.4% success rate).

the nuclear norm, $\|\mathbf{A}\|_* := \sum_i \sigma_i(\mathbf{A})$, where $\sigma_i(\mathbf{A})$ is the i -th singular value of \mathbf{A} . It has been shown in [59] that, under certain assumptions on $\text{rank}(\mathbf{A})$, the following estimation function is able to recover \mathbf{A} from a noisy observation \mathbf{Y} , and with a significant fraction of its coefficients arbitrarily corrupted,

$$\hat{\mathbf{A}} = \arg \min_{\mathbf{W}} \|\mathbf{W}\|_* + \lambda \|\mathbf{Y} - \mathbf{W}\|_1, \quad \lambda = 1/\sqrt{\max\{m, n\}}. \quad (3.8)$$

The setting chosen in our case is the robust background estimation problem in camera surveillance video sequences [60], which is a common proof of concept when applying (D.19). To perform our MDL-based model selection within this formulation, we solve (D.19) for increasing values of λ , obtaining a low-rank approximation to \mathbf{A} , $(\mathbf{A}(\lambda), \mathbf{E}(\lambda) = \mathbf{Y} - \mathbf{A}(\lambda))$ in each case. To solve the sequence of problems (D.19) efficiently, we modified the algorithm described in [61] to allow for warm restarts, using the solution for the previous λ as a starting point for the next λ , this resulting in a significant reduction of the overall computational cost of the model selection process.

In order to encode the low-rank approximation $(\mathbf{A}(\lambda), \mathbf{E}(\lambda))$ efficiently, we exploit the potential sparsity of $\mathbf{E}(\lambda)$ and the low-rank of $\mathbf{A}(\lambda)$. The first is done as in the case of the learned sparse models described above by first describing the support of the non-zero errors using an universal enumerative code [62], and then its non-zero values using the universal models for Laplacian variables `MOE` developed in Appendix C. In order to exploit the low-rank of $\mathbf{A}(\lambda)$, we use its reduced SVD decomposition $\mathbf{A}(\lambda) = \mathbf{U}(\lambda)\Sigma(\lambda)\mathbf{V}(\lambda)^T$. For $\text{rank}(\mathbf{A}(\lambda)) = r$, we have that $\mathbf{U}(\lambda) \in \mathbb{R}^{m \times r}$ are the left-eigenvectors, $\Sigma \in \mathbb{R}^{r \times r}$ is the diagonal matrix whose diagonal are the non-zero singular values of $\mathbf{A}(\lambda)$, and $\mathbf{V}(\lambda) \in \mathbb{R}^{r \times n}$ are the right-eigenvectors of $\mathbf{A}(\lambda)$. The overall low-rank decomposition is depicted in Figure 3.5. Each component in this decomposition is assigned a codelength, so that the total description codelength of \mathbf{Y} under a given model $(\mathbf{A}(\lambda), \mathbf{E}(\lambda))$ is given by $L(\mathbf{Y}; \lambda) =$

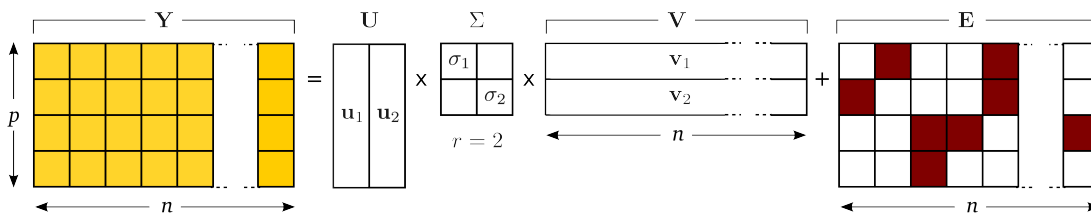
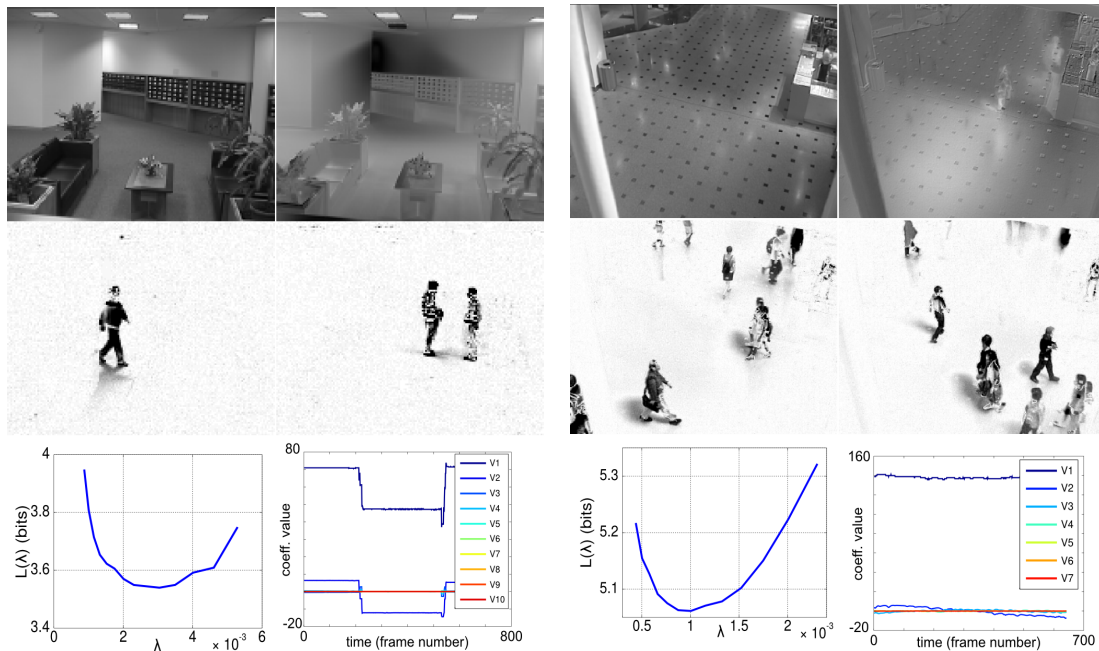


Figure 3.5: Scheme of low-rank decomposition of the data $\mathbf{Y} = \mathbf{A} + \mathbf{E}$, showing the assumed sparsity of \mathbf{E} and the reduced SVD decomposition of $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}$ (of rank $r = 2$ in the example).

$L(\mathbf{E}(\lambda)) + L(\mathbf{U}(\lambda)) + L(\Sigma(\lambda)) + L(\mathbf{V}(\lambda))$. The MDL-based estimation algorithm then chooses the model for which the codelength $L(\mathbf{Y}; \lambda) = L(\mathbf{E}(\lambda)) + L(\mathbf{U}(\lambda)) + L(\Sigma(\lambda)) + L(\mathbf{V}(\lambda))$ is minimized. The details on how these codelengths are computed, including the probability models used, and choices such as the quantization of the different components, are given in Appendix E. As in [60], here we show results for two sequences taken from [63]: “Lobby” (Figure 3.6(a)), and “ShoppingMall” (Figure 3.6(b)). Full videos of these and other video sequences can be viewed at <http://www.tc.umn.edu/~nacho/lowrank/>.



(a) Results for “Lobby” sequence, featuring a room with lights that are switched off and on. The rank of the approximation for this case is rank = 10. The moment where the lights are turned off is clearly seen here as the “square pulse” in the middle of the first two right-eigenvectors (bottom-right). Also note how \mathbf{u}_2 (top-right) compensates for changes in shadows.

(b) Results for “ShoppingMall”, a fixed camera looking at a crowded hall. In this case, the rank of the approximation decomposition is rank = 7. Here, the first left-eigenvector models the background, whereas the rest tend to capture people that stood still for a while. Here we see the “phantom” of two such persons in the second left-eigenvector (top-right).

Figure 3.6: Low-rank approximation results. Both figures show the first two left-eigenvectors as 2D images at the top, two sample frames from the approximation error sequences in the middle, which should contain the people that were removed from the videos, and the curve $L(\lambda)$ and the right-eigenvalues, scaled by Σ (representing the “activity” of each left-eigenvector along time), at the bottom.

4 Conclusions

In the present thesis, we have provided new theoretical and practical building blocks towards defining a new generation of learned sparse models. These include:

New sparse coding formulations incorporating further structure, beyond sparsity, which is found in important sparse modeling applications such as source separation and identification, obtaining state-of-the-art results in those applications.

Novel dictionary learning formulations promoting incoherence and block-incoherence in the learned dictionaries for improved sparse recovery accuracy and reduced computational cost during coding. The benefits of imposing these features on dictionaries were demonstrated in tasks such as image denoising and classification.

A new family of universal sparsity-promoting regularizers which are flexible, robust, and replace critical hand-tuned parameters in favor of non-critical, non-informative hyperparameters. The information-theoretic framework on which these regularizers are based further sheds new light into ad-hoc regularizers that have been proposed in the literature, and defines a general methodology for defining new regularizers for other types of data.

A parameter-free sparse coding and dictionary learning framework that automatically adapts to the inherent complexity of the modeled data. This framework provides a new MDL-based information-theoretic approach to understanding sparse models as data modeling tools, highlighting the role of sparsity, and overcompleteness, in the

flexibility and robustness of such models for adapting to given data. The results obtained match, and some times surpass, the state-of-the-art in the applications on which the present framework was deployed. In particular, this is the first MDL-based framework that yields results that are competitive with the state-of-the-art in image denoising. The framework is also efficient, in the sense that the computational requirements of both the sparse coding and dictionary learning algorithms developed is similar to those already available for traditional sparse models.

A parameter-free low-rank matrix approximation algorithm derived using the same ideas as the sparse modeling framework. We applied this framework to the problem of complex background extraction in video sequences, obtaining state-of-the-art results out of the box.

Future work

There are several ways in which the above work can be extended and/or improved. Some possible lines of work include:

Generalizing the concept of sparsity, for example, beyond indicating solutions with “many zeroes”. Instead, one could think of signals which alternate between a predominant “background model” (for example, background noise), and a “large signal model”, for indicating relevant samples. The MDL-based framework presented in Appendix D can easily accommodate this idea, making this generalization an appealing line of research in the immediate future.

Automatic scale selection in image processing applications. Our framework in Appendix D could be extended to learn the patch width w automatically, a parameter whose choice is often critical for the success of any patch-based application. Currently available multi-scale analysis tools could provide us with efficient ways to perform this

search.

MDL-based structured sparse modeling. This would combine the two main lines of work of this thesis, by complementing the structured sparse models developed in Appendix B, including group, hierarchical and collaborative hierarchical models, with the MDL-based model selection framework developed in Appendix D.

References

- [1] B. Olshausen and D. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.
- [2] K. Engan, S. Aase, and J. Husoy. Multi-frame compression: Theory and design. *Signal Processing*, 80(10):2121–2140, Oct. 2000.
- [3] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. Lee, and T. J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Computation*, 15(2):349–396, 2003.
- [4] M. Aharon, M. Elad, and A. Bruckstein. The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Trans. SP*, 54(11):4311–4322, Nov. 2006.
- [5] D. Donoho and M. Elad. Optimal sparse representation in general (nonorthogonal) dictionaries via l_1 minimization. In *Proc. of the National Academy of Sciences*, volume 100, pages 2197–2202, Mar. 2003.
- [6] J. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. IT*, 50(10):2231–2242, Oct. 2004.
- [7] E. J. Candès. Compressive sampling. *Proc. of the International Congress of Mathematicians*, 3, Aug. 2006.

- [8] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.
- [9] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. on Pure and Applied Mathematics*, 57:1413–1457, 2004.
- [10] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.
- [11] J. Friedman, H. Hastie, and R. Tibshirani. Regularized paths for generalized linear models via coordinate descent. *Journal of Stat. Soft.*, 33(1), 2008.
- [12] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.
- [13] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Trans. PAMI*, 27(6):957–968, 2005.
- [14] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Adv. NIPS*, volume 21, 2009.
- [15] J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. *SIAM MMS*, 7(1):214–241, Apr. 2008.
- [16] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML*, pages 759–766, June 2007.
- [17] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, Feb. 2009.

- [18] R. Rubinstein, A. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, June 2010.
- [19] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proc. IEEE*, 98(6):1031–1044, June 2010.
- [20] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin. Nonparametric Bayesian dictionary learning for analysis of noisy and incomplete images. To appear in *IEEE Trans. IP*, 2011.
- [21] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Trans. IT*, 30(4):629–636, 1984.
- [22] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
- [23] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Trans. IT*, 44(6):2743–2760, 1998.
- [24] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.
- [25] G. Peyré. Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, 34:17–31, May 2009.
- [26] A. Y. Yang, J. Wright, Y. Ma, and S. Sastry. Feature selection in face recognition: A sparse representation perspective. *IEEE Trans. PAMI*, 2007. submitted.
- [27] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.
- [28] G. Schwartz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

- [29] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Trans. SP*, 41(12):3397–3415, 1993.
- [30] D. Donoho. De-noising by soft-thresholding. *IEEE Trans. IT*, 41(3):613–627, 1995.
- [31] R. Giryes, M. Elad, and Y.C. Eldar. The projected GSURE for automatic parameter tuning in iterative shrinkage methods. *Applied and Computational Harmonic Analysis*, 30(3):407–422, 2011.
- [32] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.
- [33] J Portilla, V Strela, M J Wainwright, and E P Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. IP*, 12(11):1338–1351, Nov. 2003.
- [34] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.
- [35] I. Ramirez, P. Sprechmann, and G. Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3501 –3508, june 2010.
- [36] P. Sprechmann, I. Ramirez, G. Sapiro, and Y.C. Eldar. C-hilasso: A collaborative hierarchical sparse modeling framework. *Signal Processing, IEEE Transactions on*, 59(9):4183 –4198, Sept. 2011.
- [37] I. Ramírez and G. Sapiro. Universal regularizers for robust sparse coding and modeling. Submitted. Preprint available in <http://arxiv.org/abs/1003.2941> [cs.IT], Aug. 2010.

- [38] I. Ramírez and G. Sapiro. An MDL framework for sparse coding and dictionary learning. Submitted to IEEE Trans. SP. Preprint available at <http://arxiv.org/abs/1110.2436>.
- [39] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Roy. Statist. Soc. Ser. B*, 68:49–67, 2006.
- [40] R. Jenatton, J. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. Technical Report arXiv:0904.3523v1, INRIA, 2009.
- [41] J. Tropp. Algorithms for simultaneous sparse approximation. part ii:convex relaxation. *Signal Processing*, 86(3):589–602, 2006.
- [42] J. A. Tropp, A. C. Gilbert, and M. J. Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing, special issue "Sparse approximations in signal and image processing"*, 86:572–588, Apr. 2006.
- [43] Y. C. Eldar and M. Mishali. Robust recovery of signals from a structured union of subspaces. *IEEE Trans. Inform. Theory*, 55(11):5302–5316, Nov. 2009.
- [44] Y. C. Eldar and H. Rauhut. Average case analysis of multichannel sparse recovery using convex relaxation. *IEEE Trans. IT*, 56(1):505–519, 2010.
- [45] M. Mishali and Y. C. Eldar. Reduce and boost: Recovering arbitrary sets of jointly sparse vectors. *IEEE Trans. SP*, 56(10):4692–4702, Oct. 2008.
- [46] B. A. Turlach, W. N. Venables, and S. J. Wright. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.
- [47] M. Elad, B. Matalon, J. Shtok, , and M. Zibulevsky. A wide-angle view at iterated shrinkage algorithms. In *SPIE (Wavelet XII)*, pages 26–29, San-Diego CA, Aug. 2007.

- [48] I. Ramírez and G. Sapiro. Low-rank data modeling via the Minimum Description Length principle. Submitted to ICASSP 2012. Preprint available at <http://arxiv.org/abs/1109.6297>.
- [49] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Trans. IP*, 17(1):53–69, Jan. 2008.
- [50] T. Cover and J. Thomas. *Elements of information theory*. John Wiley and Sons, Inc., 2 edition, 2006.
- [51] M. Weinberger, G. Seroussi, and G. Sapiro. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Trans. IP*, 9, 2000.
- [52] P. Moulin and J. Liu. Analysis of multiresolution image denoising schemes using generalized-Gaussian and complexity priors. *IEEE Trans. IT*, Apr. 1999.
- [53] E. J. Candès, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *J. Fourier Anal. Appl.*, 14(5):877–905, Dec. 2008.
- [54] H. Zou and R. Li. One-step sparse estimates in nonconcave penalized likelihood models. *Annals of Statistics*, 36(4):1509–1533, 2008. DOI: 10.1214/009053607000000802.
- [55] E. Lam and J. Goodman. A mathematical analysis of the DCT coefficient distributions for images. *IEEE Trans. IP*, 9(10):1661–1666, 2000.
- [56] J. Trzasko and A. Manduca. Relaxed conditions for sparse signal recovery with general concave priors. *IEEE Trans. SP*, 57(11):4347–4354, 2009.
- [57] P. J. Huber. Robust estimation of a location parameter. *Annals of Statistics*, 53:73–101, 1964.

- [58] T. Roos, P. Myllymäki, and J. Rissanen. MDL denoising revisited. *IEEE Trans. SP*, 57(9):3347–3360, 2009.
- [59] E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3), May 2011.
- [60] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. In *Adv. NIPS*, Dec. 2009.
- [61] Z. Lin, M. Chen, and Y. Ma. The Augmented Lagrange Multiplier Method for exact recovery of corrupted low-rank matrices. <http://arxiv.org/abs/1009.5055>.
- [62] T. M. Cover. Enumerative source encoding. *IEEE Trans. IT*, 19(1):73–77, 1973.
- [63] L. Li, W. Huang, I. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Trans. IP*, 13(11):1459–1472, 2004.

A

Classification and Clustering via Dictionary Learning with Structured Incoherence and Shared Features

Ignacio Ramirez, Pablo Sprechmann, and Guillermo Sapiro

Electrical and Computer Engineering, University of Minnesota

A clustering framework within the sparse modeling and dictionary learning setting is introduced in this work. Instead of searching for the set of centroid that best fit the data, as in k-means type of approaches that model the data as distributions around discrete points, we optimize for a set of dictionaries, one for each cluster, for which the signals are best reconstructed in a sparse coding manner. Thereby, we are modeling the data as a union of learned low dimensional subspaces, and data points associated to subspaces spanned by just a few atoms of the same learned dictionary are clustered together. An incoherence promoting term encourages dictionaries associated to different classes to be as independent as possible, while still allowing for different classes to share features. This term directly acts on the dictionaries, thereby being applicable both in the supervised and unsupervised settings. Using learned dictionaries for classification and clustering makes this method robust and well suited to handle large datasets. The proposed framework uses a novel measurement for the quality of the sparse representation, inspired by the robustness of the ℓ_1 regularization term in sparse coding. In the case of unsupervised classification and/or clustering, a new initialization based on combining sparse coding with spectral clustering is proposed. This initialization clusters the dictionary atoms, and therefore is based on solving a low dimensional eigen-decomposition problem, being applicable to large datasets. We first illustrate the proposed framework with examples on standard image and speech datasets in the supervised classification setting, obtaining results comparable to the state-of-the-art with this simple approach. We then present experiments for fully unsupervised clustering on extended standard datasets and texture images, obtaining excellent

performance.

1 Introduction and Basic Formulation

In recent years, sparse representations have received a lot of attention from the signal processing community. This is due in part to the fact that an important variety of signals such as audio and natural images can be well approximated by a linear combination of a few elements (atoms) of some (often) redundant basis, usually called dictionaries [1].

Sparse modeling aims at learning these non parametric dictionaries from the data itself. Several algorithms have been developed for this task, e.g., the κ -SVD and the method of optimal directions (MOD). Recent publications in a wide spectrum of signals and applications have shown that this approach can be very successful, leading to state-of-the art results, e.g., in image restoration and denoising, texture synthesis, and texture classification. In the supervised or weakly supervised classification setting, this class of algorithms learn dictionaries from the labeled training dataset and use features of the sparse decomposition of the testing signal for classification (see [2, 3, 4, 5]).

In this paper we propose a framework for clustering datasets that are well represented in the sparse modeling framework with a set of learned dictionaries (see [6] for our earlier work in this direction). Given K clusters, we learn K dictionaries for representing the data, and then associate each signal to the dictionary for which the “best” sparse decomposition is obtained. Note that it is not that each data point belongs to a union of subspaces as for example in [7, 8]. Comparing with block/group sparsity, here a single dictionary (block) is selected per data point, and the point is sparsely represented (subspace) with atoms only from this dictionary.¹ Also in contrast with more classical subspace clustering, data points in the same class can belong to more than one subspace, since each dictionary represents a

¹Sharing atoms between the classes, and therefore having non-empty intersecting subspaces is permitted in our framework as well, see Section 3.1

large number of subspaces (each sparsity pattern defines one subspace). The model is then very rich and non-linear.

The first building block of the proposed clustering framework is based on considering

$$\min_{\mathbf{D}_i, C_i} \sum_{i=1}^K \sum_{\mathbf{x}_j \in C_i} \mathcal{R}(\mathbf{x}_j, \mathbf{D}_i), \quad (\text{A.1})$$

where $\mathbf{D}_i = [\mathbf{d}_1 | \mathbf{d}_2 \dots | \mathbf{d}_{k_i}] \in \mathbb{R}^{n \times k_i}$ is a dictionary of k_i atoms associated with the class C_i , $\mathbf{x}_j \in \mathbb{R}^n$ are the data vectors, and \mathcal{R} is a function that measures how good the sparse decomposition for the signal \mathbf{x}_j under the dictionary \mathbf{D}_i is. In the general case, different dictionaries may have different number of atoms, k_i might be cluster dependent. This problem is closely related with the k - q -flat algorithm that aims at finding the closest k q -dimensional flats to a dataset [9]. However, there are major differences between the two. In particular, the framework here proposed, following the sparse representation approach, considers a large number of flats per class, and does not assume a pre-defined, or even constant across classes, (q) dimension, resulting in a richer space for representing and clustering the signals.

To complete the model, we add a block/dictionary incoherence term, inspired in part by the works on standard sparse coding, e.g., [10, 11, 7, 12], where it was shown that both the speed and accuracy of sparse coding techniques such as soft-thresholding and orthogonal matching pursuit depend on the incoherence between the dictionary atoms. Here we add a term $\mathcal{Q}(\mathbf{D}_i, \mathbf{D}_j)$ that promotes incoherence between the different dictionaries, thereby obtaining a general energy of the form

$$\min_{\mathbf{D}_i, C_i} \sum_{i=1}^K \sum_{\mathbf{x}_j \in C_i} \mathcal{R}(\mathbf{x}_j, \mathbf{D}_i) + \eta \sum_{i \neq j} \mathcal{Q}(\mathbf{D}_i, \mathbf{D}_j). \quad (\text{A.2})$$

This energy will then lead to the learning of dictionaries optimized to properly represent

the corresponding class, due to \mathcal{R} , while at the same time being weak for the other classes, due to the term \mathcal{Q} . We will later show how classes can still share atoms, an important property for classification algorithms [13], and a unique characteristic of our proposed model. Note that in contrast with prior work on dictionary learning for classification, this novel cross dictionary learning term \mathcal{Q} is independent of the data, is intrinsic to the dictionaries being learned, thereby rendering itself also to the case of unsupervised or semi-supervised classification and clustering. For the experiments in this paper we use the terms $\mathcal{Q}(\mathbf{D}_i, \mathbf{D}_j) = \|\mathbf{D}_i^T \mathbf{D}_j\|_F^2$, where the subscript F denotes Frobenius norm.²

We propose a measurement \mathcal{R} for the quality of the sparse representation that naturally takes into account both the reconstruction error and the sparseness (complexity) of the representation on the corresponding learned dictionary. Such measurement can be applied to image patches directly or to image features, e.g., SIFT as in [5]. In practice this measurement has shown enormous discrimination power. To further show this, we performed experiments in the supervised classification setting using labeled data; we first learned a dictionary for each class (with the incoherence promoting term \mathcal{Q}), and then classified each testing signal according to this measure. This very simple approach gives results comparable with the state-of-the-art for several benchmark datasets. Thereby, as a by-product of our proposed clustering framework, we obtain a very simple and efficient supervised classification technique as well.

In the unsupervised clustering case, the initialization is very important for the success of the algorithm. Due to the cost associated with the procedure, repeating random initializations is practically impossible. Thus a “smart” initialization is needed. We propose an approach that combines sparse coding with spectral clustering [15], and is applicable to large datasets.

²We can also easily add internal incoherence between the atoms of each dictionary [14], in order to further stabilize not only the dictionary selection but the particular atoms in the corresponding dictionary. This is done here for the initialization step.

Ideas related to the ones here proposed were previously employed for subspace clustering [16, 17, 18], clustering using the so-called ℓ_1 -graph by Huang and Yan (see description in [19]), and label propagation [20]. In contrast with our proposed dictionary learning framework, these works model all the data points in a given class as belonging to the same unique subspace, while we model them as “belonging” to the same dictionary, a richer non-linear model since each subset of atoms from the dictionary represents a different subspace. Moreover, these very inspiring approaches all use the data itself as dictionary, sparsely representing every data point as a linear combination of the rest of the data. Such representation is computationally expensive (virtually unusable for datasets of thousands of points). In addition, the large redundancy and coherence expected from using the data itself as dictionary is prompt to make the sparse coding very unstable: as mentioned above, it is well known that such coding techniques strongly depend on the internal coherence of the dictionary. Furthermore, the performance of these methods decreases when the number of clusters grows. We propose as part of our framework a method to bypass this problem that divides the clustering problem into several binary ones. In a natural way, we use the proposed energy function to decide which partition to choose. Such binary division framework is not so natural for these other related clustering methods.

In Section 2 we summarize the main ideas of sparse coding and dictionary learning. In Section 3 we define the measure \mathcal{R} and analyze its discriminative power providing examples of supervised classification. In Section 4 we present the proposed clustering algorithm, together with theoretical guarantees and experimental results. Finally, we conclude the paper in Section 5.

2 Sparse Coding and Dictionary Learning

Sparse coding means to represent a signal as a linear combination of a few atoms of a given dictionary. Mathematically, given a signal $\mathbf{x} \in \mathbb{R}^n$ and a dictionary $\mathbf{D} \in \mathbb{R}^{n \times k}$, the sparse representation problem can be stated as $\min_{\mathbf{a}} \|\mathbf{a}\|_0$, s.t. $\mathbf{x} = \mathbf{D}\mathbf{a}$, where $\|\mathbf{a}\|_0$ is the ℓ_0 pseudo-norm of the coefficient vector $\mathbf{a} \in \mathbb{R}^k$, the number of non-zero elements. As minimizing ℓ_0 is NP-hard, a common approximation is to replace it with the ℓ_1 -norm. In the noisy case the equality constraint must be relaxed as well. An alternative then is to solve the unconstrained problem,

$$\min_{\mathbf{a}} \|\mathbf{x} - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_1, \quad (\text{A.3})$$

where λ is a parameter that balances the tradeoff between reconstruction error and sparsity. It is a well known fact that the ℓ_1 constraint induces sparse solutions for the coefficient vectors \mathbf{a} . Furthermore, this is a convex problem that can be solved very efficiently using for example the LARS-Lasso algorithm [21]. This alternative has also been shown to be more stable than the ℓ_0 approach in the sense that in the latter, small variations in the input signal can produce very different active sets (the set of non-zero coefficients in \mathbf{a} , or selected atoms from \mathbf{D}).

Now, what about the actual dictionary \mathbf{D} ? State-of-the-art results have shown that it should in general be learned from data. Given a set of signals $\{\mathbf{x}_i\}_{i=1 \dots m}$ in \mathbb{R}^n , the goal is to find a dictionary $\mathbf{D} \in \mathbb{R}^{n \times k}$ such that each signal in the set can be represented as a sparse linear combination of its atoms. In this work we use a variation of [22], where learning the dictionary is done by seeking a (local) solution to the following optimization problem,

$$\min_{\mathbf{D}, \{\mathbf{a}_i\}_{i=1, \dots, m}} \sum_{i=1}^m \|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2 + \lambda \|\mathbf{a}_i\|_1, \quad (\text{A.4})$$

while restricting the atoms to have norm less than one. The optimization is carried out using an iterative approach that is composed of two (convex) steps: the sparse coding step on a fixed \mathbf{D} and the dictionary update step on fixed \mathbf{a} .

3 The Sparse Representation Quality $\hat{\mathcal{R}}$ and Supervised Classification

A common approach when using dictionaries for classification is to train class specific dictionaries using labeled data and then assign each testing signal to the class for which the best reconstruction is obtained [3, 4]. The measure employed for this task is often the reconstruction error, $\mathcal{R}(\mathbf{x}, \mathbf{D}) = \|\mathbf{x} - \mathbf{D}\mathbf{a}\|_2^2$, where \mathbf{a} is the optimal coefficient vector in the sparse coding. While this strategy leads to very good results, it does not take into account the actual sparsity of the reconstruction. Suppose that we have two dictionaries for which almost the same reconstruction error is obtained, but one of them requires double the atoms than the other. In such a situation one would rather select the dictionary that gives the sparsest solution (simplest, following Akaike's Information Principle [23]), even if the reconstruction error is slightly larger.

In practice, this problem can be addressed using a small pre-defined sparsity level L in an ℓ_0 approach. This strategy is not longer valid when the convex relaxation (A.3) is employed (such relaxation is critical for classification tasks requiring robustness and stability). In this situation, comparing the reconstruction errors alone has little meaning. We propose then to use the actual cost function in (A.3) as a measure of performance, as in the dictionary learning (A.4), $\hat{\mathcal{R}}(\mathbf{x}, \mathbf{D}) = \min_{\mathbf{a}} \|\mathbf{x} - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_1$. This alternative takes into account both the reconstruction error and the complexity of the sparse decomposition. The reconstruction error measures the quality of the approximation while the complexity is measured by the ℓ_1 norm of the optimal \mathbf{a} .

dataset	proposed	data	A	B	C	SVM	k-NN
MNIST	1.26	1.35	3.41	1.05	-	1.4	5.0
USPS	3.98	4.14	3.56	4.38	6.05	4.2	5.2
ISOLET	3.01	3.34	4.3	3.4	-	3.3	8.7

Table A.1: Error rate (in percentage) for the algorithm discussed in Section 3. We present comparisons with recently published approaches (results taken from the corresponding papers). The “data” column corresponds to using our discrimination function with dictionaries formed with the whole training dataset. MNIST: (A) is the best reconstructive method presented in [24], while (B) is the best discriminative one. USPS: (A) is the best reconstructive and (B) is the best discriminative method, both reported in [24]. (C) is the best result obtained in [25] (only USPS available). ISOLET: (A) is the supervised k - q -flats and (B) is the k -metrics in [26]. We also compare with an SVM with Gaussian kernel and the Euclidean k-NN.

Let \mathbf{X}_i , $i = 1, \dots, K$, be a collection of K (labeled) classes of signals and \mathbf{D}_i the corresponding dictionaries trained for each of them independently following for example (A.4). This gives, for each class, a (reconstructive) dictionary unaware of the task (classification/clustering) and of the data in the other classes. Thereby, as detailed in the introduction, it is more appropriate to add the dictionary incoherence $\mathcal{Q}(\mathbf{D}_i, \mathbf{D}_j)$, and the proposed optimization is

$$\min_{\{\mathbf{D}_i, \mathbf{A}_i\}_{i=1..K}} \sum_{i=1}^K \left\{ \|\mathbf{X}_i - \mathbf{D}_i \mathbf{A}_i\|_2^2 + \lambda \sum_{j=1}^{m_i} \|\mathbf{a}_i^j\|_1 \right\} + \eta \sum_{i \neq j} \|\mathbf{D}_i^T \mathbf{D}_j\|_F^2. \quad (\text{A.5})$$

Here we used the standard notation $\mathbf{A}_i = [\mathbf{a}_i^1 \dots \mathbf{a}_i^{m_i}] \in \mathbb{R}^{k_i \times m_i}$, each column \mathbf{a}_i^j is the sparse code corresponding to the signal $j \in [1..m_i]$ in class i . Note that the first term in the optimization is as in (A.4), where each dictionary is optimized for the data from its own class. The second term provides the coupling. In contrast with works such as [3], the coupling is between the dictionaries, the labeled data points do not form part of this term, thereby this can be used also in the non-supervised learning process, see next section.

Once the dictionaries have been learned, the class \hat{j}_0 for a given new signal \mathbf{x} is found by solving $\hat{j}_0 = \arg \min_{j=1, \dots, K} \hat{\mathcal{R}}(\mathbf{x}, \mathbf{D}_j)$.³ This procedure is very simple and its few parameters can be found via cross-validation.

3.1 Sharing Atoms

In practice, it turns out that even though we impose incoherence in the dictionaries, atoms representing common features in all classes tend to appear repeated almost exactly in dictionaries corresponding to different classes. Being so common, these atoms are used often and their associated reconstruction coefficients have a high absolute value $|\mathbf{a}_r|, r \in \{1, \dots, k_i\}$, thus making the reconstruction costs $\hat{\mathcal{R}}(\mathbf{x}, \mathbf{D}_i)$ similar. By ignoring the coefficients associated to these common atoms when computing $\hat{\mathcal{R}}$, we can improve the discriminatory power of the system. The natural way to detect such atoms is to inspect the already available $\mathbf{D}_i^T \mathbf{D}_j$ matrices, whose absolute values represent the inner products between atoms. In the following experiments, a threshold of 0.95 consistently improves the results, sometimes significantly. Note that this procedure accounts for allowing classes to share features [13], and the corresponding subspaces to have intersections, in contrast for example with [16]. Figure A.1 illustrates examples of automatically learned shared atoms in the task of learning to classify digits from the MNIST dataset. See [27] for the selection of features (atoms) for parametric dictionaries.

3.2 Experimental Results

We first test this simple classification method with standard datasets, the MNIST and USPS digit datasets and the ISOLET data that consists of 617 audio features extracted from 200 speakers saying each letter of the alphabet twice. We used in every case the usual training/testing split. In Table A.1 we present the obtained results. We compare our results with

³We actually obtain more than this information, since for each \mathbf{x} we compute all the $\hat{\mathcal{R}}$ s for all the K classes, and thereby can provide a soft classification with probabilities, or a feature vector for an SVM.

several much more sophisticated classification algorithms. The results obtained are comparable and sometimes even better. We also compare with the standard Euclidean k-NN and with SVM with a Gaussian kernel. In all our experiments we used a penalty parameter $\lambda = 0.1$. The size of the dictionary depends on the number of training samples as well as the intrinsic complexity of the data. For MNIST, which has many samples, our best results were obtained with $k = 800$. In contrast, USPS and ISOLET have much less samples and more variability, leading to a much smaller dictionaries of size $k = 80$ and $k = 60$ respectively. These already state-of-the-art results can be further improved for example using the $\hat{\mathcal{R}}_i$ in an SVM.

One could think of using the whole training datasets as dictionaries for each class as with the approaches mentioned in the introduction [16, 18, 19]. In that case, in all our experiments the error rates obtained are not better than the ones reported in Table A.1. Using the data as dictionaries has the additional disadvantage that the computational cost of the classification becomes prohibitive,⁴ and the method is highly susceptible to label errors due to the high coherence of the “dictionary.”

Finally, we illustrate the discrimination power of the measure $\hat{\mathcal{R}}$ in a more challenging scenario using images from the Graz02 dataset [28]. We address the object detection task by learning dictionaries for the local SIFT descriptors of an object class.

We chose the “bike” class from the Graz02 as an example, and test our proposed framework in two different weakly supervised settings. In the first setting, along with the training images, we provide the algorithm with a bounding box enclosing the bikes present in each of these images. In the second, the only supplied information is whether a bike is present or not in each of the training images. Clearly, the second case is more challenging. On each image we extract 128 dimensional SIFT descriptors from patches of 32×32 pixels computed over a grid with spacing of 4 pixels. For the first setting, we randomly pick

⁴The cost of learning the dictionaries in our approach is off-line.

300,000 SIFT descriptors from inside and outside of the bounding box respectively, and learn corresponding dictionaries with 500 atoms each. In the second setting, the bike and background dictionaries were learned from all the patches extracted from images marked as either containing a bike or purely background respectively.

In both cases, the dictionary for the class “bike” was learned iteratively, keeping the 90% of the descriptors that were more clearly assigned to the class “bike” at each iteration. This allows us to gradually discard background descriptors labeled as “bike.” The choice of training and testing images was performed as it is usual for this dataset, where the first 300 images are split in two, the odd images for training, and the even images for testing.

Since classified patches overlap, each pixel in the image has several possible energy values $\hat{\mathcal{R}}$ for each of the two dictionaries, one per patch covering it. This spatial redundancy helps the algorithm to determine a more accurate energy value at the pixel level by means of a simple spatial average, with a Gaussian kernel, giving more weight to patches in which the pixel is closer to the center. Using a Gaussian regularization on the energy images has proven to improve the results. This is in part due to the way the ground truth masks are defined, Figure A.2. The wheels are labeled as belonging to the class “bike” while most of the time one can see the background behind them. A strategy that considers the features globally or at several scales would help [29, 5], but this is beyond the scope of this example.

In Figure A.2 we show the detection results obtained with this framework. We also show the corresponding precision vs. recall curve for the whole testing set. The results are very good, comparable to state-of-the-art, considering that we are using one single dictionary to categorize each of these highly complex categories. In this object localization application, the cancelation of atoms of high coherence has a crucial role because of the similarities that both classes have at the local level. If one uses directly dictionaries trained independently for each class, then most of the diagonal vertexes on the image tend to be classified as “bikes” and the opposite happens with horizontal and vertical edges, which are



Figure A.1: Atoms discarded due to excessive coherence. From left to right: 1 vs. 9, 3 vs. 5, 4 vs. 7, 5 vs. 8, 8 vs. 9. Notice how these atoms have learned features shared between different classes.

very frequent in urban environments.

4 Dictionary Learning for Clustering

We now proceed to extend the above dictionary learning and sparse coding frameworks to unsupervised clustering. Given a set of signals, $\{\mathbf{x}_j\}_{j=1\dots m}$ in \mathbb{R}^n , and the number of clusters/classes, K ,⁵ we want to find the set of K dictionaries $\mathbf{D}_i \in \mathbb{R}^{n \times k_i}$, $i = 1, \dots, K$, that best represents the data. We formulate this as an energy minimization problem of the form of Equation (A.2), and use the measure proposed in Section 3,

$$\min_{\mathbf{D}_i, C_i} \sum_{i=1}^K \sum_{\mathbf{x}_j \in C_i} \min_{\mathbf{a}_i^j} \|\mathbf{x}_j - \mathbf{D}_i \mathbf{a}_i^j\|_2^2 + \lambda \|\mathbf{a}_i^j\|_1 + \eta \sum_{i \neq j} \|\mathbf{D}_i^T \mathbf{D}_j\|_F^2, \quad (\text{A.6})$$

where as before, the atoms of all the dictionaries are restricted to have unit norm. In contrast with (A.5), class assignments are unknown, and the optimization is carried out iteratively using a Lloyd's-type algorithm: *Assignment step*: The dictionaries are fixed and each signal is assigned to the cluster for which the best representation is obtained: $C_{j_0} := \{\mathbf{x} : \hat{\mathcal{R}}(\mathbf{x}, \mathbf{D}_{j_0}) \leq \hat{\mathcal{R}}(\mathbf{x}, \mathbf{D}_i) \forall i = 1, \dots, K\}$ (omitting the contribution of shared atoms). *Update step*: The new dictionaries are computed fixing the assignments found in the previous step. This

⁵When K is over-estimated, a micro-detailed partition is observed.

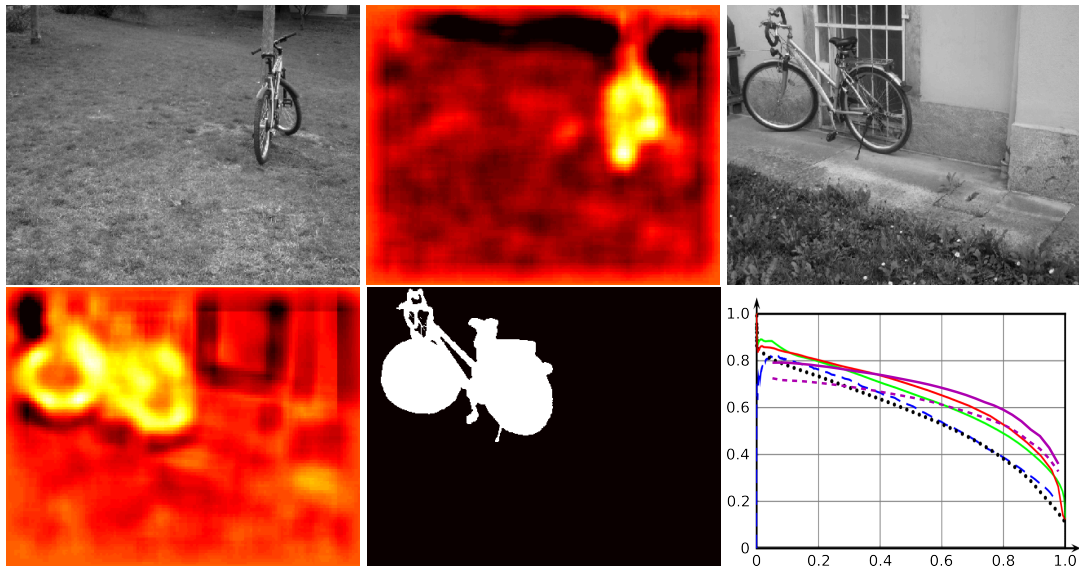


Figure A.2: Bike detection on the Graz dataset using the measure $\hat{\mathcal{R}}$. The two topmost rows show the obtained detection for two sample images. The colored area corresponds to the regions for which the representation energy using the bike dictionary is smaller than the background one. The lighter the color, the more “bike-like” is the pixel. Bottom left: shows the ground truth for the middle row. Bottom right: precision vs. recall curve for several algorithms [30] (blue,dashed), [31] (black,dotted), [3] (red and green), and the proposed algorithm, using a bounding box (magenta,solid), and weakly supervised (magenta,dashed).

is the dictionary learning problem (A.4), with the addition of the incoherence term.

The algorithm stops when the relative change in the energy is less than a given constant. In practice few iterations are needed to reach good results. While the energy is being reduced at every step, there is no guarantee of arriving to a global minimum. In this setting, repeated initializations are computationally very expensive, thus a good initialization is required. This is explained next.

4.1 Initialization: Spectral Clustering Meets Dictionary Learning

The initialization for the algorithm presented in the previous section can be given as a set of K dictionaries or as an initial partition of the data, this is the C_i sets. We propose two closely related algorithms one corresponding to each of these two alternatives. In both

cases the main idea is to construct a similarity matrix and use it as the input for a spectral clustering algorithm [32].

Let $\mathbf{D}_0 \in \mathbb{R}^{n \times k_0}$ be an initial global dictionary trained (with internal incoherence) to reconstruct the data for the whole (unlabeled) set $X := [\mathbf{x}_1, \dots, \mathbf{x}_m]$. For each signal \mathbf{x}_j we have the corresponding sparse representation \mathbf{a}_j . Let us define $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_m] \in \mathbb{R}^{k_0 \times m}$. Two signals belonging to the same cluster are expected to have decompositions that use similar atoms. Thus one can measure the similarity of two signals by comparing the corresponding sparse representations. Inversely, the similarity of two atoms can be determined by comparing how many signals use them simultaneously, and how they contribute, in their sparse decomposition. We compute two matrices representing each one of these cases respectively:

Clustering the signals: Construct a similarity matrix $\mathbf{S}_1 \in \mathbb{R}^{m \times m}$, $\mathbf{S}_1 := |\mathbf{A}^T \mathbf{A}|$.

Clustering the atoms: Construct a similarity matrix $\mathbf{S}_2 \in \mathbb{R}^{k_0 \times k_0}$, $\mathbf{S}_2 := |\mathbf{A} \mathbf{A}^T|$.

In both cases the similarity matrix obtained is positive semidefinite and can be associated with a graph, $G_1 := \{\mathbf{X}, \mathbf{S}_1\}$ and $G_2 := \{\mathbf{D}, \mathbf{S}_2\}$, where the data or the atoms are the sets of vertexes with the corresponding \mathbf{S}_i as edge weights matrixes. This graph is partitioned using standard spectral clustering algorithms to obtain the initialization for the algorithm described in the previous section.

As we mentioned before, G_1 is closely related with the ℓ_1 -graph. In that case, the weights of the graph are determined using the sparse decomposition of the signals with the data itself as a dictionary. When the number of signals m is large, the computational cost of constructing the similarity matrix is too expensive. Also the spectral clustering algorithm requires the computation of the largest singular values (and corresponding singular vectors), which is also computationally demanding when m is large (although not so demanding if only a few eigenvectors are needed). In the case of G_2 , clustering the atoms bypasses these difficulties, the size of \mathbf{S}_2 depends on the significantly smaller size of the

initial dictionary k_0 . This parameter does not depend on the amount of data, it just needs to be large enough to model it properly, and is often just in the hundreds. Note that the obtained sub-dictionaries may have different cardinalities (different k_i), reflecting different complexities of the associated clusters.

When the number of clusters, K , is large, the performance of the initial clusterization decreases. We propose a more robust initialization. Starting with the whole set as the only partition, at each iteration we subdivide in two sets each of the current partitions, keeping the division that produces the biggest decrease in the cost energy defined in Equation (A.6). The procedure stops when the desired number of clusters is reached. This can be applied for any of the two graphs presented in this section, and such partition is consistent with the energy driving the clustering.

4.2 Theoretical Guarantees

In this Section we show that, under certain ideal conditions, one can prove that the initialization step presented in the previous section produces a perfect clustering of the data. Because this assumptions do not hold in general with real data, the result of the initial step does not always give a correct clustering, but it gives a very good first approximation that is be later refined by the iterative step.

Following the ideas presented in [16], let us consider the ideal situation in which every signal in the K clusters can be exactly reconstructed as a sparse linear combination of the atoms of a dictionary and that the subspace that they span (using all the atoms) are independent, this is, that their sum is direct. Let us call those subspaces S_i , $i = 1, \dots, K$. Now, assume that the initial dictionary is composed of K (redundant) sub-dictionaries, $\mathbf{D}_0 = [\mathbf{D}_1, \dots, \mathbf{D}_K]$, one corresponding to each cluster in the dataset. For simplicity we assume that the atoms of the dictionary \mathbf{D}_0 are ordered but this is not required for proving any of the results discussed here.

Dataset	k-means	$\eta = 0$	$\eta \neq 0$
MNIST	21.2	6.9	3.0
USPS	22.3	2.9	2.0
ISOLET	20.0	6.0	1.5
Brodatz(x2)	-	2.5	0.4

Table A.2: Error rate (in percentage) for the clustering algorithm discussed in Section 4. In MNIST and USPS we used digits form 0 to 5 and for ISOLET we used the last 6 letters. We also tested clustering combinations of 2 randomly chosen Brodatz textures. In this case, the result is the average performance over 10 random realizations. In all cases, the results are shown with ($\eta \neq 0$), and without ($\eta = 0$) added incoherence.

Then, given a vector \mathbf{x} belonging to one of the subspaces, it is easy to show that the optimal \mathbf{a} in the ℓ_1 -relaxation of ℓ_0 with this \mathbf{D}_0 , will use only atoms from the correct block of the initial dictionary, producing K connected components in both graphs \mathbf{G}_1 and \mathbf{G}_2 . In this situation a spectral clustering technique will successfully separate the clusters [32].

The hypothesis from [16] that the subspaces span independent subspaces is very strong. In practice, different clusters very often have non trivial intersections. This is exactly what is tackled by not considering highly coherent atoms in the comparison of the quality of the sparse representations presented in Section 3.1. One can still prove that the solution to the ℓ_0 problem will pick atoms from the correct block for a given $\mathbf{x} \in S_i$ if the atoms of the dictionaries that compose \mathbf{D}_0 satisfy $\max_d \|\mathbf{D}_i^\dagger \mathbf{d}\|_1 < 1$, where \mathbf{D}_i^\dagger is the Moore-Penrose pseudoinverse of \mathbf{D}_i and \mathbf{d} is any atom of the dictionaries \mathbf{D}_j with $j \neq i$. This condition is similar to the one required for the exact recovery of the orthogonal matching pursuit algorithm [12]. It is related to the incoherence between atoms belonging to different dictionaries. The proof can be made following similar ideas to the ones used in that case, and is here omitted due to space limitations.

4.3 Clustering Results

We now apply the proposed algorithm to several clustering problems and texture segmentation. We first clustered the digits from 0 to 5 ($K = 6$) from the testing set of MNIST and the training set of USPS (ignoring the labels, of course). We also clustered the last six letters of ISOLET ($K = 6$), combining the standard training and testing sets. We further applied the clustering scheme to the combined patches of randomly chosen samples of 2 textures from the Brodatz database. The results are reported in Table A.2 for different values of the incoherence penalty term η . The size of the initial dictionaries are $k = 120$ for USPS, $k = 300$ for MNIST and $k = 90$ for ISOLET. The dictionaries representing each cluster are of size 60, 25 and 15 respectively. The initial clustering of the data was done using spectral clustering on the graph G_1 . In all the cases involving images, the atoms learned for each cluster were visually identifiable with the classes they represented.

Effect of imposing incoherence: As can be seen in Table A.2, encouraging incoherence in the dictionaries is of paramount importance, first to the initial dictionaries as internal incoherence, and then between dictionaries during the iterative clustering. To further understand this effect, we show in Figure A.3 how the strength of the incoherence term, controlled by the parameter η , affects the reconstruction error. Also shown is the actual average incoherence obtained between the cluster dictionaries, that is the average value of $|\mathbf{d}_i^T \mathbf{d}_j|$ between all possible pairs of atoms \mathbf{d}_i and \mathbf{d}_j from all the dictionaries involved.

Effect of the iterative clustering: In Figure A.4 we show an example of how the classification error is monotonically reduced for successive iterations of the proposed algorithm, thus empirically assessing its stability.

Texture segmentation: We also apply our clustering algorithm for the texture segmentation problem. The goal is to assign each pixel on an objective image to one of K possible textures. The approach is related to the one used in [4] for the supervised case. Overlapping 16×16 patches from the original images and used as input signals.

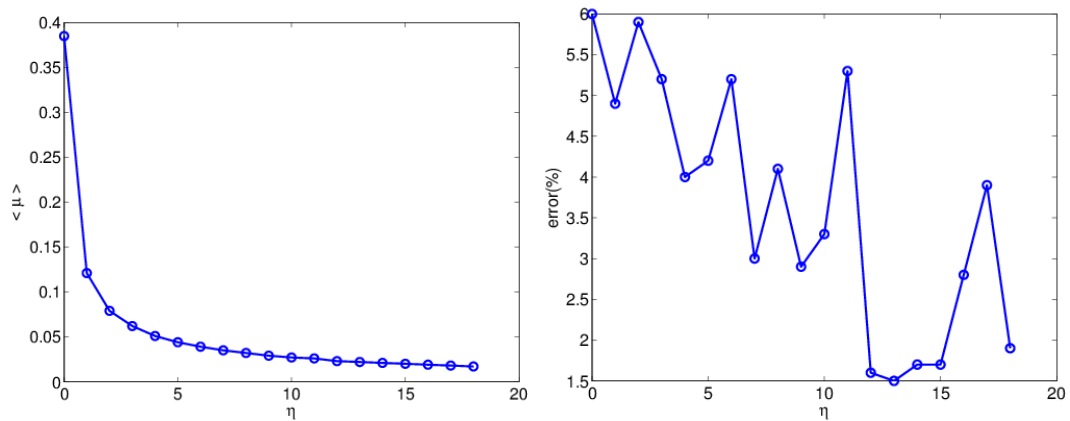


Figure A.3: Effect of incoherence in clustering performance of the ISOLET database. Left: average incoherence between all the dictionaries vs. η . Right: classification error vs. η . Note that, due to the small size of this dataset, fluctuations of $\pm 1\%$ such as the ones here observed are common.

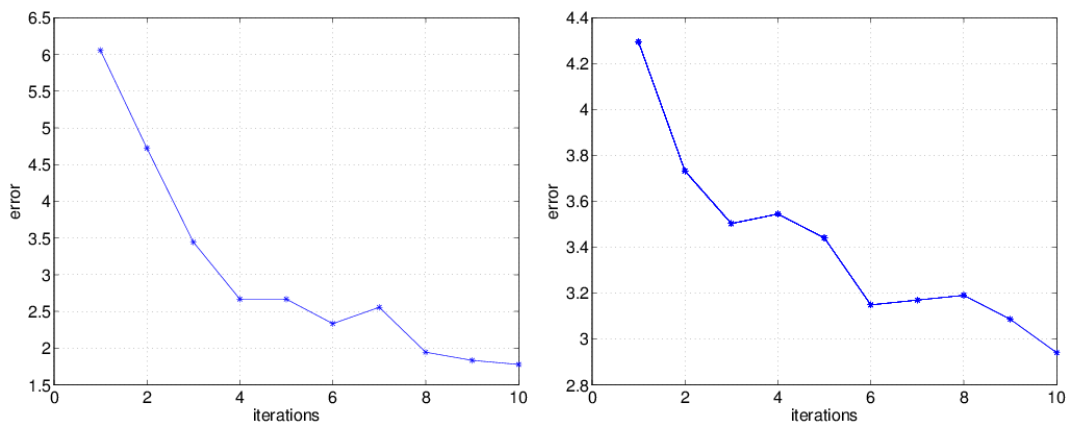


Figure A.4: Classification error vs. number of iterations for the clustering algorithm when applied to the ISOLET (left) and USPS (right) datasets.

After each iteration (that is, before recomputing the dictionaries), we obtain K different energy values for each patch, each corresponding to a candidate texture class. Following the same strategy than in the object detection example of Section 3, we use the energy value obtained for each patch to construct K different “energy images,” and combine such energies to obtain the pixel-wise classification.

In Figure A.5 we show some of the results. The number of patches extracted was on the order of several thousands, so the initialization with G_2 was applied. The algorithm gave sub-dictionaries that have a cardinality that intuitively reflects the complexity of the corresponding texture (in other words, k_i was not constant). We got very low rates of missclassified pixels, for example in Figure A.5(c), where we obtained 0.25%, which is better than the 0.37% obtained in [3] for the supervised case (which was, as far as we know, the best reported result in the literature for that image).

5 Concluding Remarks

A framework for classification and clustering based on dictionary learning and sparse representations was introduced in this paper. The basic idea is to simultaneously learn a set of dictionaries that optimally represent each one of the classes. Toward this goal, we introduced a new measurement of representation quality, a new term that promotes incoherence between the dictionaries, and an initialization procedure that combines sparse coding, dictionary learning, and spectral clustering. The clusters are allowed to share atoms. The obtained model is much richer than standard subspace clustering algorithms, since multiple subspaces represent a given class, and classes can have intersecting subspaces. Soft-clustering can be obtained as well in this framework, and the experimental results can be further improved using these soft measures in an SVM.

While we considered signals sparsely represented each one by a single dictionary, this

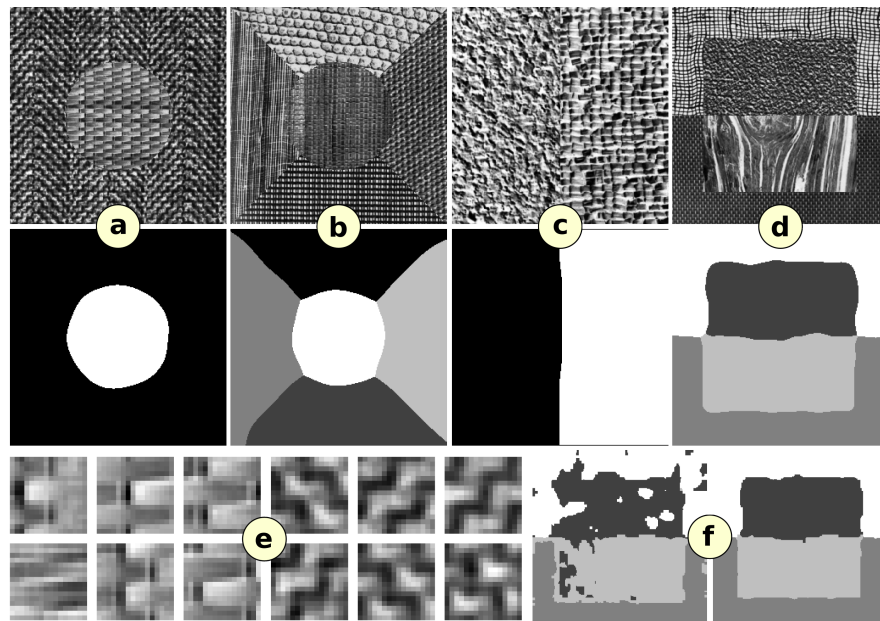


Figure A.5: Texture segmentation results on mosaics from the Brodatz database. (a)-(d) above are the mosaics and below the respective segmentation. The missclassification rates are 1.75%, 4.25%, 0.25% and 3.4% respectively. In (e) we show sample atoms from the final cluster dictionaries for the textures in (a). The texture in the circle required $k_1 = 82$ atoms, while the other one received $k_2 = 118$, which goes along with the intuition of larger complexity for this texture. (f) shows the first and third iteration of the iterative clustering algorithm.

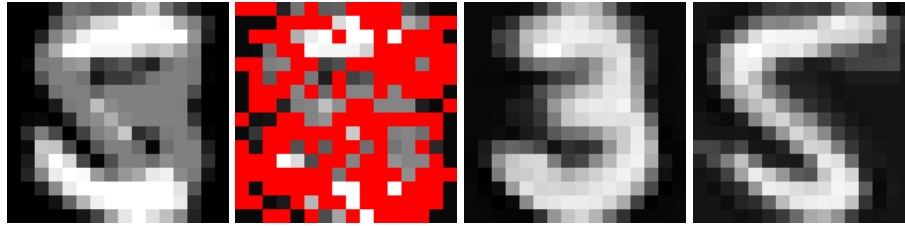


Figure A.6: Example of the recovered digits from a mixture (left) with 60% of missing components (red pixels in second figure), each digit is sparsely represented in its own learned dictionary, and the signal is the result of the sum of two digits (mixture of two dictionaries). As with the case in this paper with signals composed from a single dictionary, the technique described in [33] finds from the corrupted mixture, the correct classification classes (dictionaries) and the reconstruction (last two images).

can be extended to signals resulting from mixtures of dictionaries [33], Figure A.6.

Acknowledgments: Work supported by ONR, NGA, NSF, and ARO. Code follows [34].

References

- [1] A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Rev.*, 51(1):34–81, 2009.
- [2] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, volume 19, pages 801–808. MIT Press, 2007.
- [3] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *CVPR*, 2008.
- [4] G. Peyré. Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, 34:17–31, May 2009.
- [5] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.
- [6] P. Sprechmann and G. Sapiro. Dictionary learning and sparse coding for unsupervised clustering. In *Proc. ICASSP*, Mach 2010.
- [7] Y. C. Eldar and M. Mishali. Robust recovery of signals from a structured union of subspaces. *IEEE Trans. IT.*, 2009.
- [8] A. Ganesh, Z. Zhou, and Y. Ma. Separation of a subspace-sparse signal: Algorithms and conditions. In *ICASSP*, volume 14, april 2009.

- [9] P. Tseng. Nearest q -flat to m points. *J. Optim. Theory Appl.*, 105(1):249–252, 2000.
- [10] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. on Pure and Applied Mathematics*, 57:1413–1457, 2004.
- [11] M. Elad. Optimized projections for compressed-sensing. *IEEE Trans. SP*, 55(12):5695–5702, Dec. 2007.
- [12] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. IP*, 50:2231–2242, 2004.
- [13] A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. PAMI*, 29(5):854–869, 2007.
- [14] I. Ramirez, F. Lecumberry, and G. Sapiro. Universal priors for sparse modeling. In *Int. Workshop on Computational Advances in Multi-Sensor Adaptive Proc.*, December 2009.
- [15] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, pages 849–856. MIT Press, 2001.
- [16] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *CVPR*, 2009.
- [17] B.V. Gowreesunker and Ahmed H. Tewfik. A novel subspace clustering method for dictionary design. In *ICA, Lecture Notes in Computer Science*, pages 34–41. Springer, 2009.
- [18] S. R. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *CVPR*, 2008.
- [19] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan. Sparse representations for computer vision and pattern recognition. In *Proc. IEEE*, 2010, to appear.
- [20] H. Cheng, Z. Liu, and J. Yang. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *ICCV*, 2009.
- [21] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.
- [22] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, 2009.
- [23] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19, 1974.
- [24] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *NIPS*, volume 21, pages 1033–1040. MIT Press, 2009.
- [25] K. Huang and S. Aviyente. Sparse representation for signal classification. In *NIPS*, 2006.

- [26] A.D. Szlam and G. Sapiro. Discriminative k -metrics. In *ICML*, 2009.
- [27] K. Etemad and R. Chellappa. Separability-based multiscale basis selection and feature extraction for signal and image classification. *IEEE Transactions on Image Processing*, 7:1453–1465, 1998.
- [28] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *IEEE Trans. on PAMI*, 28(3), March 2006.
- [29] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.
- [30] Caroline Pantofaru, Gyuri Dorko, Cordelia Schmid, and Martial Hebert. Combining regions and patches for object class localization. In *The Beyond Patches Workshop. CVPR*, 2006.
- [31] T. Tuytelaars and C. Schmid. Vector quantizing feature space with a regular lattice. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007.
- [32] Ulrike von Luxburg. A tutorial on spectral clustering. *CoRR*, abs/0711.0189, 2007.
- [33] P. Sprechmann, I. Ramirez, G. Sapiro, and Y. Eldar. Collaborative hierarchical sparse modeling. In *Annual Conference on Information Sciences and Systems*, March 2010.
- [34] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:16–60, 2010.

B

C-HiLasso: A Collaborative Hierarchical Sparse Modeling Framework

Pablo Sprechmann,^{1†} Ignacio Ramírez,^{1†} Guillermo Sapiro¹ and Yonina C. Eldar²

¹University of Minnesota and ²Technion

Sparse modeling is a powerful framework for data analysis and processing. Traditionally, encoding in this framework is performed by solving an ℓ_1 -regularized linear regression problem, commonly referred to as *Lasso* or *Basis Pursuit*. In this work we combine the sparsity-inducing property of the Lasso at the individual feature level, with the block-sparsity property of the *Group Lasso*, where sparse groups of features are jointly encoded, obtaining a sparsity pattern hierarchically structured. This results in the *Hierarchical Lasso (HiLasso)*, which shows important practical advantages. We then extend this approach to the collaborative case, where a set of simultaneously coded signals share the same sparsity pattern at the higher (group) level, but not necessarily at the lower (inside the group) level, obtaining the collaborative HiLasso model (*C-HiLasso*). Such signals then share the same active groups, or classes, but not necessarily the same active set. This model is very well suited for applications such as source identification and separation. An efficient optimization procedure, which guarantees convergence to the global optimum, is developed for these new models. The underlying presentation of the framework and optimization approach is complemented by experimental examples and theoretical results regarding recovery guarantees.

1 Introduction and Motivation

Sparse signal modeling has been shown to lead to numerous state-of-the-art results in signal processing, in addition to being very attractive at the theoretical level. The standard model assumes that a signal can be efficiently represented by a sparse linear combination of atoms from a given or learned dictionary. The selected atoms form what is usually referred to as the *active set*, whose cardinality is significantly smaller than the size of the dictionary and the dimension of the signal.

In recent years, it has been shown that adding structural constraints to this active set has value both at the level of representation robustness and at the level of signal interpretation (in particular when the active set indicates some physical properties of the signal); see [1, 2, 3] and references therein. This leads to *group* or *structured* sparse coding, where instead of considering the atoms as singletons, the atoms are grouped, and a few groups are active at a time. An alternative way to add structure (and robustness) to the problem is to consider the simultaneous encoding of multiple signals, requesting that they all share the same active set. This is a natural collaborative filtering approach to sparse coding; see, for example, [4, 5, 6, 7, 8, 9].

In this work we extend these approaches in a number of directions. First, we present a hierarchical sparse model, where not only a few (sparse) groups of atoms are active at a time, but also each group enjoys internal sparsity.¹ At the conceptual level, this means that the signal is represented by a few groups (classes), and inside each group only a few members are active at a time. A simple example of this is a piece of music (numerous applications in genomics and image processing exist as well), where only a few instruments are active at a time (each instrument is a group), and the sound produced by each instrument at each instant is efficiently represented by a few atoms of the sub-dictionary/group corresponding to it. Thereby, this proposed hierarchical sparse coding framework permits

¹While we consider only 2 levels of sparsity, the proposed framework is easily extended to multiple levels.

to efficiently perform source identification and separation, where the individual sources (classes/groups) that generated the signal are identified at the same time as their representation is reconstructed (via the sparse code inside the group). An efficient optimization procedure, guaranteed to converge to the global optimum, is proposed to solve the hierarchical sparse coding problems that arise in our framework. Theoretical recovery bounds are derived, which guarantee that the output of the optimization algorithm is the true underlying signal.

Next, we go one step beyond. Continuing with the above example, if we know that the same few instruments will be playing simultaneously during different passages of the piece, then we can assume that the active groups at each instant, within the same passage, will be the same. We can exploit this information by applying the new hierarchical sparse coding approach in a collaborative way, enforcing that the same groups will be active at all instants within a passage (since they are of the same instruments and then efficiently representable by the same sub-dictionaries), while allowing each group for each music instant to have its own unique internal sparsity pattern (depending on how the sound of each instrument is represented at each instant). We propose a collaborative hierarchical sparse coding framework following this approach, (*C-HiLasso*), along with an efficient optimization procedure. We then comment on results regarding the correct recovery of the underlying active groups.

The proposed optimization techniques for both *HiLasso* and *C-HiLasso* is based on the Proximal Method [10], more specifically, on its particular implementation for sparse problems, *Sparse Reconstruction by Separable Approximation* (sparsa) [11]. This is an iterative method which solves a subproblem at each iteration which, in our case, has a closed form and can be solved in linear time. Furthermore, this closed form solution combines a vector thresholding and a scalar thresholding, naturally yielding to the desired hierarchical sparsity patterns.

The rest of the paper is organized as follows: Section 2 provides an introduction to

traditional sparse modeling and presents our proposed HiLasso and C-HiLasso models. We discuss their relationship with the recent works of [2, 12, 13, 14, 15, 16]. In Section 3 we describe the optimization techniques applied to solve the resulting sparse coding problems and we discuss its relationship with other optimization methods recently proposed in the literature [15, 17]. Theoretical recovery guarantees for HiLasso in the noiseless setting are developed in Section 4, demonstrating improved performance when compared with Lasso and Group Lasso. We also comment on existing results regarding correct recovery of group-sparse patterns in the collaborative case. Experimental results and simulations are given in Section 5, and finally concluding remarks are presented in Section 6.

2 Collaborative Hierarchical Sparse Coding

2.1 Background: Lasso and Group Lasso

Assume we have a set of data samples $\mathbf{x}_j \in \mathbb{R}^m, j = 1, \dots, n$, and a dictionary of p atoms in \mathbb{R}^m , assembled as a matrix $\mathbf{D} \in \mathbb{R}^{m \times p}, \mathbf{D} = [\mathbf{d}_1 \mathbf{d}_2 \dots \mathbf{d}_p]$. Each sample \mathbf{x}_j can be written as $\mathbf{x}_j = \mathbf{D}\mathbf{a}_j + \epsilon$, $\mathbf{a}_j \in \mathbb{R}^p, \epsilon \in \mathbb{R}^m$, that is, as a linear combination of the atoms in the dictionary \mathbf{D} plus some perturbation ϵ , satisfying $\|\epsilon\|_2 \ll \|\mathbf{x}_j\|_2$. The basic underlying assumption in sparse modeling is that, for all or most j , the “optimal” \mathbf{a}_j has only a few nonzero elements. Formally, if we define the ℓ_0 cost as the pseudo-norm counting the number of nonzero elements of \mathbf{a}_j , $\|\mathbf{a}_j\|_0 := |\{k : a_{kj} \neq 0\}|$, then we expect that $\|\mathbf{a}_j\|_0 \ll p$ and $\|\mathbf{a}_j\|_0 \ll m$ for all or most j .

Seeking the sparsest representation \mathbf{a} is known to be NP-hard. To determine \mathbf{a}_j in practice, a multitude of efficient algorithms have been proposed, which achieve high correct recovery rates. The ℓ_1 -minimization method is the most extensively studied recovery technique. In this approach, the non-convex ℓ_0 norm is replaced by the convex ℓ_1 norm, leading

to

$$\min_{\mathbf{a} \in \mathbb{R}^p} \|\mathbf{a}\|_1 \quad \text{s.t.} \quad \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 \leq \epsilon. \quad (2.1)$$

The use of general purpose or specialized convex optimization techniques allows for efficient reconstruction using this strategy. The above approximation is known as the Lasso [18] or Basis Pursuit [19, 20]. A popular variant is to use the unconstrained version

$$\min_{\mathbf{a} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_1, \quad (2.2)$$

where λ is an appropriate parameter value, usually found by cross-validation, or based on statistical principles [21].

The fact that the $\|\cdot\|_1$ regularizer induces sparsity in the solution \mathbf{a}_j is desirable not only from a regularization point of view, but also from a model selection perspective, where one wants to identify the relevant factors (atoms) that conform each sample \mathbf{x}_j . In many situations, however, the goal is to represent the relevant factors not as singletons but as groups of atoms. For a dictionary of p atoms, we define groups of atoms through their indices, $G \subseteq \{1, \dots, p\}$. Given a group G of indexes, we denote the sub-dictionary of the columns indexed by them as $\mathbf{D}_{[G]}$, and the corresponding set of reconstruction coefficients as $\mathbf{a}_{[G]}$. Define $\mathcal{G} = \{G_1, \dots, G_q\}$ to be a partition of $\{1, \dots, p\}$.² In order to perform model selection at the group level (relative to the partition \mathcal{G}), the Group Lasso problem was introduced in [1],

$$\min_{\mathbf{a} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \psi_{\mathcal{G}}(\mathbf{a}), \quad (2.3)$$

where $\psi_{\mathcal{G}}$ is the Group Lasso regularizer defined in terms of \mathcal{G} as $\psi_{\mathcal{G}}(\mathbf{a}) := \sum_{G \in \mathcal{G}} \|\mathbf{a}_{[G]}\|_2$. The function $\psi_{\mathcal{G}}$ can be seen as a generalization of the ℓ_1 regularizer, as the latter arises from the special case $\mathcal{G} = \{\{1\}, \{2\}, \dots, \{p\}\}$ (the groups are singletons), and as such, its

²While in this paper we concentrate and develop the important non-overlapping case, it will be clear that the concepts of collaborative hierarchical sparse modeling introduced here apply to the case of overlapping groups as well.

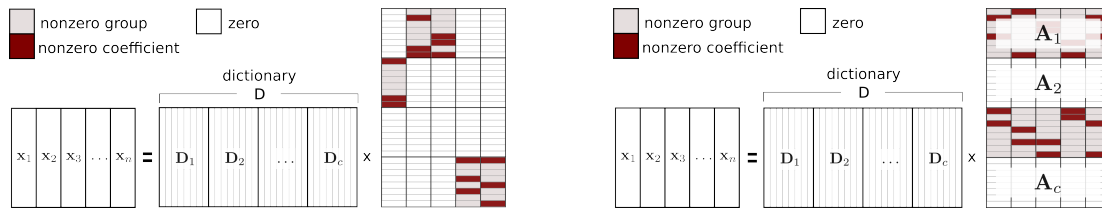


Figure B.1: Sparsity patterns induced by HiLasso (left) and C-HiLasso (right) model selection programs. Notice that the C-HiLasso imposes the same group-sparsity pattern in all the samples (same class), whereas the in-group sparsity patterns can vary between samples (samples themselves are different).

effect on the groups of \mathbf{a} is also a natural generalization of the one obtained with the Lasso: it “turns on/off” atoms in groups.

We can always consider the “noiseless” sparse coding problem $\min_{\mathbf{a} \in \mathbb{R}^p} \{\psi(\mathbf{a}) : \mathbf{x}_j = \mathbf{D}\mathbf{a}\}$, for a generic regularizer $\psi(\cdot)$, as the limit of the Lagrangian sparse coding problem $\min_{\mathbf{a} \in \mathbb{R}^p} \left\{ \frac{1}{2} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \lambda\psi \right\}$ when $\lambda \rightarrow 0$. In the remainder of this section, as well as in Section 3, we only present the corresponding Lagrangian formulations.

2.2 The Hierarchical Lasso

The Group Lasso trades sparsity at the single-coefficient level with sparsity at a group level, while, inside each group, the solution is generally dense. Let us consider for example that each group is a sub-dictionary trained to efficiently represent, via sparse modeling, an instrument, a type of image, or a given class of signals in general. The entire dictionary \mathbf{D} is then appropriate to represent all classes of the signal as well as mixtures of them, and Group Lasso will properly represent (dense) mixtures with one group or sub-dictionary per class. At the same time, since each class is properly represented in a sparse mode via its corresponding group or sub-dictionary, we expect sparsity inside its groups as well (which is not achieved by Group Lasso, whose solutions are dense inside each group). This will become even more critical in the collaborative case, where signals will share groups because

they are of the same class, but will not necessarily share the full active sets, since they are not the same signal. To achieve the desired in-group sparsity, we simply re-introduce the ℓ_1 regularizer together with the group regularizer, leading to the proposed *Hierarchical Lasso* (*HiLasso*) model,³

$$\min_{\mathbf{a} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \lambda_2 \psi_{\mathcal{G}}(\mathbf{a}) + \lambda_1 \|\mathbf{a}\|_1. \quad (2.4)$$

The hierarchical sparsity pattern produced by the solutions of (2.4) is depicted in Figure B.1(left). For simplicity of the description, we assume that all the groups have the same number of elements. The extension to the general case is obtained by multiplying each group norm by the square root of the corresponding group size. This model then achieves the desired effect of promoting sparsity at the group/class level while at the same time leading to overall sparse feature selection. As mentioned above, additional levels of hierarchy can be considered as well, e.g., with groups inside the blocks. This is relevant for example in audio analysis.

As with models such as Lasso and Group Lasso, the optimal parameters λ_1 and λ_2 are application and data dependent. In some specific cases, closed form solutions exist for such parameters. For example, for signal restoration in the presence of noise using Lasso ($\lambda_2 = 0$), the `GSURE` method provides a simple way to compute the optimal λ_1 [21]. As extending such methods to HiLasso (or the C-HiLasso model presented next) is beyond the scope of this work, we rely on cross-validation for the choice of such parameters. The selection of λ_1 and λ_2 has an important influence on the sparsity of the obtained solution. Intuitively, as λ_2/λ_1 increases, the group constraint becomes dominant and the solution tends to be more sparse at a group level but less sparse within groups (see Figure B.2). This relation allows in practice to intuitively select a set of parameters that performs well. We also noticed empirically that the selection of the parameters is quite robust, since small variations in their numerical value don't change considerably the obtained results.

³We can similarly define a hierarchical sparsity model with ℓ_0 instead of ℓ_1 .

Some recent modeling frameworks for sparse coding do not rely on the selection of such model parameters, e.g., following the Minimum Description Length criterion in [22], or non-parametric Bayesian techniques in [23]. Applying such techniques to the here proposed models is subject of future research.

2.3 Collaborative Hierarchical Lasso

In numerous applications, one expects that certain collections of samples \mathbf{x}_j share the same active components from the dictionary, that is, that the indices of the nonzero coefficients in \mathbf{a}_j are the same for all the samples in the collection. Imposing such dependency in the ℓ_1 regularized regression problem gives rise to the so called collaborative (also called “multitask” or “simultaneous”) sparse coding problem [4, 8, 9, 24]. Considering the coefficients matrix $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n] \in \mathbb{R}^{p \times n}$ associated with the reconstruction of the samples $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$, this model is given by

$$\min_{\mathbf{A} \in \mathbb{R}^{p \times n}} \frac{1}{2} \|\mathbf{X} - \mathbf{DA}\|_F^2 + \lambda \sum_{k=1}^p \|\mathbf{a}^k\|_2, \quad (2.5)$$

where $\mathbf{a}^k \in \mathbb{R}^n$ is the k -th row of \mathbf{A} , that is, the vector of the n different values that the coefficient associated to the k -th atom takes for each sample $j = 1, \dots, n$. If we now extend this idea to the Group Lasso, we obtain a *collaborative Group Lasso (C-GLasso)* formulation,

$$\min_{\mathbf{A} \in \mathbb{R}^{p \times n}} \frac{1}{2} \|\mathbf{X} - \mathbf{DA}\|_F^2 + \lambda \psi_{\mathcal{G}}(\mathbf{A}), \quad (2.6)$$

where $\psi_{\mathcal{G}}(\mathbf{A}) = \sum_{G \in \mathcal{G}} \|\mathbf{A}^G\|_F$, and \mathbf{A}^G is the sub-matrix formed by all the rows belonging to group G . This regularizer is the natural collaborative extension of the regularizer in (2.3).

In this paper, we take an additional step and treat this together with the hierarchical

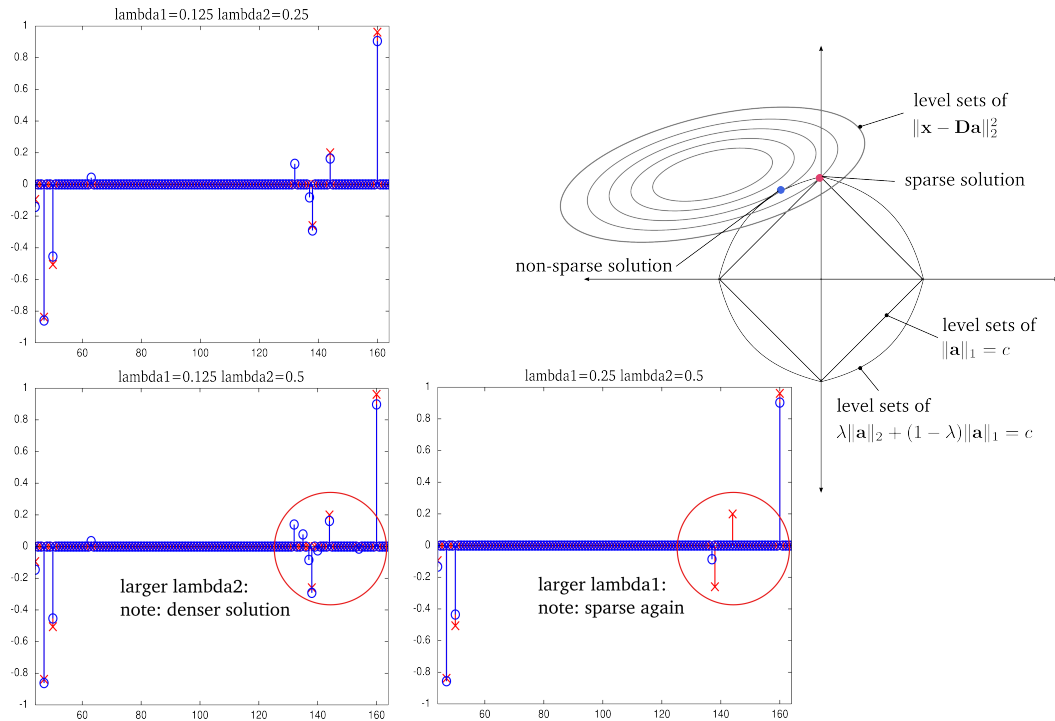


Figure B.2: Effect of different combinations of λ_1 and λ_2 on the solutions of the HiLasso coding problem. Three cases are given in which we want to recover a sparse signal (red crosses) \mathbf{a}_0 by means of the solution \mathbf{a} of the HiLasso problem (blue dots). In this example we have two active groups out of ten possible (the sub dictionaries associated to each group have 30 atoms) and $\mathbf{a}_0 = 8$ (four non-zero coefficient per active group). The estimate that is closest to \mathbf{a}_0 in ℓ_1 norm is shown in the top left. As the ratio λ_2/λ_1 increases (bottom left), the level sets of the regularizer $\psi_{\mathcal{G}}(\cdot)$ become rounder, thus encouraging denser solutions. This is depicted in the rightmost figure for a simple case of $q = 1$ groups. Increasing λ_1 again (bottom right) increases sparsity, although here the final effect is too strong and some non-zero coefficients are not detected.

extension presented in the previous section. The combined model that we propose, *C-HiLasso*, is given by

$$\min_{\mathbf{A} \in \mathbb{R}^{p \times n}} \frac{1}{2} \|\mathbf{X} - \mathbf{DA}\|_F^2 + \lambda_2 \psi_{\mathcal{G}}(\mathbf{A}) + \lambda_1 \sum_{j=1}^n \|\mathbf{a}_j\|_1. \quad (2.7)$$

The sparsity pattern obtained using (2.7) is shown in Figure B.1 (right). The C-GLasso is a particular case of our model when $\lambda_1 = 0$. On the other hand, one can obtain independent Lasso solutions for each \mathbf{x}_i by setting $\lambda_2 = 0$. We see that (2.7) encourages all the signals to share the same groups (classes), while the active set inside each group is signal dependent. We thereby obtain a collaborative hierarchical sparse model, with collaboration at the class level (all signals collaborate to identify the classes), and freedom at the individual levels inside the class to adapt to each particular signal. This new model is particularly well suited, for example, when the data vectors have missing components. In this case combining the information from all the samples is very important in order to obtain a correct representation and model (group) selection. This can be done by slightly changing the data term in (2.7). For each data vector \mathbf{x}_j one computes the reconstruction error using only the observed elements. Note that the missing components do not affect the other terms of the equation. Examples will be shown in Section 5.

2.4 Relationship to Recent Literature

A number of recent works have addressed hierarchy, grouping and collaboration within the sparse modeling community. We now discuss the ones most closely related to the proposed HiLasso and C-HiLasso models.

In [2], the authors propose a general framework in which one can define a regularization term to encourage a variety of sparsity patterns, and provide theoretical results

(different to the ones developed here) for the single-signal case. The HiLasso model presented here, in the single signal scenario, can be seen as a particular case of that model (where the groups in [2] should be blocks and singletons), although the particularly and important case of hierarchical structure introduced here is not mentioned in that paper. In [12] the authors simultaneously (see [25]) proposed a model that coincides with ours again in the single-signal scenario. None of these approaches develop the collaborative framework introduced here, nor the theoretical guarantees. The recovery of mixed signals with ℓ_0 optimization was addressed in [16]. This model does not include block sparsity (no hierarchy), collaboration, or the theoretical results we obtain here.

The special case of C-HiLasso when $\lambda_1 = 0$, C-GLasso, is investigated in [26], where a theoretical analysis of the signal recovery properties of the model is developed. Collaborative coding with structured sparsity has also been used recently in the context of gene expression analysis [13, 14]. In [13], the authors propose a model, that can be interpreted as a particular case of the collaborative approach presented here, in which a set of signals is simultaneously coded using a small (sparse) number of atoms of the dictionary. They modify the classical collaborative sparse coding regularization so that each signal can use any subset of the detected atoms. This is equivalent to our model when the groups have only one element and therefore there is no hierarchy in the coding. A collaborative model is presented in [14], where signals sharing the same active atoms are grouped together in a hierarchical way by means of a tree structure. The regularization term proposed is analogous to the one proposed in our work, but it is used to group signals rather than atoms (features), having once again no hierarchical coding.

Tree-based sparse coding has also been used recently to learn dictionaries [15, 17]. Under this model, if a particular learned atom is not used in the decomposition of a signal, then none of its descendants (in terms of the given tree structure) can be used. Although not explicitly considered in these works, the HiLasso model is an important particular case,

among the wide spectrum of hierarchical sparse models considered in this line of work, where the hierarchy has two levels and no single atoms are in the upper level.

To conclude, while particular instances of the proposed C-HiLasso have been recently reported in the literature, none of them are as comprehensive. C-HiLasso includes both collaboration, at a block/group level, and hierarchical coding. Such collaborative hierarchical structure is novel and fundamental to address new important problems such as collaborative source identification and separation. The new theoretical results presented here extend the block sparsity results of [3, 27], complementing the modeling and algorithmic work.

3 Optimization

3.1 Single-Signal Problem: HiLasso

In the last decade, optimization of problems of the form of (2.2) and (2.3) have been deeply studied, and there exist very efficient algorithms for solving them. Recently, Wright et. al [11] proposed a framework, `spARSA`, for solving the general problem

$$\min_{\mathbf{a} \in \mathbb{R}^P} f(\mathbf{a}) + \lambda \psi(\mathbf{a}). \quad (3.8)$$

be a smooth and convex function, while $\psi : \mathbb{R}^P \rightarrow \mathbb{R}$ only needs to be finite and convex in \mathbb{R}^P . This formulation, which is a particular case of the Proximal Method framework developed by Nesterov [10], includes as important particular cases the Lasso, Group-Lasso and HiLasso problems by setting $f(\cdot)$ as the reconstruction error and then choosing the corresponding regularizers for $\psi(\cdot)$. When the regularizer, $\psi(\cdot)$, is group separable, the optimization can be subdivided into smaller problems, one per group. The framework becomes powerful when these sub-problems can be solved efficiently. This is the case of the Lasso and Group Lasso (with non overlapping groups) settings, and also of the HiLasso,

as we will show later in this Section. In all cases, the solution of the sub-problems are obtained in linear time.

The SPARSA algorithm generates a sequence of iterates $\{\mathbf{a}^{(t)}\}_{t \in \mathbb{N}}$ that, under certain conditions, converges to the solution of (3.8). At each iteration, $\mathbf{a}^{(t+1)}$ is obtained by solving

$$\min_{\mathbf{z} \in \mathbb{R}^p} (\mathbf{z} - \mathbf{a}^{(t)})^\top \nabla f(\mathbf{a}^{(t)}) + \frac{\alpha^{(t)}}{2} \|\mathbf{z} - \mathbf{a}^{(t)}\|_2^2 + \lambda \psi(\mathbf{z}), \quad (3.9)$$

for a sequence of parameters $\{\alpha^{(t)}\}_{t \in \mathbb{N}}$, $\alpha^{(t)} = \alpha_0 \eta^t$, where $\alpha_0 > 0$ and $\eta > 1$ need to be chosen properly for the algorithm to converge (see [11] for details). It is easy to show that (3.9) is equivalent to

$$\min_{\mathbf{z} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{z} - \mathbf{u}^{(t)}\|_2^2 + \frac{\lambda}{\alpha^{(t)}} \psi(\mathbf{z}), \quad (3.10)$$

where $\mathbf{u}^{(t)} = \mathbf{a}^{(t)} - \frac{1}{\alpha^{(t)}} \nabla f(\mathbf{a}^{(t)})$. In this new formulation, it is clear that the first term in the cost function can be separated element-wise. Thus, when the regularization function $\psi(\mathbf{z})$ is group separable, so is the overall optimization, and one can solve (3.10) independently for each group, leading to

$$\mathbf{a}_{[G]}^{(t+1)} = \arg \min_{\mathbf{z} \in \mathbb{R}^{|G|}} \frac{1}{2} \|\mathbf{z} - \mathbf{u}_{[G]}^{(t)}\|_2^2 + \frac{\lambda}{\alpha^{(t)}} \psi_{\mathcal{G}}(\mathbf{z}),$$

$\mathbf{z}_{[G]}$ being the corresponding variable for the group. In the case of HiLasso, this becomes,

$$\mathbf{a}_{[G]}^{(t+1)} = \arg \min_{\mathbf{z} \in \mathbb{R}^{|G|}} \frac{1}{2} \|\mathbf{z} - \mathbf{w}\|_2^2 + \frac{\lambda_2}{\alpha^{(t)}} \|\mathbf{z}\|_2 + \frac{\lambda_1}{\alpha^{(t)}} \|\mathbf{z}\|_1, \quad (3.11)$$

where we have defined $\mathbf{w} = \mathbf{u}_{[G]}^{(t)}$. Problem (3.11) is a second order cone program (SOCP), for which one could use generic solvers. However, since it needs to be solved many times within the SPARSA iterations, it is crucial to solve it efficiently. It turns out that (3.11) admits a closed form solution with cost linear in the dimension of \mathbf{w} . By inspecting the subgradient

of (3.11) for the case where the optimum $\mathbf{z}^* \neq 0$,

$$\mathbf{w} - \left(1 + \frac{\tilde{\lambda}_2}{\|\mathbf{z}^*\|_2}\right) \mathbf{z}^* \in \tilde{\lambda}_1 \partial \|\mathbf{z}^*\|_1,$$

where we have defined $\tilde{\lambda}_2 = \lambda_2/\alpha^{(t)}$ and $\tilde{\lambda}_1 = \lambda_1/\alpha^{(t)}$. If we now define $C(\mathbf{z}^*) = 1 + \tilde{\lambda}_2/\|\mathbf{z}^*\|_2$, we observe that each element of $C(\mathbf{z}^*)\mathbf{z}^*$ is the solution of the well known scalar soft thresholding operator,

$$z_i^* = \frac{1}{C(\mathbf{z}^*)} \text{sgn}(w_i) \max\{0, |w_i| - \tilde{\lambda}_1\} = \frac{h_i}{C(\mathbf{z}^*)}, \quad i = 1, \dots, g, \quad (3.12)$$

where we have defined $h_i = \text{sgn}(w_i) \max\{0, |w_i| - \tilde{\lambda}_1\}$, the result of the scalar thresholding of \mathbf{w} . Taking squares on both sides of (3.12) and summing over $i = 1, \dots, g$ we obtain

$$\|\mathbf{z}^*\|_2^2 = C^2(\mathbf{z}^*) \|\mathbf{h}\|_2^2 = \frac{\|\mathbf{z}^*\|_2^2}{(\|\mathbf{z}^*\|_2 + \tilde{\lambda}_2)^2} \|\mathbf{h}\|_2^2,$$

from which the equality $\|\mathbf{z}^*\|_2 = \|\mathbf{h}^*\|_2 - \tilde{\lambda}_2$ follows. Since all terms are positive, this can only hold as long as $\|\mathbf{h}^*\|_2 > \tilde{\lambda}_2$, which gives us a vector thresholding condition on the solution \mathbf{z}^* in terms of $\|\mathbf{h}\|_2$. It is easy to show that $\|\mathbf{h}^*\|_2 \leq \tilde{\lambda}_2$ is a sufficient condition for $\mathbf{z}^* = 0$. Thus we obtain,

$$\mathbf{a}_{[G]}^{(t+1)} = \begin{cases} \frac{\max\{0, \|\mathbf{h}\|_2 - \tilde{\lambda}_2\}}{\|\mathbf{h}\|_2} \mathbf{h} & , \quad \|\mathbf{h}\|_2 > 0 \\ \mathbf{0} & , \quad \|\mathbf{h}\|_2 = 0. \end{cases} \quad (3.13)$$

The above expression requires g scalar thresholding operations, and one vector thresholding, which is also linear with respecto to the group size g . Therefore, for all groups, the cost of solving the subproblem (3.11) is linear in m , the same as for Lasso and Group Lasso. The complete HiLasso optimization algorithm is summarized in Algorithm 2. The parameter η has very little influence in the overall performance (see [11] for details); we used

Algorithm 2: HiLasso optimization algorithm.

Input: Data \mathbf{X} , dictionary \mathbf{D} , group set \mathcal{G} , constants $\alpha_0 > 0$, $\eta > 1$, $c > 0$,
 $0 < \alpha_{\min} < \alpha_{\max}$
Output: The optimal point \mathbf{a}^*
Initialize $t := 0$, $\mathbf{a}^{(0)} := \mathbf{0}$;
while *stopping criterion is not satisfied* **do**
 choose $\alpha^{(t)} \in [\alpha_{\min}, \alpha_{\max}]$;
 set $\mathbf{u}^{(t)} := \mathbf{a}^{(t)} - \frac{1}{\alpha^{(t)}} \nabla f(\mathbf{a}^{(t)})$;
 while *stopping criterion is not satisfied* **do**
 // Here we use the group separability of (3.10) and solve (3.11) for each group
 for $i := 1$ to q **do**
 Compute $\mathbf{a}_{[G]}^{(t+1)}$ as the solution to (3.13);
 end
 set $\alpha^{(t+1)} := \eta \alpha^{(t)}$;
 end
 set $t := t + 1$;
end

$\eta = 2$ in all our experiments. Note that, as expected, the solution to the sub-problem for the cases $\lambda_2 = 0$ or $\lambda_1 = 0$, corresponds respectively to scalar soft thresholding and vector soft thresholding. In particular, when $\lambda_2 = 0$, the proposed optimization reduces to the Iterative Soft Thresholding algorithm [28].

3.2 Optimization of the Collaborative HiLasso

The multi-signal (collaborative) case is equivalent to the one-dimensional case where the signal is a concatenation of the columns of \mathbf{X} , and the dictionary is an $nm \times np$ block-diagonal matrix, where each of the n blocks is a copy of the original dictionary \mathbf{D} . However, in practice, it is not needed to build such (possibly very large) dictionary, and we can operate directly with the matrices \mathbf{D} and \mathbf{X} to find \mathbf{A} . If we define the matrix $\mathbf{U}^{(t)} \in \mathbb{R}^{m \times n}$ whose

i -th column is given by $\mathbf{u}_i^{(t)} = \mathbf{a}_i^{(t)} - \frac{1}{\alpha^{(t)}} \nabla f(\mathbf{a}_i^{(t)})$, we get the following SpARSA iterates,

$$\mathbf{A}^{(t+1)} = \arg \min_{\mathbf{Z} \in \mathbb{R}^{m \times n}} \frac{1}{2} \left\| \mathbf{Z} - \mathbf{U}^{(t)} \right\|_F^2 + \frac{\lambda_2}{\alpha^{(t)}} \|\mathbf{Z}\|_F + \frac{\lambda_1}{\alpha^{(t)}} \sum_{j=1}^n \|\mathbf{z}_j\|_1,$$

which again is group separable, so that it can be solved as q independent problems in the corresponding bands of $\mathbf{U}^{(t)}$,

$$(\mathbf{A}^{(t+1)})^G = \arg \min_{\mathbf{Z} \in \mathbb{R}^{g \times n}} \frac{1}{2} \left\| \mathbf{Z} - (\mathbf{U}^{(t)})^G \right\|_F^2 + \frac{\lambda_2}{\alpha^{(t)}} \|\mathbf{Z}\|_F + \frac{\lambda_1}{\alpha^{(t)}} \sum_{j=1}^n \|\mathbf{z}_j\|_1.$$

The correspondent closed form solutions for these subproblems, which are obtained in an analogous way to (3.12)–(3.13), are given by

$$(\mathbf{A}^{(t+1)})^G = \begin{cases} \frac{\max\{0, \|\mathbf{H}\|_F - \tilde{\lambda}_2\}}{\|\mathbf{H}\|_F} \mathbf{H} & , \quad \|\mathbf{H}\|_F > 0 \\ \mathbf{0} & , \quad \|\mathbf{H}\|_F = 0 \end{cases}, \quad h_{ij} = \text{sgn}(w_{ij}) \max\{0, |w_{ij}| - \tilde{\lambda}_1\}, \quad (3.14)$$

and we have defined $\mathbf{W} := (\mathbf{U}^{(t)})^G$.

As mentioned in Section 2.4, [17] addresses a wide spectrum of hierarchical sparse models for coding and dictionary learning. They propose a proximal method optimization procedure that, when restricted to the formulation of HiLasso, is very similar to the one developed in Section 3.1. The main difference with our method is that they solve the subproblem (3.10) using a dual approach (based on conic duality) that finds the exact solution in a finite number of operations. Our method, being tailored to the specific case of HiLasso, provides such solution in closed form, requiring just two thresholdings, both linear in the dimension of \mathbf{X} , $n \times m$.

4 Theoretical guarantees

In our current theoretical analysis, we study the case of a single measurement vector (signal) \mathbf{x} (we comment on the collaborative case at the end of this section), and assume that there is no measurement noise or perturbation, so that $\mathbf{x} = \mathbf{D}\mathbf{a}$. Without loss of generality, we further assume that the cardinality $|G_r| = g, r = 1, \dots, q$, that is, all groups in \mathcal{G} have the same size. The goal is to recover the code \mathbf{a} , from the observed \mathbf{x} , by solving the noise-free HiLasso problem:

$$\min_{\mathbf{a} \in \mathbb{R}^p} \{ \lambda \psi_{\mathcal{G}}(\mathbf{a}) + (1 - \lambda) \|\mathbf{a}\|_1 \quad \text{s.t.} \quad \mathbf{x} = \mathbf{D}\mathbf{a} \}. \quad (4.15)$$

Note that we have replaced the two regularization parameters λ_1 and λ_2 by a single parameter λ , since scaling does not effect the optimal solution. Therefore, we can always assume that $\lambda_1 + \lambda_2 = 1$.

Our goal is to develop conditions under which the HiLasso program of (4.15) will recover the true unknown vector \mathbf{a} . As we will see, the resulting set of recoverable signals is a superset of those recoverable by Lasso, that is, HiLasso is able to recover signals for which Lasso (or Group Lasso) will fail to do so.

We assume throughout this section that \mathbf{a} has group sparsity k , namely, no more than k of the group vectors $\mathbf{a}_{[G_i]}, i = 1, \dots, q$, have non-zero norm. In addition, within each group, we assume that not more than s elements are non zero, that is, $\|\mathbf{a}_{[G]}\|_0 \leq s$.

For $\lambda = 1$, (4.15) reduces to the Group Lasso problem, (2.3), whereas with $\lambda = 0$, (4.15) becomes equivalent to the Lasso problem, (2.2). Both cases have been treated previously in the literature and sufficient conditions have been derived on the sparsity levels and on the dictionary \mathbf{D} to ensure that the resulting optimization problem recovers the true unknown vector \mathbf{a} . For example, in [3, 29, 30], conditions are given in terms of the restricted isometry property (RIP) of \mathbf{D} . In an alternative line of work, recovery conditions are based

on coherence measures, which are easier to compute [27, 31]. Here, we follow the same spirit and consider coherence bounds that ensure recovery using the HiLasso approach. We also draw from [9] to briefly describe conditions under which the probability of error of recovering the correct groups, using the special case of the C-HiLasso with $\lambda_1 = 0$ (C-GLasso), falls exponentially to 0 as the number of collaborating samples n grows. Finally, recent theoretical results on block sparsity were reported in [32]. In particular, bounds on the number of measurements required for block sparse recovery were developed under the assumption that the measurement matrix \mathbf{D} has a basis of the null-space distributed uniformly in the Grassmanian. The model is a block-sparse model, without hierarchical or collaborative components.

In this section we extend the group-wise indexing notation to refer both to subsets of rows and columns of arbitrary matrices as $\mathbf{W}_{[F,G]} := \{w_{ij} : i \in F, j \in G\}$. This is, $\mathbf{W}_{[F,G]} = \mathbf{I}_{[F]}^T \mathbf{W} \mathbf{J}_{[G]}$, where \mathbf{I} and \mathbf{J} are the identity matrices of the column and row spaces of \mathbf{W} respectively. We define the sets $\Omega = \{1, 2, \dots, p\}$ and $\Gamma = \{1, 2, \dots, g\}$, and use \bar{S} to denote the complement of a set of indices S , either with respect to Ω or Γ , depending on the context. The set difference between S and T is denoted as $S \setminus T$, \emptyset represents the empty set, and $|S|$ denotes the cardinality of S .

4.1 Block-Sparse Coherence Measures

We begin by reviewing previously proposed coherence measures. For a given dictionary \mathbf{D} , the (standard) coherence is defined as $\mu := \max_{i,j \neq i \in \Gamma} |\mathbf{d}_i^T \mathbf{d}_j|$. This coherence was extended to the block-sparse setting in [27], leading to the definition of *block coherence*:

$$\mu_B := \max \left\{ \frac{1}{g} \rho(\mathbf{D}_{[G]}^T \mathbf{D}_{[F]}), G, F \in \mathcal{G}, G \neq F \right\},$$

where $\rho(\cdot)$ is the spectral norm, that is, $\rho(\mathbf{Z}) := \lambda_{\max}^{1/2}(\mathbf{Z}^T \mathbf{Z})$, with $\lambda_{\max}(\mathbf{W})$ denoting the largest eigenvalue of the positive semi-definite matrix \mathbf{W} . An alternate atom-wise measure of block coherence is given by the *cross coherence*,

$$\chi := \max \left\{ \max \left\{ |\mathbf{d}_i^T \mathbf{d}_j|, i \in G, j \in F \right\} \mid G, F \in \mathcal{G}, G \neq F \right\}. \quad (4.16)$$

When $g = 1$ (each block is a singleton), $\mathbf{D}_{[G_r]} = \mathbf{d}_r$, so that as expected, $\chi = \mu_B = \mu$. While μ_B and χ quantify global properties of the dictionary \mathbf{D} , local block properties are characterized by the *sub-coherence*, defined as

$$\nu := \max \left\{ \max \left\{ |\mathbf{d}_i^T \mathbf{d}_j|, i, j \in G, i \neq j \right\} \mid G \in \mathcal{G} \right\}. \quad (4.17)$$

We define $\nu = 0$ for $g = 1$. Clearly, if the columns of $\mathbf{D}_{[G]}$ are orthonormal for each group G , then $\nu = 0$. Assuming the columns of \mathbf{D} have unit norm, it can be easily shown that μ , ν , χ and μ_B all lie in the range $[0, 1]$. In addition, we can easily prove that $\nu, \mu_B, \chi \leq \mu$. In our setting, \mathbf{a} is block sparse, but has further internal structure: each sub-vector of \mathbf{a} is also sparse. In order to quantify our ability to recover such signals, we expect that an appropriate coherence measure will be based on the definition of block sparsity, but will further incorporate the internal sparsity as well. Let $\mathbf{M} := \mathbf{D}^T \mathbf{D}$ denote the Gram matrix of \mathbf{D} . Then, the standard block coherence μ_B is defined in terms of the largest singular value of an off-diagonal $g \times g$ sub-block of \mathbf{M} . In a similar fashion, we will define *sparse block coherence* measures in terms of *sparse singular values*. As we will see, two different definitions will play a role, depending on where exactly the sparsity within the block enters. To define these, we note that the *spectral norm* $\rho(\mathbf{Z})$ of a matrix \mathbf{Z} can be defined as

$$\rho(\mathbf{Z}) := \max_{\mathbf{u}, \mathbf{v}} |\mathbf{u}^T \mathbf{Z} \mathbf{v}| \quad \text{s.t.} \quad \|\mathbf{u}\|_2 = 1, \|\mathbf{v}\|_2 = 1.$$

Alternatively, we can define $\rho(\mathbf{Z})$ as the largest singular value of \mathbf{Z} , $\rho(\mathbf{Z}) := \sigma_{\max}(\mathbf{Z}) = \sqrt{\lambda_{\max}(\mathbf{Z}^T \mathbf{Z})}$,

$$\lambda_{\max}(\mathbf{Z}^T \mathbf{Z}) := \max_{\mathbf{v}} \mathbf{v}^T (\mathbf{Z}^T \mathbf{Z}) \mathbf{v} \quad \text{s.t.} \quad \|\mathbf{v}\|_2 = 1.$$

We now develop sparse analogs of $\rho(\mathbf{Z})$ and $\lambda_{\max}(\mathbf{Z}^T \mathbf{Z})$. As we will see, the simple square-root relation no longer holds in this case. The *largest sparse singular value* is defined as [33]:

$$\rho^{ss}(\mathbf{Z}) := \max_{\mathbf{u}, \mathbf{v}} |\mathbf{u}^T \mathbf{Z} \mathbf{v}| \quad \text{s.t.} \quad \|\mathbf{u}\|_2 = 1, \|\mathbf{v}\|_2 = 1, \|\mathbf{u}\|_0 \leq s, \|\mathbf{v}\|_0 \leq s. \quad (4.18)$$

Similarly, the *largest sparse eigenvalue* of $\mathbf{Z}^T \mathbf{Z}$ is defined as [33, 34, 35],

$$\lambda_{\max}^s(\mathbf{Z}^T \mathbf{Z}) := \max_{\mathbf{v}} \mathbf{v}^T (\mathbf{Z}^T \mathbf{Z}) \mathbf{v} \quad \text{s.t.} \quad \|\mathbf{v}\|_2 = 1, \|\mathbf{v}\|_0 \leq s. \quad (4.19)$$

The *sparse matrix norm* is then given by

$$\rho^s(\mathbf{Z}) := \sqrt{\lambda_{\max}^s(\mathbf{Z}^T \mathbf{Z})}. \quad (4.20)$$

Note that, in general, $\rho^s(\mathbf{Z})$ is not equal to $\rho^{ss}(\mathbf{Z})$. It is easy to see that $\rho^{ss}(\mathbf{Z}) \leq \rho^s(\mathbf{Z})$. For any matrix \mathbf{Z} , $\rho^{ss}(\mathbf{Z}) = \rho(\mathbf{Z}_{[F,G]})$ and $\rho^s(\mathbf{Z}) = \rho(\mathbf{Z}_{[T]})$, where F, G, T are subsets of $\Gamma = \{1, 2, \dots, g\}$ of size s , chosen to maximize the corresponding singular value. Using (4.18) and (4.20), we define two sparse block coherence measures:

$$\mu_B^{ss} := \max \left\{ \frac{1}{g} \rho^{ss}(\mathbf{D}_{[G]}^T \mathbf{D}_{[F]}), G, F \in \mathcal{G}, G \neq F \right\}, \quad (4.21)$$

$$\mu_B^s := \max \left\{ \frac{1}{g} \rho^s(\mathbf{D}_{[G]}^T \mathbf{D}_{[F]}), G, F \in \mathcal{G}, G \neq F \right\}. \quad (4.22)$$

The choice of scaling is to ensure that $\mu_B^s, \mu_B^{ss} \leq \mu_B$.

Note that, while $\rho^s(\mathbf{Z})$ (also referred to in the literature as *sparse principal component*

analysis (SPCA)) and $\rho^{ss}(\mathbf{Z})$ are in general NP-hard to compute, in many cases they can be computed exactly, or approximated, using convex programming techniques [33, 34, 35].

The following proposition establishes some relations between these new definitions and the standard coherence measures.

Proposition 1. The sparse block-coherence measures μ_B^{ss}, μ_B^s satisfy

$$0 \leq \mu_B^{ss} \leq \frac{s}{g}\mu, \quad 0 \leq \mu_B^s \leq \sqrt{\frac{s}{g}}\mu. \quad (4.23)$$

Proof: The inequalities $\mu_B^{ss}, \mu_B^s \geq 0$ follow immediately from the definition. We obtain the upper bounds by rewriting $\rho^{ss}(\mathbf{Z})$ and $\rho^s(\mathbf{Z})$ and then using the Geršgorin Theorem,

$$\rho^{ss}(\mathbf{Z}) = \lambda_{\max}^{1/2}(\mathbf{Z}_{[F,G]}^T \mathbf{Z}_{[F,G]}) \stackrel{(a)}{\leq} \sqrt{\max_l \sum_{r=1}^s |e_{lr}|} \leq \sqrt{s \max_{l,r} |e_{lr}|}, \quad (4.24)$$

$$\rho^s(\mathbf{Z}) = \lambda_{\max}^{1/2}(\mathbf{Z}_{[T]}^T \mathbf{Z}_{[T]}) \stackrel{(b)}{\leq} \sqrt{\max_l \sum_{r=1}^s |e'_{lr}|} \leq \sqrt{s \max_{l,r} |e'_{lr}|}, \quad (4.25)$$

where e_{lr} and e'_{lr} are the elements of $\mathbf{E} = \mathbf{Z}_{[F,\Gamma]}^T \mathbf{Z}_{[F,\Gamma]}$ and $\mathbf{E}' = \mathbf{Z}^T \mathbf{Z}$, and (a), (b) are a consequence of Geršgorin's disc theorem.

The entries of $\mathbf{Z} = \mathbf{D}_{[G_i]}^T \mathbf{D}_{[G_j]}$ for $i \neq j$ have absolute value smaller than or equal to μ , and the size of \mathbf{Z} is $g \times g$. Therefore, $|e_{kl}| \leq s\mu^2$ and $|e'_{kl}| \leq g\mu^2$. Substituting these values into (4.24) and (4.25) concludes the proof of the upper bounds on μ_B^{ss} and μ_B^s . ■

4.2 Recovery Proof

Our main recovery result is stated as follows. Suppose that \mathbf{a} is a block k -sparse vector with blocks of length g , where each block has sparsity exactly s ,⁴ and let $\mathbf{x} = \mathbf{D}\mathbf{a}$. We rearrange the columns in \mathbf{D} and the coefficients in \mathbf{a} so that the first k groups, $\{G_1, G_2, \dots, G_k\}$

⁴These conditions are non-limiting, since we can always complete the vector with zeros.

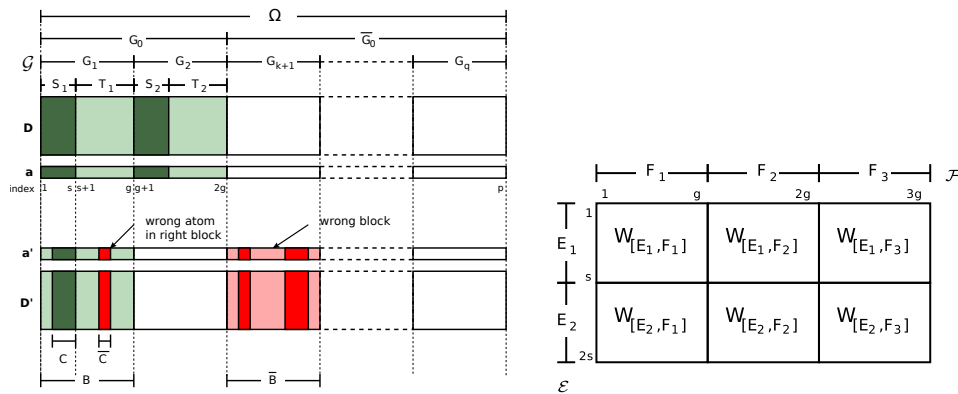


Figure B.3: Left: Indexing conventions, here shown for $g = 8$, $k = 2$ and $s = 3$. Shaded regions correspond to active elements/atoms. Active blocks are light-colored, and active elements/coefficients are dark colored. Here \mathbf{a}' represents an alternate representation of \mathbf{x} , $\mathbf{x} = \mathbf{D}\mathbf{a}'$. Blocks and atoms that are not part of the true solution \mathbf{a} are marked in red. Right: partitioning of a matrix \mathbf{W} performed by the measure $\rho_{[\mathcal{E}, \mathcal{F}]}(\mathbf{W})$ with $\mathcal{E} = \{E_1, E_2\}$ and $\mathcal{F} = \{F_1, F_2, F_3\}$, where $|E_i| = s$ and $|F_j| = g$.

correspond to the non-zero (active) blocks. Within each block G_i , $i \leq k$, the first s indices, represented by the set S_i , correspond to the s nonzero coefficients in that block, and the index set $T_i = G_i \setminus S_i$ represents its $(g - s)$ inactive elements, so that $G_i = [S_i \ T_i]$. The set $G_0 = \bigcup_{i=1}^k G_i$ contains the indices of all the active blocks of \mathbf{a} , whereas $\overline{G_0} = \Omega \setminus G_0$ contains the inactive ones. Similarly, $S_0 = \bigcup_{i=1}^k S_i$ contains the indices of all the active coefficients/atoms in \mathbf{a} and \mathbf{D} respectively, $\overline{S_0} = \Omega \setminus S_0$ indexes the inactive coefficients/atoms in \mathbf{a}/\mathbf{D} , and $T_0 = \bigcup_{i=1}^k T_i$ indexes the inactive coefficients/atoms within the active blocks. These indexing conventions are exemplified in Figure B.3(left). With these conventions we can write $\mathbf{x} = \mathbf{D}_{[G_0]} \mathbf{a}_{[G_0]} = \mathbf{D}_{[S_0]} \mathbf{a}_{[S_0]}$.

An important assumption that we will rely on throughout, is that the columns of $\mathbf{D}_{[G_0]}$ must be linearly independent for any G_0 as defined above. Under this assumption, $\mathbf{D}_{[S_0]}^\top \mathbf{D}_{[S_0]}$ is invertible and we can define the pseudo-inverse $\mathbf{H} := (\mathbf{D}_{[S_0]}^\top \mathbf{D}_{[S_0]})^{-1} \mathbf{D}_{[S_0]}^\top$. For reasons that will become clear later, we will also need a second, oblique pseudo-inverse, $\mathbf{Q} := (\mathbf{D}_{[S_0]}^\top (\mathbf{I} - \mathbf{P}) \mathbf{D}_{[S_0]})^{-1} \mathbf{D}_{[S_0]}^\top (\mathbf{I} - \mathbf{P})$, where \mathbf{P} is an orthogonal projection onto the range of

$\mathbf{D}_{[T_0]}$, that is, $\mathbf{PD}_{[T_0]} = \mathbf{D}_{[T_0]}$. It is easy to check that

$$\mathbf{QD}_{[T_0]} = \mathbf{0}, \quad \text{and} \quad \mathbf{QD}_{[S_0]} = \mathbf{I}. \quad (4.26)$$

Equipped with these definitions we can now state our main result.

Theorem 1. Let \mathbf{a} be a block k -sparse vector with blocks of length g , where each block has sparsity s . Let $\mathbf{x} = \mathbf{D}\mathbf{a}$ for a given matrix \mathbf{D} . A sufficient condition for the HiLasso algorithm (4.15) to recover \mathbf{a} from \mathbf{x} is that, for some $\alpha \leq 1$,

$$\rho_{[\mathcal{S}_0, \overline{\mathcal{G}}_0]}(\mathbf{QD}_{[\overline{\mathcal{G}}_0]}) < \alpha, \quad (4.27)$$

$$\|\mathbf{HD}_{[\overline{\mathcal{G}}_0]}\|_{1,1} < \gamma, \quad \gamma \leq 1 + \frac{\lambda(1-\alpha)}{\sqrt{g}(1-\lambda)}, \quad (4.28)$$

$$\|\mathbf{HD}_{[T_0]}\|_{1,1} < 1. \quad (4.29)$$

Here $\rho_{[\mathcal{E}, \mathcal{F}]}(\mathbf{Z}) := \max_{F \in \mathcal{F}} \sum_{E \in \mathcal{E}} \rho(\mathbf{Z}_{[E,F]})$, is the block spectral norm defined in [27], the blocks defined by the sets of index sets \mathcal{E} and \mathcal{F} (see Figure B.3(right)). We also define $\mathcal{S}_0 = \{S_i : i = 1, \dots, k\}$, $\overline{\mathcal{G}}_0 = \{G_i : i = k+1, \dots, q\}$ and $\mathcal{T}_0 = \{T_i : i = 1, \dots, k\}$. Finally, $\|\mathbf{Z}\|_{1,1} := \max_r \|\mathbf{z}_r\|_1$, where \mathbf{z}_r is the r -th column of \mathbf{Z} .

The above theorem can be interpreted as follows. With $\gamma = 1$, the conditions (4.28)–(4.29) are sufficient both for Lasso ($\lambda = 0$) and HiLasso to recover \mathbf{a} . However, if there exists a $\gamma > 1$ for which condition (4.28) holds, then HiLasso will be able to recover \mathbf{a} in a situation where Lasso is not guaranteed to do so. The idea is that, for $0 < \lambda < 1$, HiLasso trades off between the minimization of its ℓ_1 and ℓ_2 terms, by tightening the ℓ_2 term ($\alpha \leq 1$) to improve group recovery, while loosening the ℓ_1 term ($\gamma > 1$). Also, although not yet clear from conditions (4.27)–(4.29), we will see in Theorem 2 that the final data independent bounds are also a relaxation of the ones corresponding to Group Lasso when the solutions are block-dense. Therefore, the proposed model outperforms both standard Lasso and

Group Lasso with regard to recovery guarantees. This is also reflected in the experimental results presented in the next section.

The sufficient conditions (4.27)–(4.29) depend on $\mathbf{D}_{[S_0]}$ and therefore on the nonzero blocks in \mathbf{a} , G_0 , and the nonzero locations within the blocks, S_0 , which, of course, are not known in advance. Nonetheless, Theorem 2 provides sufficient conditions ensuring that (4.27)–(4.29) hold, which are independent of the unknown signals, and depend only on the dictionary \mathbf{D} .

We now prove Theorem 1.

Proof: To prove that (4.15) recovers the correct vector \mathbf{a} , let \mathbf{a}' be an alternative solution satisfying $\mathbf{x} = \mathbf{D}\mathbf{a}'$. We will show that $\lambda\psi_{\mathcal{G}}(\mathbf{a}) + (1 - \lambda)\|\mathbf{a}\|_1 < \lambda\psi_{\mathcal{G}}(\mathbf{a}') + (1 - \lambda)\|\mathbf{a}'\|_1$. Let the set G_0 contain the indices of all elements in the active blocks of \mathbf{a} . Let G'_0 contain the indices of the active blocks in \mathbf{a}' . Then $\mathbf{x} = \mathbf{D}_{[G_0]}\mathbf{a}_{[G_0]} = \mathbf{D}_{[G'_0]}\mathbf{a}'_{[G'_0]}$.

By our assumptions, in each block of $\mathbf{a}_{[G_0]}$ there are exactly s nonzero values. Let the set $S_0 \subset G_0$ contain the indices of all nonzero elements in \mathbf{a} . We thus have $|S_0| = ks$. Using (4.26) we can write

$$\mathbf{a}_{[S_0]} = \mathbf{QD}_{[S_0]}\mathbf{a}_{[S_0]} = \mathbf{QD}_{[G_0]}\mathbf{a}_{[G_0]} = \mathbf{QD}_{[G'_0]}\mathbf{a}'_{[G'_0]}. \quad (4.30)$$

To proceed, we separate G'_0 into two parts: $B = G'_0 \cap G_0$, and $\bar{B} = G'_0 \setminus G_0$, so that $G'_0 = [B \bar{B}]$ and $\mathbf{D}_{[G'_0]}\mathbf{a}'_{[G'_0]} = \mathbf{D}_{[B]}\mathbf{a}'_{[B]} + \mathbf{D}_{[\bar{B}]}\mathbf{a}'_{[\bar{B}]}$. We can now rewrite (4.30) as

$$\mathbf{a}_{[S_0]} = \mathbf{QD}_{[B]}\mathbf{a}'_{[B]} + \mathbf{QD}_{[\bar{B}]}\mathbf{a}'_{[\bar{B}]}, \quad (4.31)$$

and use the triangle inequality to obtain

$$\psi_{\mathcal{G}}(\mathbf{a}_{[S_0]}) \leq \psi_{\mathcal{G}}(\mathbf{QD}_{[B]}\mathbf{a}'_{[B]}) + \psi_{\mathcal{G}}(\mathbf{QD}_{[\bar{B}]}\mathbf{a}'_{[\bar{B}]}). \quad (4.32)$$

We now analyze the two terms in the right hand side of (4.32) using [27, Lemma 3]:

Lemma 1. Let $\mathbf{v} \in \mathbb{R}^p$ be a vector, $\mathbf{Z} \in \mathbb{R}^{m \times p}$ be a matrix, \mathcal{F} be a partition of $\Omega = \{1, 2, \dots, p\}$, and \mathcal{E} a partition of $\{1, \dots, m\}$. We then have that, $\psi_{\mathcal{G}}(\mathbf{Z}\mathbf{v}) \leq \rho_{[\mathcal{E}, \mathcal{F}]}(\mathbf{Z})\psi_{\mathcal{G}}(\mathbf{v})$.⁵

Since $\bar{B} \subset \bar{G}_0$, it follows from (4.27) that $\rho_{[\mathcal{S}_0, \bar{\mathcal{B}}]}(\mathbf{QD}_{[\bar{B}]}) < \alpha$ (here $\bar{\mathcal{B}}$ is the set of the blocks that comprise \bar{B}). To analyze $\rho_{[\mathcal{S}_0, \mathcal{B}]}(\mathbf{QD}_{[B]})$, we use its definition,

$$\rho_{[\mathcal{S}_0, \mathcal{B}]}(\mathbf{QD}_{[B]}) = \max_{F \in \mathcal{B}} \sum_{E \in \mathcal{S}_0} \rho((\mathbf{QD})_{[E, F]}) = \max_{F \in \mathcal{B}} \sum_{S_j: j=1, \dots, k} \rho((\mathbf{QD})_{[S_j, F]}), \quad (4.33)$$

and analyze each of its terms. By definition of \mathcal{B} , each $F \in \mathcal{B}$ corresponds to some $G_i = [S_i \ T_i]$ for some $i \leq k$. We can thus write $(\mathbf{QD})_{[S_j, F]} = [(\mathbf{QD})_{[S_j, S_i]} \ (\mathbf{QD})_{[S_j, T_i]}]$. Then, by recalling that $\mathbf{QD}_{[T_0]} = \mathbf{0}$ we see that $(\mathbf{QD})_{[S_j, T_i]} = \mathbf{0}$ for all i, j . Now, when $i = j$ we have $(\mathbf{QD})_{[S_j, S_i]} = \mathbf{I}$, thus $\rho((\mathbf{QD})_{[S_j, F]}) = \rho([\mathbf{I} \ \mathbf{0}]) = 1$. When $i \neq j$, $(\mathbf{QD})_{[S_j, S_i]} = \mathbf{0}$, and $\rho((\mathbf{QD})_{[S_j, F]}) = \rho([\mathbf{0} \ \mathbf{0}]) = 0$ in that case. From (4.33) we conclude that $\rho_{[\mathcal{S}_0, \mathcal{B}]}(\mathbf{QD}_{[B]}) = 1$. Plugging into (4.32) leads to

$$\psi_{\mathcal{G}}(\mathbf{a}) < \psi_{\mathcal{G}}(\mathbf{a}'_{[B]}) + \alpha \psi_{\mathcal{G}}(\mathbf{a}'_{[\bar{B}]}). \quad (4.34)$$

For the ℓ_1 term, we follow the same path as (4.30) and (4.31), now using the Moore-Penrose pseudo-inverse \mathbf{H} instead, yielding $\mathbf{a}_{[S_0]} = \mathbf{HD}_{[B]}\mathbf{a}'_{[B]} + \mathbf{HD}_{[\bar{B}]}\mathbf{a}'_{[\bar{B}]}$, from which $\|\mathbf{a}\|_1 \leq \|\mathbf{HD}_{[B]}\mathbf{a}'_{[B]}\|_1 + \|\mathbf{HD}_{[\bar{B}]}\mathbf{a}'_{[\bar{B}]}\|_1$ follows. Using the fact that $\|\mathbf{W}\mathbf{v}\|_{1,1} \leq \|\mathbf{W}\|_{1,1} \|\mathbf{v}\|_1$ [31], we get $\|\mathbf{a}\|_1 \leq \|\mathbf{HD}_{[B]}\|_{1,1} \|\mathbf{a}'_{[B]}\|_1 + \|\mathbf{HD}_{[\bar{B}]}\|_{1,1} \|\mathbf{a}'_{[\bar{B}]}\|_1$. Now, since $B \subset G_0$, and $\|\mathbf{HD}_{[G_0]}\|_{1,1} = 1$, we have that $\|\mathbf{HD}_{[B]}\|_{1,1} \leq 1$. Together with condition (4.29) this yields,

$$\|\mathbf{a}\|_1 < \|\mathbf{a}'_{[B]}\|_1 + \gamma \|\mathbf{a}'_{[\bar{B}]}\|_1. \quad (4.35)$$

⁵Note that the statement of Lemma 1 as shown here is actually a slight generalization of [27, Lemma 3], where the groups in the partitions need not have the same size.

Combining (4.34) and (4.35) into the HiLasso cost function we get

$$\lambda\psi_{\mathcal{G}}(\mathbf{a}) + (1 - \lambda)\|\mathbf{a}\|_1 < \lambda \left[\psi_{\mathcal{G}}(\mathbf{a}'_{[B]}) + \alpha\psi_{\mathcal{G}}(\mathbf{a}'_{[\bar{B}]}) \right] + (1 - \lambda) \left[\|\mathbf{a}'_{[B]}\|_1 + \gamma \|\mathbf{a}'_{[\bar{B}]}\|_1 \right]. \quad (4.36)$$

Now, to finish the proof, we need to bound the right hand side of (4.36) by $\lambda\psi_{\mathcal{G}}(\mathbf{a}') + (1 - \lambda)\|\mathbf{a}'\|_1$, in order to show that the alternate \mathbf{a}' is not a minimum of the HiLasso problem. For any γ satisfying

$$\gamma \leq 1 + \frac{\lambda(1 - \alpha)\psi_{\mathcal{G}}(\mathbf{a}'_{[\bar{B}]})}{(1 - \lambda)\|\mathbf{a}'_{[\bar{B}]}\|_1},$$

we have,

$$\lambda[\psi_{\mathcal{G}}(\mathbf{a}'_{[B]}) + \alpha\psi_{\mathcal{G}}(\mathbf{a}'_{[\bar{B}]})] + (1 - \lambda)[\|\mathbf{a}'_{[B]}\|_1 + \gamma\|\mathbf{a}'_{[\bar{B}]}\|_1] \leq \lambda\psi_{\mathcal{G}}(\mathbf{a}') + (1 - \lambda)\|\mathbf{a}'\|_1, \quad (4.37)$$

where we have used the fact that $\|\mathbf{a}'\|_1 = \|\mathbf{a}'_{[B]}\|_1 + \|\mathbf{a}'_{[\bar{B}]}\|_1$ and $\psi_{\mathcal{G}}(\mathbf{a}') = \psi_{\mathcal{G}}(\mathbf{a}'_{[B]}) + \psi_{\mathcal{G}}(\mathbf{a}'_{[\bar{B}]})$. To obtain a signal independent relationship between γ and α , we bound $\psi_{\mathcal{G}}(\mathbf{a}'_{[\bar{B}]})$ in terms of $\|\mathbf{a}'_{[\bar{B}]}\|_1$,

$$\|\mathbf{a}'_{[\bar{B}]}\|_1 = \sum_i \|\mathbf{a}'_{[\bar{R}_i]}\|_1 \leq \sum_i \sqrt{g} \|\mathbf{a}'_{[\bar{B}_i]}\|_2 = \sqrt{g} \psi_g(\mathbf{a}'_{[\bar{B}]})$$

resulting in the condition

$$\gamma \leq 1 + \frac{\lambda(1 - \alpha)}{(1 - \lambda)\sqrt{g}} \leq 1 + \frac{\lambda(1 - \alpha)\psi_{\mathcal{G}}(\mathbf{a}'_{[\bar{B}]})}{(1 - \lambda)\|\mathbf{a}'_{[\bar{B}]}\|_1},$$

which completes the proof. ■

We conclude that we can guarantee recovery for every choice of λ as long as (4.27)–(4.29) are satisfied. Note that when $\lambda = 0$ (Lasso mode) we get $\gamma \leq 1$, and, as expected,

(4.28)–(4.29) reduce to the Lasso recovery condition. Also, if $\alpha = 1$ we have $\gamma \leq 1$, meaning that we must tighten the constraints related to the ℓ_2 part of the cost function in order to relax the ℓ_1 part. For $\gamma > 1$, the HiLasso conditions are a relaxation of the Lasso conditions, thus allowing for more signals to be correctly recovered.

Theorem 2 below provides signal independent replacements of the conditions (4.27)–(4.29). The signal independent bound for (4.27) derived here, depends on coherence measures between the dictionary \mathbf{D} and its image under the projection $\mathbf{I} - \mathbf{P}$, $\mathbf{C} = (\mathbf{I} - \mathbf{P})\mathbf{D}$. Since \mathbf{P} depends on S_0 , \mathbf{C} itself is signal dependent. Thus, we need to maximize also over all possible sets S_0 . These are defined as follows,

$$\nu_p := \max \left\{ \max \left\{ \max \left\{ \frac{\mathbf{d}_i^T \mathbf{c}_j}{(\mathbf{d}_i^T \mathbf{c}_i)^{1/2} (\mathbf{d}_j^T \mathbf{c}_j)^{1/2}}, i, j \in G, i \neq j \right\}, G \in \mathcal{G} \right\}, S_0 \right\}, \quad (4.38)$$

$$\mu_p^s := \max \left\{ \max \left\{ \frac{1}{g} \rho^s(\mathbf{D}_{[G]}^T \mathbf{C}_{[F]}), G, F \in \mathcal{G}, G \neq F \right\}, S_0 \right\}, \quad (4.39)$$

$$\mu_p^{ss} := \max \left\{ \max \left\{ \frac{1}{g} \rho^{ss}(\mathbf{D}_{[G]}^T \mathbf{C}_{[F]}), G, F \in \mathcal{G}, G \neq F \right\}, S_0 \right\}, \quad (4.40)$$

$$\zeta := \max \left\{ \max \{ (\mathbf{d}_i^T \mathbf{c}_i)^{-1/2} : i = 1, \dots, p \}, S_0 \right\}. \quad (4.41)$$

We are now in position to state the theorem.

Theorem 2. Let χ , ν_p , μ_p^s , μ_p^{ss} and ζ be the coherence measures defined respectively in (4.16) and (4.38)–(4.41). Then the conditions (4.27)–(4.29) are satisfied if

$$\frac{\zeta^2 k g \mu_p^s}{1 - (s-1)\nu_p + g \mu_p^{ss} (k-1)\zeta^2} \leq \alpha, \quad (4.42)$$

$$\frac{k s \chi}{1 - (s-1)\nu - (k-1)s\chi} < \gamma, \quad (4.43)$$

$$\frac{k s \nu}{1 - (s-1)\nu - (k-1)s\chi} < 1. \quad (4.44)$$

We also require the denominators in (4.42)–(4.44) to be positive. Note that, although the

interpretation of (4.42) is rather counter-intuitive, it is easy to check that $\mu_p^s, \mu_p^{ss} \leq \mu_B$. This can be seen when $s = g$ (a case included in our theorems), in which case $\mathbf{P} = \mathbf{0}$, $\mathbf{C} = \mathbf{D}$, and $\mu_p^s = \mu_p^{ss} = \mu_B$. Therefore, the condition (4.42) is a relaxation of the standard (dense) block-sparse recovery one [27, Theorem 2].

Proof: Recall that $\mathbf{QD}_{[\overline{G_0}]} = \left(\mathbf{D}_{[S_0]}^\top \mathbf{C}_{[S_0]} \right)^{-1} \mathbf{D}_{[S_0]}^\top \mathbf{C}_{[\overline{G_0}]}$. Since $\rho_{[\cdot, \cdot]}(\cdot)$ is submultiplicative, [27],⁶

$$\rho_{[\mathcal{S}_0, \overline{\mathcal{G}_0}]}(\mathbf{QD}_{[\overline{G_0}]}) \leq \rho_{[\mathcal{S}_0, \mathcal{S}_0]}((\mathbf{D}_{[S_0]}^\top \mathbf{C}_{[S_0]})^{-1}) \rho_{[\mathcal{S}_0, \overline{\mathcal{G}_0}]}(\mathbf{D}_{[S_0]}^\top \mathbf{C}_{[\overline{G_0}]}). \quad (4.45)$$

Applying the definitions of $\rho_{[\mathcal{S}_0, \overline{\mathcal{G}_0}]}$ and μ_p^s we have,

$$\rho_{[\mathcal{S}_0, \overline{\mathcal{G}_0}]}(\mathbf{D}_{[S_0]}^\top \mathbf{C}_{[\overline{G_0}]}) = \max_{F \in \overline{\mathcal{G}_0}} \sum_{E \in \mathcal{S}_0} \rho(\mathbf{D}_{[E]}^\top \mathbf{C}_{[F]}) \leq k \max_{F \in \overline{\mathcal{G}_0}} \max_{E \in \mathcal{S}_0} \{\rho(\mathbf{D}_{[E]}^\top \mathbf{C}_{[F]})\} \leq kg \mu_p^s, \quad (4.46)$$

where the last inequality in (4.46) derives from (4.39) and the fact that each $E \in \mathcal{S}_0$ belongs to some G_i , and $|E| = s$, thus playing the role of the set T in the definition of $\rho^s(\cdot)$. Our goal is now to obtain a bound for $\rho_{[\mathcal{S}_0, \mathcal{S}_0]}((\mathbf{D}_{[S_0]}^\top \mathbf{C}_{[S_0]})^{-1})$. To this end, we define $\mathbf{Z} = \mathbf{D}_{[S_0]}^\top \mathbf{C}_{[S_0]}$, and rewrite it as $\mathbf{Z} = \Lambda^{-1}(\mathbf{I} - (\mathbf{I} - \Lambda \mathbf{Z} \Lambda))\Lambda^{-1}$. Here Λ is a $ks \times ks$ block-diagonal scaling matrix to be defined later. Assume for now that $\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{I} - \Lambda \mathbf{Z} \Lambda) < 1$. This allows us to apply the following result from [27]:

Lemma 2. Suppose that $\rho_{[\mathcal{E}, \mathcal{F}]}(\mathbf{W}) < 1$. Then $(\mathbf{I} + \mathbf{W})^{-1} = \sum_{k=0}^{\infty} (-\mathbf{W})^k$.

By applying Lemma 2 to $\mathbf{W} = -\mathbf{I} + \Lambda \mathbf{Z} \Lambda$ we can write $\mathbf{Z}^{-1} = \Lambda \left[\sum_{i=0}^{\infty} (\mathbf{I} - \Lambda \mathbf{Z} \Lambda)^i \right] \Lambda$.

⁶There is a slight abuse of notation here, in that, in our case of non-square blocks, each norm $\rho_{[\cdot, \cdot]}(\cdot)$ in the right hand of the submultiplicativity inequality (4.45) is actually a different norm. However, it is easy to see that the referred inequality holds in this case as well.

With this,

$$\begin{aligned}
\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{Z}^{-1}) &\stackrel{(a)}{\leq} [\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\Lambda)]^2 \rho_{[\mathcal{S}_0, \mathcal{S}_0]} \left(\sum_{i=0}^{\infty} (\mathbf{I} - \Lambda \mathbf{Z} \Lambda)^i \right) \stackrel{(b)}{\leq} [\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\Lambda)]^2 \sum_{i=0}^{\infty} \rho_{[\mathcal{S}_0, \mathcal{S}_0]} \left((\mathbf{I} - \Lambda \mathbf{Z} \Lambda)^i \right) \\
&\stackrel{(c)}{\leq} [\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\Lambda)]^2 \sum_{i=0}^{\infty} \left(\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{I} - \Lambda \mathbf{Z} \Lambda) \right)^i \stackrel{(d)}{\leq} \frac{[\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\Lambda)]^2}{1 - \rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{I} - \Lambda \mathbf{Z} \Lambda)},
\end{aligned} \tag{4.47}$$

where in (a) and (c) we applied the submultiplicativity of $\rho_{[\cdot, \cdot]}(\cdot)$, (b) is a consequence of the triangle inequality, and (d) is the limit of the geometric series, which is finite when $\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{I} - \Lambda \mathbf{Z} \Lambda) < 1$.

We now bound $\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{I} - \Lambda \mathbf{Z} \Lambda)$. First, note that, since Λ is block-diagonal, we have that $(\mathbf{I} - \Lambda \mathbf{Z} \Lambda)_{[S_i, S_j]} = \mathbf{I}_{[S_i, S_j]} - \Lambda_{[S_i, S_i]} \mathbf{Z}_{[S_i, S_j]} \Lambda_{[S_j, S_j]}$. We then choose Λ to be a diagonal matrix with $\Lambda_{ii} = (\mathbf{d}_i^\top \mathbf{c}_i)^{-1/2}$, $i \in S_0$. With this choice, we have that the diagonal elements of $\mathbf{I}_{[S_j, S_j]} - \Lambda_{[S_j, S_j]} \mathbf{Z}_{[S_j, S_j]} \Lambda_{[S_j, S_j]}$ are equal to 1 for all j , and the off-diagonal elements are bounded by ν_p . Using Geršgorin Theorem we then have that

$$\rho(\mathbf{I}_{[S_j, S_j]} - \Lambda_{[S_j, S_j]} \mathbf{Z}_{[S_j, S_j]} \Lambda_{[S_j, S_j]}) \leq (s-1)\nu_p, \quad j = 1, \dots, k. \tag{4.48}$$

As for the off-diagonal $s \times s$ blocks of $\mathbf{I} - \Lambda \mathbf{Z} \Lambda$, we have $(\mathbf{I} - \Lambda \mathbf{Z} \Lambda)_{[S_i, S_j]} = -\Lambda_{[S_i, S_i]} \mathbf{Z}_{[S_i, S_j]} \Lambda_{[S_j, S_j]}$.

We then have

$$\rho((\mathbf{I} - \Lambda \mathbf{Z} \Lambda)_{[S_i, S_j]}) \stackrel{(a)}{\leq} \rho(\Lambda_{[S_i, S_i]}) \rho(\mathbf{Z}_{[S_i, S_j]}) \rho(\Lambda_{[S_j, S_j]}) \stackrel{(b)}{\leq} \zeta (g \mu_p^{ss}) \zeta, \tag{4.49}$$

where in (a) we used the submultiplicativity of $\rho(\cdot)$, and (b) derives from the definition of μ_p^{ss} , and the fact that, with our choice of Λ we have $\rho(\Lambda_{[S_i, S_i]}) \leq \zeta$ for all i . Now we can

write the definition of $\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{I} - \Lambda \mathbf{Z} \Lambda)$ and bound its summation using (4.48)–(4.49),

$$\begin{aligned} \rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{I} - \Lambda \mathbf{Z} \Lambda) &\leq \max_{S_j: j \leq k} \left\{ \rho(\mathbf{I}_{[S_j, S_j]} - \Lambda_{[S_j, S_j]} \mathbf{Z}_{[S_j, S_j]} \Lambda_{[S_j, S_j]}) + \dots \right. \\ &\quad \left. \dots \sum_{S_i: i \leq k, i \neq j} \rho\left(\Lambda_{[S_i, S_i]}(\mathbf{I} - \Lambda \mathbf{Z} \Lambda)_{[S_i, S_i]} \Lambda_{[S_i, S_i]}\right) \right\} \leq (s-1)\nu_P + g\mu_P^{ss} \zeta^2. \end{aligned} \quad (4.50)$$

By our choice of Λ , $\rho(\Lambda_{[S_i, S_i]}) \leq \zeta$ and $\rho(\Lambda_{[S_i, S_j]}) = 0$ for $i \neq j$. Therefore $\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\Lambda) \leq \zeta$ as well. Using this together with (4.50) in (4.47), we obtain

$$\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{Z}^{-1}) \leq \frac{\zeta^2}{1 - (s-1)\nu_P + g\mu_P^{ss}(k-1)\zeta^2}. \quad (4.51)$$

To ensure that $\rho_{[\mathcal{S}_0, \mathcal{S}_0]}(\mathbf{I} - \Lambda \mathbf{Z} \Lambda) < 1$, we need the denominator in the above equation to be positive. Now (4.42) follows by plugging (4.46) and (4.51) into (4.45),

$$\rho_{[\mathcal{S}_0, \overline{\mathcal{G}}_0]}(\mathbf{QD}_{[\overline{\mathcal{G}}_0]}) \leq \frac{\zeta^2 k g \mu_P^s}{1 - (s-1)\nu_P + g\mu_P^{ss}(k-1)\zeta^2}.$$

Finally, we use the same ideas to bound $\|\mathbf{HD}_{[\overline{\mathcal{G}}_0]}\|_{1,1}$ and derive (4.43). Specifically,

$$\|\mathbf{HD}_{[\overline{\mathcal{G}}_0]}\|_{1,1} \leq \|(\mathbf{D}_{[S_0]}^\top \mathbf{D}_{[S_0]})^{-1}\|_{1,1} \|\mathbf{D}_{[S_0]}^\top \mathbf{D}_{[\overline{\mathcal{G}}_0]}\|_{1,1}. \quad (4.52)$$

Now

$$\|(\mathbf{D}_{[S_0]}^\top)^\top \mathbf{D}_{[\overline{\mathcal{G}}_0]}\|_{1,1} = \max_{j \in \overline{\mathcal{G}}_0} \sum_{i \in S_0} |\mathbf{d}_i^\top \mathbf{d}_j| \stackrel{(a)}{\leq} ks\chi, \quad (4.53)$$

where (a) follows from the definition of χ and the fact that $|S_0| = ks$. It remains to develop

a bound on $\|(\mathbf{D}_{[S_0]}^\top \mathbf{D}_{[S_0]})^{-1}\|_{1,1}$. To this end we express $\mathbf{D}_{[S_0]}^\top \mathbf{D}_{[S_0]} = \mathbf{I} + \mathbf{W}$, and bound

$$\|\mathbf{W}\|_{1,1} = \max_{r \leq k} \left\{ \max_{i \in S_r} \left\{ \sum_{j \in S_r, j \neq i} |\mathbf{d}_i^\top \mathbf{d}_j| + \sum_{j \in S_0 \setminus S_r} |\mathbf{d}_i^\top \mathbf{d}_j| \right\} \right\} \leq (s-1)\nu + s(k-1)\chi. \quad (4.54)$$

since for all $S_r, r \leq k$, and all $i \in S_r$, the first sum has $(s-1)$ nonzero elements bounded by ν , and the second sum has $s(k-1)$ elements bounded by χ . Now, by requiring $(s-1)\nu + s(k-1)\chi < 1$ we can apply Lemma 2 to \mathbf{W} and follow the same path as the one that leads to (4.50), now using the matrix norm properties of $\|\cdot\|_{1,1}$, to obtain,

$$\|(\mathbf{D}_{[S_0]}^\top \mathbf{D}_{[S_0]})^{-1}\|_{1,1} \leq \frac{1}{1 - (s-1)\nu + s(k-1)\chi}. \quad (4.55)$$

Again, $(s-1)\nu + s(k-1)\chi < 1$ is implicit in the requirement that the above denominator be positive. Plugging (4.55) and (4.53) into (4.52) yields (4.43).

The proof for (4.44) is analogous to that of (4.43), only that now the upper bound on $|\mathbf{d}_i^\top \mathbf{d}_j|, i \in S_0, j \in T_0$, is $\nu \leq \mu$. Continuing as before leads to (4.44). \blacksquare

Theorems 1 and 2 are for the non-collaborative case. For the collaborative case there exist results that show that both the C-Lasso [9] and C-GLasso [26] will recover the true shared active set with a probability of error that vanishes exponentially with n . Since the in-group active sets are not necessarily equal for all samples in \mathbf{X} , C-HiLasso could only help in recovering the group sparsity pattern. Since the C-GLasso is a special case of C-HiLasso when $\lambda_1 = 0$, we can conjecture that when $\lambda_1 > 0$, the accuracy of the C-HiLasso in recovering the correct groups will improve with larger n . Furthermore, since our results for HiLasso improve on those of the Group Lasso, it is to be expected that the accuracy of C-HiLasso, for an appropriate $\lambda_1 > 0$, will be better than that of C-GLasso.

As an intuitive explanation to why this may happen, the proofs in [9] and [26] assume

a continuous probability distribution on the non-zero coefficients of the signals, and give recovery results for the average case. On the other hand, the in-group sparsity assumption of C-HiLasso implies that only s out of g samples will be nonzero within each group. This implies that, for the same group sparsity pattern, there will be much less (exactly a fraction s/g) non-zero elements in the possible signals compared to the ones that can occur under the hypothesis of C-GLasso. Since any assumed distribution of the signals under the in-group sparsity hypothesis has to be concentrated on this much smaller set of possible signals, they should be easier to recover correctly from solutions to the C-HiLasso program, compared to the dense group case of C-GLasso.

5 Experimental results

In this section we show the strength of the proposed HiLasso and C-HiLasso models. We start by comparing our model with the standard Lasso and Group Lasso using synthetic data. We created q dictionaries, $\mathbf{D}_r, r = 1, \dots, q$, with $g = 64$ atoms of dimension $m = 64$, and i.i.d. Gaussian entries. The columns were normalized to have unit ℓ_2 norm. We then randomly chose $k = 2$ groups to be active at each time (on all the signals). Sets of $n = 200$ normalized testing signals were generated, one per active group, as linear combinations of $s \ll 64$ elements of the active dictionaries, $\mathbf{x}_j^r = \mathbf{D}_r \mathbf{a}_j^r$. The mixtures were created by summing these signals and (eventually) adding Gaussian noise of standard deviation σ . The generated testing signals have a hierarchical sparsity structure and while they share groups, they do not necessarily share the sparsity pattern inside the groups. We then built a single dictionary by concatenating the sub-dictionaries, $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_q]$, and used it to solve the Lasso, Group Lasso, HiLasso and C-HiLasso problems. Table B.1 summarizes the Mean Squared Error (MSE) and Hamming distance of the recovered coefficient vectors $\mathbf{a}_j, j = 1, \dots, n$. We observe that our model is able to exploit the hierarchical structure of

$\sigma = 0.1$	417 / 22.0	1173 / 361.6	$s = 8$	388 / 22.0	1184 / 318.2
	330 / 19.8	163 / 13.3		272 / 19.5	96 / 16.2
$\sigma = 0.2$	564 / 21.6	1182 / 378.3	$s = 12$	1200 / 36.2	1166 / 350.4
	399 / 22.7	249 / 17.1		704 / 26.5	413 / 29.1
$\sigma = 0.4$	965 / 22.7	1378 / 340.3	$s = 16$	1641 / 43.9	1093 / 338.6
	656 / 19.5	595 / 27.4		1100 / 32.2	551 / 35.0
$q = 4$	1080 / 27.8	1916 / 221.7	$q = 8$	1200 / 36.2	1166 / 350.4
	1009 / 29.8	742 / 30.2		704 / 26.5	413 / 29.1
$q = 12$	1030 / 41.8	840 / 447.7	$q = 16$	1030 / 41.8	840 / 447.7
	662 / 26.4	4 / 29.8		662 / 26.4	4 / 29.8

Table B.1: Simulated signal results. In every table, each 2×2 cell contains the MSE ($\times 10^4$) and Hamming distance (MSE/Hamming) for Lasso (top,left), GLasso (top,right), HiLasso (bottom,left) and C-HiLasso (bottom,right). In the first case (left) we vary the noise σ while keeping $q = 8$ and $s = 8$ fixed. In the second and third cases we have $\sigma = 0$. For the second experiment (center) we fixed $q = 8$ while changing s . In the third case we fix $s = 12$ and vary the number of groups q . Bold blue indicates the best results, always obtained for the proposed models. In all cases, the number of active groups is $k = 2$.

the data as well as the collaborative structure. Group Lasso selects in general the correct blocks but it does not give a sparse solution within them. On the other hand, Lasso gives a solution that has nonzero elements belonging to groups that were not active in the original signal, leading to a wrong model/class selection. HiLasso gives a sparse solution that picks atoms from the correct groups but still presents some minor mistakes. For the collaborative case, in all the tested configurations, no coefficients were selected outside the correct active groups, and the recovered coefficients are consistently the best ones.

In all the examples, and for each method, the regularization parameters were the ones for which the best results were obtained. One can scale the parameter λ_2 to account for different number of signals. This situation is analogous to a change in the size of the dictionary, thus, λ_2 should be proportional to the square root of the number of signals to code.

We then experimented with the USPS digits dataset, which has been shown to be well

experiment	Lasso		GLasso		HiLasso		C-GLasso		C-HiLasso	
	AMSE	Hamm	AMSE	Hamm	AMSE	Hamm	AMSE	Hamm	AMSE	Hamm
1 digit	0.06	0.43	0.07	0.78	0.02	0.19	0.01	0.02	0.02	0.06
1 digit+n	0.08	1.31	0.08	0.87	0.04	0.48	0.05	0.25	0.02	0.01
2 digit	0.09	1.46	0.08	1.86	0.02	1.18	0.01	0.74	0.02	0.90
2 digit+n	0.11	2.21	0.08	1.99	0.04	1.46	0.09	1.60	0.03	0.70

Table B.2: Noisy digit mixtures results. Four different cases are shown: when each signal is a single digit and when it is the mixture of two different (randomly selected) digits, with and without additive Gaussian noise with standard deviation 10% of the peak value. For the 2 digits case, results are the average of 8 runs (in each round a new pair of digits was randomly selected). In the single digit case, the result is the average of the ten possible situations. Both AMSE and Hamming distance are shown, with bold blue indicating best. Without noise, both C-GLasso and C-HiLasso yield very good results. However, in the noisy case, C-HiLasso is clearly superior, showing the advantage of adding regularization inside the groups from a robustness perspective. See also Figure B.4.

represented in the sparse modeling framework [36]. Here the signals are vectors containing the unwrapped gray intensities of 16×16 images ($m = 256$). We obtained each of the $n = 200$ samples in the testing data set as the mixture of two randomly chosen digits, one from each of the two drawn sets of digits. In this case we only have ground truth at the group level. We measure the recovery performance in terms of the average MSE of the recovered signals, $\text{AMSE} = \frac{1}{nq} \sum_{r=1}^q \sum_{j=1}^n \left\| \mathbf{x}_j^r - \hat{\mathbf{x}}_j^r \right\|_2^2$, where \mathbf{x}_j^r is the component corresponding to source r in the signal j , and $\hat{\mathbf{x}}_j^r$ is the recovered one.

Using the usual training-testing split for USPS, we first learned a dictionary for each digit. We then created a single dictionary by concatenating them. In Table B.2 we show the AMSE obtained while summing $k = 2$ different digits. We also consider the situation where only one digit is present. C-HiLasso automatically detects the number of sources while achieving the best recovery performance. As in the synthetic case, only the collaborative method was able to successfully detect the true active classes. In Figure B.4 we relax the assumption that all the signals have to contain exactly the same type and amount of classes in the mixture, further demonstrating the flexibility of the proposed C-HiLasso model.

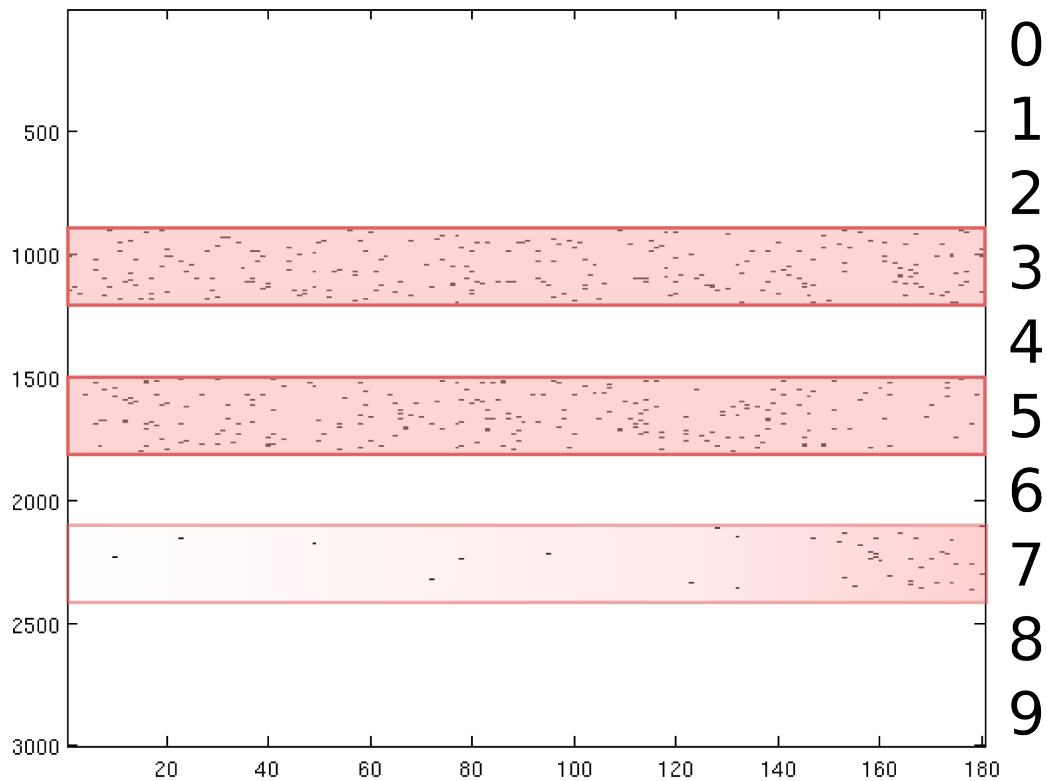


Figure B.4: In this example we used C-HiLasso to analyze mixtures where the data set contains different number and types of sources/classes. We used a set containing 180 mixtures of digit images. The first 150 images are obtained as the sum/mixture of a number “3” and an number “5” (randomly selected). Each of the last 30 images in the set are the mixture of three numbers: “3”, “5” and “7” (the 180 images are of course presented at random, the algorithm is not a priori aware which images contain 2 sources and which contain 3). The figure shows the active sets of the recovered coefficients matrix A as a binary matrix the same size as A (atom indices in the vertical and sample indices in the horizontal), where black dots indicate nonzero coefficients. C-HiLasso managed to identify the active blocks while the sub-dictionary corresponding to “7” is mostly active for the last 30 images. The accuracy of this result depends on the relationship between the sub-dictionaries corresponding to each digit.

We also used the digits dataset to experiment with missing data. We randomly discarded an average of 60% of the pixels per mixed image and then applied C-Hilasso. The algorithm is capable of correctly detecting which digits are present in the images. Some example results for this case are shown in Figure B.5. Note that this is a quite different problem than the one commonly addressed in the matrix completion literature. Here we do not aim to recover signals that all belong to a unique unknown subspace, but signals that are the combination of two non-unique spaces to be automatically identified from the available dictionary. Such unknown spaces have common models/groups for all the signals in question (the coarse level of the hierarchy), but not necessarily the exact same atoms inside the groups and therefore do not necessarily belong to the same subspaces. Both levels of the hierarchy are automatically detected, e.g., the groups corresponding to “3” and “5,” and the corresponding reconstructing atoms (subspaces) in each group, these last ones possibly different for each signal in the set. While we consider that the possible subspaces are to be selected from the provided dictionary (learned off-line from training data), in Section 6 we discuss learning such dictionaries as part of the optimization as well (see also [37, 38]). In such cases, the standard matrix completion problem becomes a particular case of the C-HiLasso framework (with a single group and all the signals having the same active set, subspace, in the group), naturally opening numerous theoretical questions for this new more general model.⁷

We also compared the performance of C-HiLasso, Lasso, GLasso and C-GLasso (without hierarchy) in the task of separating mixed textures in an image. In this case, the set of signals \mathbf{X} corresponds to all 12×12 patches in the (single) image to be analyzed. We chose 8 textures from the Brodatz dataset and trained one dictionary for each one of them using one half of the respective images (these form the $g = 8$ groups of the dictionary). Then we

⁷Prof. Carin and collaborators have new results on the case of a single group and signals in possible different subspaces of the group, an intermediate model between standard matrix completion and C-HiLasso (personal communication).

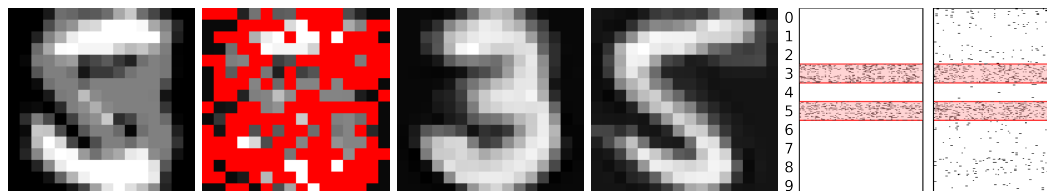


Figure B.5: Example of recovered digits (3 and 5) from a mixture with 60% of missing components. From left to right: noiseless mixture, observed mixture with missing pixels highlighted in red, recovered digits 3 and 5, and active set recovered for all samples using the C-HiLasso and Lasso respectively. In the last two figures, the active sets are represented as in Figure B.4. The coefficients blocks for digits 3 and 5 are marked as pink bands. Notice that the C-HiLasso exploits efficiently the hypothesis of collaborative group-sparsity, succeeding in recovering the correct active groups in all the samples. The Lasso, which lacks this prior knowledge, is clearly not capable of doing so, and active sets are spread over all the groups.

created an image as the sum of the other halves of the $k = 2$ textures. One can think of this experiment as a generalization to the texture separation problem proposed in [39] (without additive noise), where only two textures are present. The experiment was repeated for all possible combinations of two textures from the 8 possible ones. The results are summarized in Table B.3. A detailed example is shown in Figure B.6. For each algorithm, the best parameters were chosen using grid search, ensuring that those were not in the edges of the grid. For Lasso and C-HiLasso the best λ_1 is 0.0625. For GLasso and C-GLasso, the best λ_2 was, respectively, 0.05 and 75 (for the collaborative setting, we heuristically scale λ_2 with the number of signals as \sqrt{n} . In this experiment, $n \approx 512^2$, leading to such large value of λ_2). From Table B.3 we can conclude that the C-HiLasso is significantly better than the competing algorithms, both in the MSE of the recovered signals (we show the AMSE of recovering both active signals), and in the average Hamming distance between the recovered group-wise active sets and the true ones. In the latter case we observe that, in many cases, the C-HiLasso active set recovery performance is perfect (Hamming distance 0) or near perfect, whereas the other methods seldom approach a Hamming distance lower than 1.


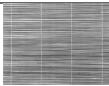

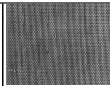
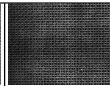
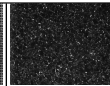


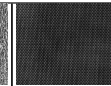
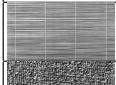

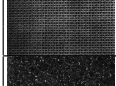
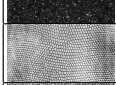

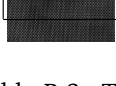
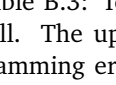
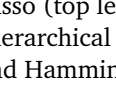
								
		110 214 117 69	18 074 069 18	63 78 126 38	19 47 47 18	85 174 132 51	107 447 102 42	7 43 27 3
			107 76 182 68	141 129 209 102	91 83 100 78	191 234 257 141	240 219 245 178	68 105 95 19
	2.80 0.42 1.36 0.00			52 42 158 43	35 62 83 29	105 112 214 62	162 141 200 107	21 93 102 10
	0.33 0.25 2.06 0.00	3.65 0.00 2.67 0.02		52 42 158 43	35 62 83 29	105 112 214 62	162 141 200 107	21 93 102 10
	0.96 0.01 1.97 0.00	3.69 0.07 2.30 0.00	1.74 0.00 2.42 0.00		49 72 81 55	123 145 224 98	182 148 214 107	26 89 85 10
	1.02 1.00 2.25 0.09	3.55 1.00 2.52 0.94	1.42 1.00 3.39 0.16	2.25 1.00 2.85 0.35		85 76 120 59	120 87 107 71	15 63 41 9
	2.26 0.32 2.50 0.00	4.12 0.53 3.23 0.82	3.48 0.44 3.54 0.20	3.49 0.32 3.11 0.01	3.16 1.00 4.07 0.40		229 240 245 162	56 95 117 27
	4.37 1.39 2.51 0.02	4.47 0.08 2.39 0.22	4.09 0.13 2.42 0.02	4.23 0.12 2.76 0.02	4.20 1.00 2.24 0.20	4.42 0.42 2.96 0.11		100 112 102 51
	0.09 0.98 0.53 0.00	3.77 1.00 1.75 0.01	0.31 1.00 2.04 0.00	1.83 1.00 1.82 0.00	1.13 1.00 2.18 0.00	3.14 0.97 3.04 0.24	4.30 1.00 1.90 0.18	

Table B.3: Texture separation results. The rows and columns indicate the active textures in each cell. The upper triangle contains the AMSE ($\times 10^4$) results, while the lower triangle shows the Hamming error in the group-wise active set recovery. Within each cell, results are shown for the Lasso (top left), Group Lasso (bottom left), Collaborative Group Lasso (top right) and Collaborative Hierarchical Lasso (bottom right). The best results are in blue bold. Note that, both for the AMSE and Hamming distance, in 26 out of 28 cases, our model outperforms previous ones.

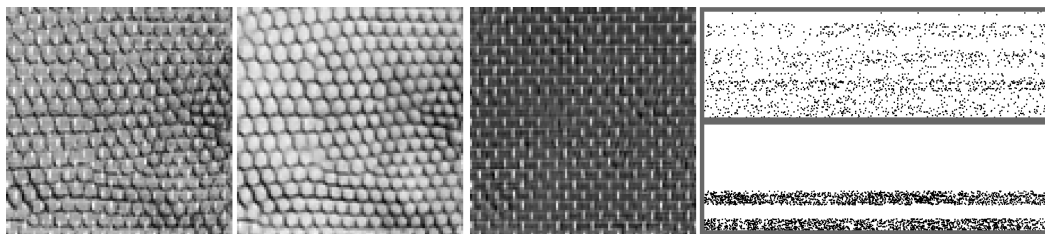


Figure B.6: Texture separation results. Left to right: sample mixture, corresponding C-HiLasso separated textures, and comparison of the active set diagrams obtained by the Lasso (as in Figure B.5). The one for Lasso is shown on top, where all groups are wrongly active, and the one for C-HiLasso on bottom, showing that only the two correct groups are selected.

Finally, we use C-HiLasso to automatically identify the sources present in a mixture of audio signals [40]. The goal is to identify the speakers talking simultaneously on a single recording. Here the task is not to fully reconstruct each of the unmixed sources from the observed signal but to identify which speakers are active. In this case, since the original sources do not need to be recovered, the modeling can be done in terms of features extracted from the original signals in a linear but non-bijective way.

Audio signals have in general very rich structures and their properties rapidly change over time. A natural approach is to decompose them into a set of overlapping local time-windows, where the properties of the signal remain stable. There is a straightforward analogy with the approach explained above for the texture segmentation case, where images were decomposed into collections of overlapping patches. These time-windows will collaborate in the identification.

A challenging aspect when identifying audio sources is to obtain features that are specific to each source and at the same time invariant to changes in the fundamental frequency (pitch) of the sources. In the case of speech, a common choice is to use the short-term power spectrum envelopes as feature vectors [41] (refer to [40] for details on the feature extraction process and implementation). The spectral envelope in human speech varies along time, producing different patterns for each phoneme. Thus, a speaker does not produce

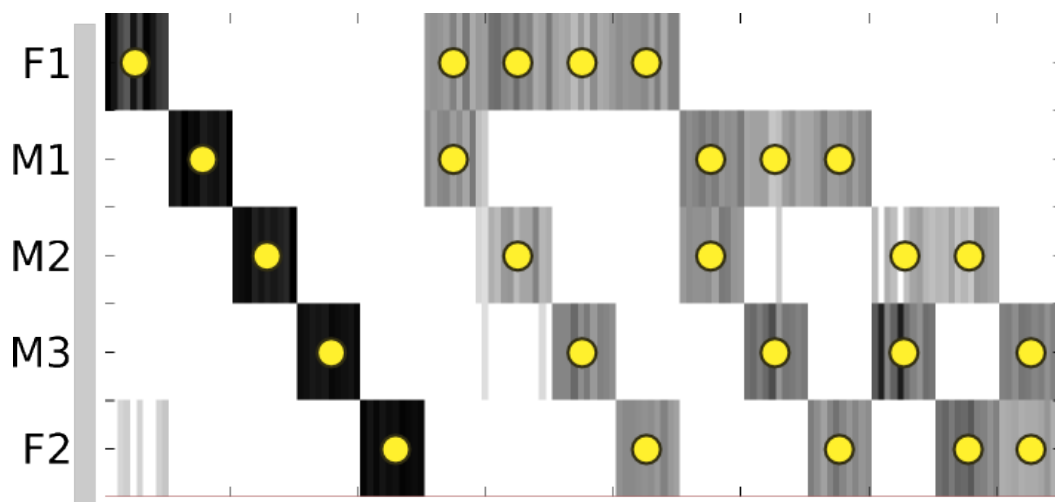


Figure B.7: Speaker identification results. Each column corresponds to the sources identified for a specific time frame, the true ones marked by yellow dots. The vertical axis indicates the estimated activity of the different sources, where darker colors indicate higher energy. For each possible combination of speakers, 10 frames (15 seconds of audio) were evaluated.

an unique spectral envelope, but a set of spectral envelopes that live in a union of manifolds. Since such manifolds are well represented by sparse models, the problem of speaker identification is well suited for the proposed C-HiLasso framework, where each block in the dictionary is trained for the features corresponding to a given speaker, and the overlapping time-windows collaborate in detecting the active blocks.

For this experiment we use a dataset consisting of recordings of five different German radio speakers, two female and three male. Each recording is six minutes long. One quarter of the samples were used for dictionary training, and the rest for testing. For each speaker, we learned a sub-dictionary from the training dataset. For testing, we extracted 10 non-overlapping frames of 15 seconds each (including silences made by the speakers while talking), and encoded them using C-HiLasso. The experiment was repeated for all possible combinations of two speakers, and all the speakers talking alone. The results are presented

in Figure B.7. C-HiLasso manages to detect automatically the number of sources very accurately, as well as the actual active speakers. Again, refer to [40] for comparisons with other sparse modeling methods (showing the clear advantage of C-HiLasso) and results obtained for the identification of wind instruments in musical recordings.

6 Discussion

We introduced a new framework of collaborative hierarchical sparse coding, where multiple signals collaborate in their encoding, sharing code groups (models) and having (possible disjoint) sparse representations inside the corresponding groups. An efficient optimization approach was developed, which guarantees convergence to the global minimum, and examples illustrating the power of this framework were presented. At the practical level, we are currently continuing our work on the applications of this proposed framework in a number of directions, including collaborative instruments separation in music, signal classification, and speaker recognition, following the here demonstrated capability to collectively select the correct groups/models.

At the theoretical level, a whole family of new problems is opened by this proposed framework, some of which we already addressed in this work. A critical one is the overall capability of selecting the correct groups in the collaborative scenario, with missing information, and thereby of performing correct model selection and source identification and separation. Results in this direction will be reported in the future.

Finally, we have also developed an initial framework for learning the dictionary for collaborative hierarchical sparse coding, meaning the optimization is simultaneously on the dictionary and the code. As it is the case with standard dictionary learning, this is expected to lead to significant performance improvements (see [36] for the particular case of this with a single group active at a time).

Acknowledgments: Work partially supported by NSF, NSSEFF, ONR, NGA, and ARO. We thank Dr.

Tristan Nguyen, when we presented him this model, he motivated us to think in a hierarchical fashion and to look at this as just the particular case of a fully hierarchical sparse coding framework. We thank Prof. Tom Luo and Gonzalo Mateos for invaluable help on optimization methods. We thank Prof. Larry Carin, Dr. Guoshen Yu and Alexey Castrodad for very stimulating conversations, and for the fact that their own work (for LC, GY, and AC) also motivated in part the example with missing information. The anonymous reviewers prompted an early mistake in the proof of Theorem 1, and that, together with their additional comments, led to improving the bounds in the theorem, as well as the overall presentation of the paper. We also want to thank the reviewer for the closed form inner loop of the proposed optimization method, which simplified it and resulted in significant practical improvements.

References

- [1] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Royal Stat. Society, Series B*, 68:49–67, 2006.
- [2] R. Jenatton, J. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. arXiv:0904.3523v1, 2009.
- [3] Y. C. Eldar and M. Mishali. Robust recovery of signals from a structured union of subspaces. *IEEE Trans. Inform. Theory*, 55(11):5302–5316, Nov. 2009.
- [4] J. Tropp. Algorithms for simultaneous sparse approximation. part II: Convex relaxation. *Signal Processing*, 86(3):589–602, 2006.
- [5] J. Tropp, A. Gilbert, and M. Strauss. Algorithms for simultaneous sparse approximation. part I: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.
- [6] S. Cotter, B. Rao, K. Engan, and K. Kreutz-Delgado. Sparse solutions to linear inverse problems with multiple measurement vectors. *IEEE Trans. Sig. Proc.*, 53(7):2477–2488, July 2005.
- [7] J. Chen and X. Huo. Theoretical results on sparse representations of multiple-measurement vectors. *IEEE Trans. Sig. Proc.*, 54(12):4634–4643, Dec. 2006.

- [8] M. Mishali and Y. C. Eldar. Reduce and boost: Recovering arbitrary sets of jointly sparse vectors. *IEEE Trans. Sig. Proc.*, 56(10):4692–4702, Oct. 2008.
- [9] Y. C. Eldar and H. Rauhut. Average case analysis of multichannel sparse recovery using convex relaxation. *IEEE Trans. Inform. Theory*, 56(1):505–519, 2010.
- [10] Y. Nesterov. Gradient methods for minimizing composite objective function. In *CORE Discussion Paper 2007/76, Center for Operations Research and Econometrics (CORE)*. Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2007.
- [11] S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Trans. Sig. Proc.*, 57(7):2479–2493, 2009.
- [12] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. preprint (2010), available at <http://www-stat.stanford.edu/~tibs>.
- [13] J. Peng, J. Zhu, A. Bergamaschi, W. Han, D. Noh, J. Pollack, and P. Wang. Regularized multivariate regression for identifying master predictors with application to integrative genomics study of breast cancer. *Annals of Applied Statistics*, 4(1):53–77, 2010.
- [14] S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, June 2010.
- [15] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *ICML*, June 2010.
- [16] J. Starck, M. Elad, and D. Donoho. Image decomposition via the combination of sparse representations and a variational approach. *IEEE Trans. Image Proc.*, 14:1570–1582, 2004.
- [17] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. Technical Report HAL : inria-00516723, INRIA, 2010.
- [18] R. Tibshirani. Regression shrinkage and selection via the LASSO. *J. Royal Stat. Society: Series B*, 58(1):267–288, 1996.
- [19] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Scientific Computing*, 20(1):33–46, 1999.

- [20] D. Donoho. Compressed sensing. *IEEE Trans. on Inf. Theory*, 52(4):1289–1306, Apr 2006.
- [21] R. Giryes, M. Elad, and Y.C. Eldar. The projected GSURE for automatic parameter tuning in iterative shrinkage methods. *Applied and Computational Harmonic Analysis*, 30(3):407–422, 2011.
- [22] I. Ramírez and G. Sapiro. Sparse coding and dictionary learning based on the MDL principle. arXiv:1010.4751, 2010.
- [23] M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin. Non-parametric bayesian dictionary learning for sparse image representations. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2009.
- [24] B. Turlach, W. Venables, and S. Wright. Simultaneous variable selection. *Technometrics*, 27:349–363, 2004.
- [25] P. Sprechmann, I. Ramirez, and G. Sapiro. Collaborative hierarchical sparse modeling. In *CISS*, Mar. 2010.
- [26] P. Boufounos, G. Kutyniok, and H. Rauhut. Sparse recovery from combined fusion frame measurements. arXiv:0912.4988v1, 2010.
- [27] Y. C. Eldar, P. Kuppinger, and H. Bölcskei. Block-sparse signals: Uncertainty relations and efficient recovery. *IEEE Trans. SP*, 58(6):3042–3054, June 2010.
- [28] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. on Pure and Applied Mathematics*, 57:1413–1457, 2004.
- [29] E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52(2):489–509, February 2006.
- [30] E. Candès. The restricted isometry property and its implications for compressed sensing. *C. R. Acad. Sci. Paris S'er. I Math.*, 346:589–592, 2008.
- [31] J. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inform. Theory*, 50(10):2231–2242, October 2004.

- [32] M. Stojnic. Block-length dependent thresholds in block-sparse compressed sensing. arXiv:0907.3679, July 2009.
- [33] A. d’Aspremont, L. El Ghaoui, M. Jordan, and G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *Neural Information Processing Systems*, 17, 2004.
- [34] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2), 2003.
- [35] B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse PCA: Exact & greedy algorithms. *Neural Information Processing Systems*, 18, 2006.
- [36] I. Ramirez, P. Sprechmann, and G. Sapiro. Classification and clustering via dictionary learning with structured incoherence. In *CVPR*, June 2010.
- [37] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and Lawrence Carin. Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images. IMA Preprint, April 2010, <http://www.ima.umn.edu/preprints/apr2010/2307.pdf>.
- [38] K. Rosenblum, L. Zelnik-Manor, and Y. C. Eldar. Sensing matrix optimization for block-sparse decoding. arXiv:1009.1533, Sep. 2010.
- [39] N. Shoham and M. Elad. Alternating KSVD-denoising for texture separation. In *The IEEE 25-th Convention of Electrical and Electronics Engineers in Israel*, 2008.
- [40] P. Sprechmann, I. Ramirez, P. Cancela, and G. Sapiro. Collaborative sources identification in mixed signals via hierarchical sparse modeling. arXiv:1010.4893, 2010.
- [41] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.



Universal Regularizers For Robust Sparse Coding and Modeling

Ignacio Ramírez and Guillermo Sapiro

Department of Electrical and Computer Engineering, University of Minnesota

Sparse data models, where data is assumed to be well represented as a linear combination of a few elements from a dictionary, have gained considerable attention in recent years, and their use has led to state-of-the-art results in many signal and image processing tasks. It is now well understood that the choice of the sparsity regularization term is critical in the success of such models. Based on a codelength minimization interpretation of sparse coding, and using tools from universal coding theory, we propose a framework for designing sparsity regularization terms which have theoretical and practical advantages when compared to the more standard ℓ_0 or ℓ_1 ones. The presentation of the framework and theoretical foundations is complemented with examples that show its practical advantages in image denoising, zooming and classification.

1 Introduction

Sparse modeling calls for constructing a succinct representation of some data as a combination of a few typical patterns (*atoms*) learned from the data itself. Significant contributions to the theory and practice of learning such collections of atoms (usually called *dictionaries* or *codebooks*), e.g., [1, 2, 3], and of representing the actual data in terms of them, e.g., [4, 5, 6], have been developed in recent years, leading to state-of-the-art results in many signal and image processing tasks [7, 8, 9, 10]. We refer the reader for example to [11] for

a recent review on the subject.

A critical component of sparse modeling is the actual sparsity of the representation, which is controlled by a regularization term (*regularizer* for short) and its associated parameters. The choice of the functional form of the regularizer and its parameters is a challenging task. Several solutions to this problem have been proposed in the literature, ranging from the automatic tuning of the parameters [12] to Bayesian models, where these parameters are themselves considered as random variables [13, 12, 14]. In this work we adopt the interpretation of sparse coding as a codelength minimization problem. This is a natural and objective method for assessing the quality of a statistical model for describing given data, and which is based on the Minimum Description Length (MDL) principle [15]. In this framework, the regularization term in the sparse coding formulation is interpreted as the cost in bits of describing the sparse linear coefficients used to reconstruct the data. Several works on image coding using this approach were developed in the 1990's under the name of "complexity-based" or "compression-based" coding, following the popularization of MDL as a powerful statistical modeling tool [16, 17, 18]. The focus on these early works was in denoising using wavelet basis, using either generic asymptotic results from MDL or fixed probability models, in order to compute the description length of the coefficients. A later, major breakthrough in MDL theory was the adoption of *universal coding* tools to compute optimal codelengths. In this work, we improve and extend on previous results in this line of work by designing regularization terms based on such universal codes for image coefficients, meaning that the codelengths obtained when encoding the coefficients of any (natural) image with such codes will be close to the shortest codelengths that can be obtained with any model fitted specifically for that particular instance of coefficients. The resulting framework not only formalizes sparse coding from the MDL and universal coding perspectives but also leads to a family of *universal regularizers* which we show to consistently improve results in image processing tasks such as denoising and classification. These

models also enjoy several desirable theoretical and practical properties such as statistical consistency (in certain cases), improved robustness to outliers in the data, and improved sparse signal recovery (e.g., decoding of sparse signals from a compressive sensing point of view [19]) when compared with the traditional ℓ_0 and ℓ_1 -based techniques in practice. These models also yield to the use of a simple and efficient optimization technique for solving the corresponding sparse coding problems as a series of weighted ℓ_1 subproblems, which in turn, can be solved with off-the-shelf algorithms such as LARS [6] or IST [5]. Details are given in the sequel.

Finally, we apply our universal regularizers not only for coding using fixed dictionaries, but also for learning the dictionaries themselves, leading to further improvements in all the aforementioned tasks.

The remainder of this paper is organized as follows: in Section 2 we introduce the standard framework of sparse modeling. Section 3 is dedicated to the derivation of our proposed universal sparse modeling framework, while Section 4 deals with its implementation. Section 5 presents experimental results showing the practical benefits of the proposed framework in image denoising, zooming and classification tasks. Concluding remarks are given in Section 6.

2 Sparse modeling and the need for better models

Let $\mathbf{X} \in \mathbb{R}^{M \times N}$ be a set of N column data samples $\mathbf{x}_j \in \mathbb{R}^M$, $\mathbf{D} \in \mathbb{R}^{M \times K}$ a dictionary of K column atoms $\mathbf{d}_k \in \mathbb{R}^M$, and $\mathbf{A} \in \mathbb{R}^{K \times N}$, $\mathbf{a}_j \in \mathbb{R}^K$, a set of reconstruction coefficients such that $\mathbf{X} = \mathbf{D}\mathbf{A}$. We use \mathbf{a}_k^T to denote the k -th row of \mathbf{A} , the coefficients associated to the k -th atom in \mathbf{D} . For each $j = 1, \dots, N$ we define the *active set* of \mathbf{a}_j as $\mathcal{A}_j = \{k : a_{kj} \neq 0, 1 \leq k \leq K\}$, and $\|\mathbf{a}_j\|_0 = |\mathcal{A}_j|$ as its cardinality. The goal of sparse modeling is to design a dictionary \mathbf{D} such that for all or most data samples \mathbf{x}_j , there exists a coefficients vector \mathbf{a}_j such that

$\mathbf{x}_j \approx \mathbf{D} \mathbf{a}_j$ and $\|\mathbf{a}_j\|_0$ is small (usually below some threshold $L \ll K$). Formally, we would like to solve the following problem

$$\min_{\mathbf{D}, \mathbf{A}} \sum_{j=1}^N \psi(\mathbf{a}_j) \quad \text{s.t.} \quad \|\mathbf{x}_j - \mathbf{D} \mathbf{a}_j\|_2^2 \leq \epsilon, \quad j = 1, \dots, N, \quad (\text{C.1})$$

where $\psi(\cdot)$ is a regularization term which induces sparsity in the columns of the solution \mathbf{A} . Usually the constraint $\|\mathbf{d}_k\|_2 \leq 1$, $k = 1, \dots, K$, is added, since otherwise we can always decrease the cost function arbitrarily by multiplying \mathbf{D} by a large constant and dividing \mathbf{A} by the same constant. When \mathbf{D} is fixed, the problem of finding a sparse \mathbf{a}_j for each sample \mathbf{x}_j is called sparse coding,

$$\mathbf{a}_j = \arg \min_{\mathbf{a}} \psi(\mathbf{a}_j) \quad \text{s.t.} \quad \|\mathbf{x}_j - \mathbf{D} \mathbf{a}_j\|_2^2 \leq \epsilon. \quad (\text{C.2})$$

Among possible choices of $\psi(\cdot)$ are the ℓ_0 pseudo-norm, $\psi(\cdot) = \|\cdot\|_0$, and the ℓ_1 norm. The former tries to solve directly for the sparsest \mathbf{a}_j , but since it is non-convex, it is commonly replaced by the ℓ_1 norm, which is its closest convex approximation. Furthermore, under certain conditions on (fixed) \mathbf{D} and the sparsity of \mathbf{a}_j , the solutions to the ℓ_0 and ℓ_1 -based sparse coding problems coincide (see for example [19]). The problem (C.1) is also usually formulated in Lagrangian form,

$$\min_{\mathbf{D}, \mathbf{A}} \sum_{j=1}^N \|\mathbf{x}_j - \mathbf{D} \mathbf{a}_j\|_2^2 + \lambda \psi(\mathbf{a}_j), \quad (\text{C.3})$$

along with its respective sparse coding problem when \mathbf{D} is fixed,

$$\mathbf{a}_j = \arg \min_{\mathbf{a}} \|\mathbf{x}_j - \mathbf{D} \mathbf{a}\|_2^2 + \lambda \psi(\mathbf{a}). \quad (\text{C.4})$$

Even when the regularizer $\psi(\cdot)$ is convex, the sparse modeling problem, in any of its forms,

is jointly non-convex in (\mathbf{D}, \mathbf{A}) . Therefore, the standard approach to find an approximate solution is to use alternate minimization: starting with an initial dictionary $\mathbf{D}^{(0)}$, we minimize (C.3) alternatively in \mathbf{A} via (C.2) or (C.4) (sparse coding step), and \mathbf{D} (dictionary update step). The sparse coding step can be solved efficiently when $\psi(\cdot) = \|\cdot\|_1$ using for example IST [5] or LARS [6], or with OMP [20] when $\psi(\cdot) = \|\cdot\|_0$. The dictionary update step can be done using for example MOD [2] or K-SVD [1].

2.1 Interpretations of the sparse coding problem

We now turn our attention to the sparse coding problem: given a fixed dictionary \mathbf{D} , for each sample vector \mathbf{x}_j , compute the sparsest vector of coefficients \mathbf{a}_j that yields a good approximation of \mathbf{x}_j . The sparse coding problem admits several interpretations. What follows is a summary of these interpretations and the insights that they provide into the properties of the sparse models that are relevant to our derivation.

Model selection in statistics

Using the ℓ_0 norm as $\psi(\cdot)$ in (C.4) is known in the statistics community as the Akaike's Information Criterion (AIC) when $\lambda = 1$, or the Bayes Information Criterion (BIC) when $\lambda = \frac{1}{2} \log M$, two popular forms of model selection (see [21, Chapter 7]). In this context, the ℓ_1 regularizer was introduced in [22], again as a convex approximation of the above model selection methods, and is commonly known (either in its constrained or Lagrangian forms) as the *Lasso*. Note however that, in the regression interpretation of (C.4), the role of \mathbf{D} and \mathbf{X} is very different.

Maximum a posteriori

Another interpretation of (C.4) is that of a maximum a posteriori (MAP) estimation of \mathbf{a}_j in the logarithmic scale, that is

$$\begin{aligned} \mathbf{a}_j &= \arg \max_{\mathbf{a}} \{\log P(\mathbf{a}|\mathbf{x}_j)\} = \arg \max_{\mathbf{a}} \{\log P(\mathbf{x}_j|\mathbf{a}) + \log P(\mathbf{a})\} \\ &= \arg \min_{\mathbf{a}} \{-\log P(\mathbf{x}_j|\mathbf{a}) - \log P(\mathbf{a})\}, \end{aligned} \quad (\text{C.5})$$

where the observed samples \mathbf{x}_j are assumed to be contaminated with additive, zero mean, IID Gaussian noise with variance σ^2 , $P(\mathbf{x}_j|\mathbf{a}) \propto e^{-\frac{1}{2\sigma^2} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2}$, and a *prior probability model* on \mathbf{a} with the form $P(\mathbf{a}) \propto e^{-\theta\psi(\mathbf{a})}$ is considered. The energy term in Equation (C.4) follows by plugging the previous two probability models into (C.5) and factorizing $2\sigma^2$ into $\lambda = 2\sigma^2\theta$. According to (C.5), the ℓ_1 regularizer corresponds to an IID Laplacian prior with mean 0 and inverse-scale parameter θ , $P(\mathbf{a}) = \prod_{k=1}^K \theta e^{-\theta|a_k|} = \theta^K e^{-\theta\|\mathbf{a}\|_1}$, which has a special meaning in signal processing tasks such as image or audio compression. This is due to the widely accepted fact that representation coefficients derived from predictive coding of continuous-valued signals, and, more generally, responses from zero-mean filters, are well modeled using Laplacian distributions. For example, for the special case of DCT coefficients of image patches, an analytical study of this phenomenon is provided in [23], along with further references on the subject.

Codelength minimization

Sparse coding, in all its forms, has yet another important interpretation. Suppose that we have a fixed dictionary \mathbf{D} and that we want to use it to compress an image, either losslessly by encoding the reconstruction coefficients \mathbf{A} and the residual $\mathbf{X} - \mathbf{D}\mathbf{A}$, or in a lossy manner, by obtaining a good approximation $\mathbf{X} \approx \mathbf{D}\mathbf{A}$ and encoding only \mathbf{A} . Consider for example the latter case. Most modern compression schemes consist of two parts: a

probability assignment stage, where the data, in this case \mathbf{A} , is assigned a probability $P(\mathbf{A})$, and an encoding stage, where a code $C(\mathbf{A})$ of length $L(\mathbf{A})$ bits is assigned to the data given its probability, so that $L(\mathbf{A})$ is as short as possible. The techniques known as Arithmetic and Huffman coding provide the best possible solution for the encoding step, which is to approximate the Shannon ideal codelength $L(\mathbf{A}) = -\log P(\mathbf{A})$ [24, Chapter 5]. Therefore, modern compression theory deals with finding the coefficients \mathbf{A} that maximize $P(\mathbf{A})$, or, equivalently, that minimize $-\log P(\mathbf{A})$. Now, to encode \mathbf{X} lossily, we obtain coefficients \mathbf{A} such that each data sample \mathbf{x}_j is approximated up to a certain ℓ_2 distortion ϵ , $\|\mathbf{x}_j - \mathbf{D}\mathbf{a}_j\|_2^2 \leq \epsilon$. Therefore, given a model $P(\mathbf{a})$ for a vector of reconstruction coefficients, and assuming that we encode each sample independently, the optimum vector of coefficients \mathbf{a}_j for each sample \mathbf{x}_j will be the solution to the optimization problem

$$\mathbf{a}_j = \arg \min_{\mathbf{a}} -\log P(\mathbf{a}) \quad \text{s.t.} \quad \|\mathbf{x}_j - \mathbf{D}\mathbf{a}_j\|_2^2 \leq \epsilon, \quad (\text{C.6})$$

which, for the choice $P(\mathbf{a}) \propto e^{-\psi(\mathbf{a})}$ coincides with the error constrained sparse coding problem (C.2). Suppose now that we want lossless compression. In this case we also need to encode the reconstruction residual $\mathbf{x}_j - \mathbf{D}\mathbf{a}_j$. Since $P(\mathbf{x}, \mathbf{a}) = P(\mathbf{x}|\mathbf{a})P(\mathbf{a})$, the combined codelength will be

$$L(\mathbf{x}_j, \mathbf{a}_j) = -\log P(\mathbf{x}_j, \mathbf{a}_j) = -\log P(\mathbf{x}_j|\mathbf{a}_j) - \log P(\mathbf{a}_j). \quad (\text{C.7})$$

Therefore, obtaining the best coefficients \mathbf{a}_j amounts to solving $\min_{\mathbf{a}} L(\mathbf{x}_j, \mathbf{a}_j)$, which is precisely the MAP formulation of (C.5), which in turn, for proper choices of $P(\mathbf{x}|\mathbf{a})$ and $P(\mathbf{a})$, leads to the Lagrangian form of sparse coding (C.4).¹

¹Laplacian models, as well as Gaussian models, are probability distributions over \mathbb{R} , characterized by continuous probability density functions, $f(a) = F'(a)$, $F(a) = P(x \leq a)$. If the reconstruction coefficients are considered real numbers, under any of these distributions, any instance of $\mathbf{A} \in \mathbb{R}^{K \times N}$ will have measure 0, that is, $P(\mathbf{A}) = 0$. In order to use such distributions as our models for the data, we assume that the coefficients in \mathbf{A} are quantized to a precision Δ , small enough for the density function $f(a)$ to be approximately constant

As one can see, the codelength interpretation of sparse coding is able to unify and interpret both the constrained and unconstrained formulations into one consistent framework. Furthermore, this framework offers a natural and objective measure for comparing the quality of different models $P(\mathbf{x}|\mathbf{a})$ and $P(\mathbf{a})$ in terms of the codelengths obtained.

Remarks on related work

As mentioned in the introduction, the codelength interpretation of signal coding was already studied in the context of orthogonal wavelet-based denoising. An early example of this line of work considers a regularization term which uses the Shannon Entropy function $\sum p_i \log p_i$ to give a measure of the sparsity of the solution [16]. However, the Entropy function is not used as measure of the ideal codelength for describing the coefficients, but as a measure of the sparsity (actually, group sparsity) of the solution. The MDL principle was applied to the signal estimation problem in [18]. In this case, the codelength term includes the description of both the location and the magnitude of the nonzero coefficients. Although a pioneering effort, the model assumed in [18] for the coefficient magnitude is a uniform distribution on $[0, 1]$, which does not exploit a priori knowledge of image coefficient statistics, and the description of the support is slightly wasteful. Furthermore, the codelength expression used is an asymptotic result, actually equivalent to BIC (see Section 2.1) which can be misleading when working with small sample sizes, such as when encoding small image patches, as in current state of the art image processing applications. The uniform distribution was later replaced by the *universal code for integers* [25] in [17]. However, as in [18], the model is so general that it does not perform well for the specific case of coefficients arising from image decompositions, leading to poor results. In contrast, our models are derived following a careful analysis of image coefficient statistics. Finally,

in any interval $[a - \Delta/2, a + \Delta/2]$, $a \in \mathbb{R}$, so that we can approximate $P(a) \approx \Delta f(a)$, $a \in \mathbb{R}$. Under these assumptions, $-\log P(a) \approx -\log f(a) - \log \Delta$, and the effect of Δ on the codelength produced by any model is the same. Therefore, we will omit Δ in the sequel, and treat density functions and probability distributions interchangeably as $P(\cdot)$. Of course, in real compression applications, Δ needs to be tuned.

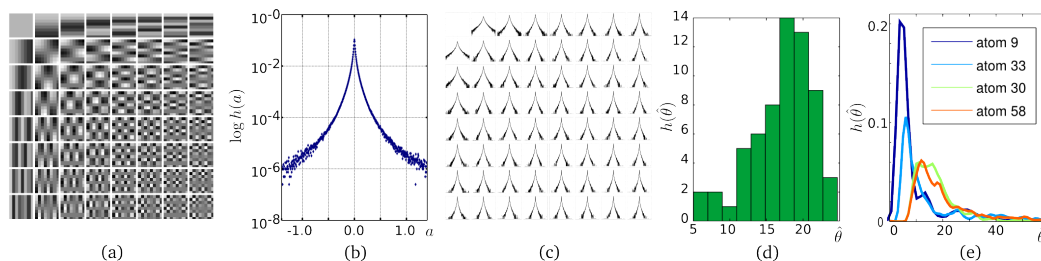


Figure C.1: Standard 8×8 DCT dictionary (a), global empirical distribution of the coefficients in \mathbf{A} (b, log scale), empirical distributions of the coefficients associated to each of the $K = 64$ DCT atoms (c, log scale). The distributions in (c) have a similar heavy tailed shape (heavier than Laplacian), but the variance in each case can be significantly different. (d) Histogram of the $K = 64$ different $\hat{\theta}_k$ values obtained by fitting a Laplacian distribution to each row \mathbf{a}_k^T of \mathbf{A} . Note that there are significant occurrences between $\hat{\theta} = 5$ to $\hat{\theta} = 25$. The coefficients \mathbf{A} used in (b-d) were obtained from encoding 10^6 8×8 patches (after removing their DC component) randomly sampled from the Pascal 2006 dataset of natural images [26]. (e) Histograms showing the spatial variability of the best local estimations of $\hat{\theta}_k$ for a few rows of \mathbf{A} across different regions of an image. In this case, the coefficients \mathbf{A} correspond to the sparse encoding of all 8×8 patches from a single image, in scan-line order. For each k , each value of $\hat{\theta}_k$ was computed from a random contiguous block of 250 samples from \mathbf{a}_k^T . The procedure was repeated 4000 times to obtain an empirical distribution. The wide supports of the empirical distributions indicate that the estimated $\hat{\theta}$ can have very different values, even for the same atom, depending on the region of the data from where the coefficients are taken.

probability models suitable to image coefficient statistics of the form $P(a) \propto e^{-|a|^\beta}$ (known as generalized Gaussians) were applied to the MDL-based signal coding and estimation framework in [17]. The justification for such models is based on the empirical observation that sparse coefficients statistics exhibit “heavy tails” (see next section). However, the choice is ad hoc and no optimality criterion is available to compare it with other possibilities. Moreover, there is no closed form solution for performing parameter estimation on such family of models, requiring numerical optimization techniques. In Section 3, we derive a number of probability models for which parameter estimation can be computed efficiently in closed form, and which are guaranteed to optimally describe image coefficients.

2.2 The need for a better model

As explained in the previous subsection, the use of the ℓ_1 regularizer implies that all the coefficients in \mathbf{A} share the same Laplacian parameter θ . However, as noted in [23] and references therein, the empirical variance of coefficients associated to different atoms, that is, of the different rows \mathbf{a}_k^T of \mathbf{A} , varies greatly with $k = 1 \dots, K$. This is clearly seen in Figures C.1(a-c), which show the empirical distribution of DCT coefficients of 8×8 patches. As the variance of a Laplacian is $2/\theta^2$, different variances indicate different underlying θ . The histogram of the set $\{\hat{\theta}_k, k = 1, \dots, K\}$ of estimated Laplacian parameters for each row k , Figure C.1(d), shows that this is indeed the case, with significant occurrences of values of $\hat{\theta}$ in a range of 5 to 25.

The straightforward modification suggested by this phenomenon is to use a model where each row of \mathbf{A} has its own weight associated to it, leading to a weighted ℓ_1 regularizer. However, from a modeling perspective, this results in K parameters to be adjusted instead of just one, which often results in poor generalization properties. For example, in the cases studied in Section 5, even with thousands of images for learning these parameters, the results of applying the learned model to new images were always significantly worse (over 1dB in estimation problems) when compared to those obtained using simpler models such as an unweighted ℓ_1 .² One reason for this failure may be that real images, as well as other types of signals such as audio samples, are far from stationary. In this case, even if each atom k is associated to its own θ_k (λ_k), the optimal value of θ_k can have significant local variations at different positions or times. This effect is shown in Figure C.1(e), where, for each k , each θ_k was re-estimated several times using samples from different regions of an image, and the histogram of the different estimated values of $\hat{\theta}_k$ was computed. Here again we used the DCT basis as the dictionary \mathbf{D} .

²Note that this is the case when the weights are found by maximum likelihood. Other applications of weighted ℓ_1 regularizers, using other types of weighting strategies, are known to improve over ℓ_1 -based ones for certain applications (see e.g. [14]).

The need for a flexible model which at the same time has a small number of parameters leads naturally to Bayesian formulations where the different possible λ_k are “marginalized out” by imposing an hyper-prior distribution on λ , sampling λ using its posterior distribution, and then averaging the estimates obtained with the sampled sparse-coding problems. Examples of this recent line of work, and the closely related Bayesian Compressive Sensing, are developed for example in [27, 28, 29, 30]. Despite of its promising results, the Bayesian approach is often criticized due to the potentially expensive sampling process (something which can be reduced for certain choices of the priors involved [27]), arbitrariness in the choice of the priors, and lack of proper theoretical justification for the proposed models [30].

In this work we pursue the same goal of deriving a more flexible and accurate sparse model than the traditional ones, while avoiding an increase in the number of parameters and the burden of possibly solving several sampled instances of the sparse coding problem. For this, we deploy tools from the very successful information-theoretic field of universal coding, which is an extension of the compression scenario summarized above in Section 2.1, when the probability model for the data to be described is itself unknown and has to be described as well.

3 Universal models for sparse coding

Following the discussion in the preceding section, we now have several possible scenarios to deal with. First, we may still want to consider a single value of θ to work well for all the coefficients in \mathbf{A} , and try to design a sparse coding scheme that does not depend on prior knowledge on the value of θ . Secondly, we can consider an independent (but not identically distributed) Laplacian model where the underlying parameter θ can be different for each atom \mathbf{d}_k , $k = 1, \dots, K$. In the most extreme scenario, we can consider each single coefficient

a_{kj} in \mathbf{A} to have its own unknown underlying θ_{kj} and yet, we would like to encode each of these coefficients (almost) as if we knew its hidden parameter.

The first two scenarios are the ones which fit the original purpose of universal coding theory [31], which is the design of optimal codes for data whose probability models are unknown, and the models themselves are to be encoded as well in the compressed representation.

Now we develop the basic ideas and techniques of universal coding applied to the first scenario, where the problem is to describe \mathbf{A} as an IID Laplacian with unknown parameter θ . Assuming a known parametric form for the prior, with unknown parameter θ , leads to the concept of a *model class*. In our case, we consider the class $\mathcal{M} = \{P(\mathbf{A}|\theta) : \theta \in \Theta\}$ of all IID Laplacian models over $\mathbf{A} \in \mathbb{R}^{K \times N}$, where

$$P(\mathbf{A}|\theta) = \prod_{j=1}^N \prod_{k=1}^K P(a_{kj}|\theta), \quad P(a_{kj}|\theta) = \theta e^{-\theta|a_{kj}|}$$

and $\Theta \subseteq \mathbb{R}^+$. The goal of universal coding is to find a probability model $Q(\mathbf{A})$ which can fit \mathbf{A} as well as the model in \mathcal{M} that best fits \mathbf{A} after having observed it. A model $Q(\mathbf{A})$ with this property is called *universal* (with respect to the model \mathcal{M}).

For simplicity, in the following discussion we consider the coefficient matrix \mathbf{A} to be arranged as a single long column vector of length $n = K \times N$, $\mathbf{a} = (a_1, \dots, a_n)$. We also use the letter a without sub-index to denote the value of a random variable representing coefficient values.

First we need to define a criterion for comparing the fitting quality of different models. In universal coding theory this is done in terms of the codelengths $L(\mathbf{a})$ required by each model to describe \mathbf{a} .

If the model consists of a single probability distribution $P(\cdot)$, we know from Section 2.1

that the optimum codelength corresponds to $L_p(\mathbf{a}) = -\log P(\mathbf{a})$. Moreover, this relationship defines a one-to-one correspondence between distributions and codelengths, so that for any coding scheme $L_Q(\mathbf{a})$, $Q(\mathbf{a}) = 2^{-L_Q(\mathbf{a})}$. Now suppose that we are restricted to a class of models \mathcal{M} , and that we need choose the model $\hat{P} \in \mathcal{M}$ that assigns the shortest codelength to a particular instance of \mathbf{a} . We then have that \hat{P} is the model in \mathcal{M} that assigns the maximum probability to \mathbf{a} . For a class \mathcal{M} parametrized by θ , this corresponds to $\hat{P} = P(\mathbf{a}|\hat{\theta}(\mathbf{a}))$, where $\hat{\theta}(\mathbf{a})$ is the maximum likelihood estimator (MLE) of the model class parameter θ given \mathbf{a} (we will usually omit the argument and just write $\hat{\theta}$). Unfortunately, we also need to include the value of $\hat{\theta}$ in the description of \mathbf{a} for the decoder to be able to reconstruct it from the code $C(\mathbf{a})$. Thus, we have that any model $Q(\mathbf{a})$ inducing valid codelengths $L_Q(\mathbf{a})$ will have $L_Q(\mathbf{a}) > -\log P(\mathbf{a}|\hat{\theta})$. The overhead of $L_Q(\mathbf{a})$ with respect to $-\log P(\mathbf{a}|\hat{\theta})$ is known as the *codelength regret*,

$$\mathcal{R}(\mathbf{a}, Q) := L_Q(\mathbf{a}) - (-\log P(\mathbf{a}|\hat{\theta}(\mathbf{a}))) = -\log Q(\mathbf{a}) + \log P(\mathbf{a}|\hat{\theta}(\mathbf{a})).$$

A model $Q(\mathbf{a})$ (or, more precisely, a sequence of models, one for each data length n) is called *universal* if $\mathcal{R}(\mathbf{a}, Q)$ grows sublinearly in n for all possible realizations of \mathbf{a} , that is $\frac{1}{n}\mathcal{R}(\mathbf{a}, Q) \rightarrow 0$, $\forall \mathbf{a} \in \mathbb{R}^n$, so that the codelength regret with respect to the MLE becomes asymptotically negligible.

There are a number of ways to construct universal probability models. The simplest one is the so called *two-part code*, where the data is described in two parts. The first part describes the optimal parameter $\hat{\theta}(\mathbf{a})$ and the second part describes the data according to the model with the value of the estimated parameter $\hat{\theta}$, $P(\mathbf{a}|\hat{\theta}(\mathbf{a}))$. For uncountable parameter spaces Θ , such as a compact subset of \mathbb{R} , the value of $\hat{\theta}$ has to be quantized in order to be described with a finite number of bits d . We call the quantized parameter $\hat{\theta}_d$.

The regret for this model is thus

$$\mathcal{R}(\mathbf{a}, Q) = L(\hat{\theta}_d) + L(\mathbf{a}|\hat{\theta}_d) - L(\mathbf{a}|\hat{\theta}) = L(\hat{\theta}_d) - \log P(\mathbf{a}|\hat{\theta}_d) - (-\log P(\mathbf{a}|\hat{\theta})).$$

The key for this model to be universal is in the choice of the quantization step for the parameter $\hat{\theta}$, so that both its description $L(\hat{\theta}_d)$, and the difference $-\log P(\mathbf{a}|\hat{\theta}_d) - (-\log P(\mathbf{a}|\hat{\theta}))$, grow sublinearly. This can be achieved by letting the quantization step shrink as $O(1/\sqrt{n})$ [15], thus requiring $d = O(0.5 \log n)$ bits to describe each dimension of $\hat{\theta}_d$. This gives us a total regret for two-part codes which grows as $\frac{\dim(\Theta)}{2} \log n$, where $\dim(\Theta)$ is the dimension of the parameter space Θ .

Another important universal code is the so called *Normalized Maximum Likelihood* (NML) [32]. In this case the universal model $Q^*(\mathbf{a})$ corresponds to the model that minimizes the worst case regret,

$$Q^*(\mathbf{a}) = \min_Q \max_{\mathbf{a}} \{-\log Q(\mathbf{a}) + \log P(\mathbf{a}|\hat{\theta}(\mathbf{a}))\},$$

which can be written in closed form as $Q^*(\mathbf{a}) = \frac{P(\mathbf{a}|\hat{\theta}(\mathbf{a}))}{\mathcal{C}(\mathcal{M}, n)}$, where the normalization constant $\mathcal{C}(\mathcal{M}, n) := \sum_{\mathbf{a} \in \mathbb{R}^n} P(\mathbf{a}|\hat{\theta}(\mathbf{a})) d\mathbf{a}$ is the value of the minimax regret and depends only on \mathcal{M} and the length of the data n .³ Note that the NML model requires $\mathcal{C}(\mathcal{M}, n)$ to be finite, something which is often not the case.

The two previous examples are good for assigning a probability to coefficients \mathbf{a} that have already been computed, but they cannot be used as a model for computing the coefficients themselves since they depend on having observed them in the first place. For this and other reasons that will become clearer later, we concentrate our work on a third important

³The minimax optimality of $Q^*(\mathbf{a})$ derives from the fact that it defines a complete uniquely decodable code for all data \mathbf{a} of length n , that is, it satisfies the Kraft inequality with equality. $\sum_{\mathbf{a} \in \mathbb{R}^n} 2^{-L_{Q^*}(\mathbf{a})} = 1$. Since every uniquely decodable code with lengths $\{L_Q(\mathbf{a}) : \mathbf{a} \in \mathbb{R}^n\}$ must satisfy the Kraft inequality (see [24, Chapter 5]), if there exists a value of \mathbf{a} such that $L_Q(\mathbf{a}) < L_{Q^*}(\mathbf{a})$ (that is $2^{-L_Q(\mathbf{a})} > 2^{-L_{Q^*}(\mathbf{a})}$), then there exists a vector \mathbf{a}' for which $L_Q(\mathbf{a}') > L_{Q^*}(\mathbf{a}')$ for the Kraft inequality to hold. Therefore the regret of Q for \mathbf{a}' is necessarily greater than $\mathcal{C}(\mathcal{M}, n)$, which shows that Q^* is minimax optimal.

family of universal codes derived from the so called *mixture models* (also called *Bayesian mixtures*). In a mixture model, $Q(\mathbf{a})$ is a convex mixture of all the models $P(\mathbf{a}|\theta)$ in \mathcal{M} , indexed by the model parameter θ , $Q(\mathbf{a}) = \int_{\Theta} P(\mathbf{a}|\theta)w(\theta)d\theta$, where $w(\theta)$ specifies the weight of each model. Being a convex mixture implies that $w(\theta) \geq 0$ and $\int_{\Theta} w(\theta)d\theta = 1$, thus $w(\theta)$ is itself a probability measure over Θ . We will restrict ourselves to the particular case when \mathbf{a} is considered a sequence of independent random variables,⁴

$$Q(\mathbf{a}) = \prod_{j=1}^n Q_j(a_j), \quad Q_j(a_j) = \int_{\Theta} P(a_j|\theta)w_j(\theta)d\theta, \quad (\text{C.8})$$

where the mixing function $w_j(\theta)$ can be different for each sample j . An important particular case of this scheme is the so called *Sequential Bayes* code, in which $w_j(\theta)$ is computed sequentially as a posterior distribution based on previously observed samples, that is $w_j(\theta) = P(\theta|a_1, a_2, \dots, a_{n-1})$ [33, Chapter 6]. In this work, for simplicity, we restrict ourselves to the case where $w_j(\theta) = w(\theta)$ is the same for all j . The result is an IID model where the probability of each sample a_j is a mixture of some probability measure over \mathbb{R} ,

$$Q_j(a_j) = Q(a_j) = \int_{\Theta} P(a_j|\theta)w(\theta)d\theta, \quad \forall j = 1, \dots, N. \quad (\text{C.9})$$

A well known result for IID mixture (Bayesian) codes states that their asymptotic regret is $O(\frac{\dim(\Theta)}{2} \log n)$, thus stating their universality, as long as the weighting function $w(\theta)$ is positive, continuous and unimodal over Θ (see for example [33, Theorem 8.1],[34]). This gives us great flexibility on the choice of a weighting function $w(\theta)$ that guarantees universality. Of course, the results are asymptotic and the $o(\log n)$ terms can be large, so that the choice of $w(\theta)$ can have practical impact for small sample sizes.

In the following discussion we derive several IID mixture models for the Laplacian model

⁴More sophisticated models which include dependencies between the elements of \mathbf{a} are out of the scope of this work.

class \mathcal{M} . For this purpose, it will be convenient to consider the corresponding one-sided counterpart of the Laplacian, which is the exponential distribution over the absolute value of the coefficients, $|a|$, and then symmetrize back to obtain the final distribution over the signed coefficients a .

3.1 The conjugate prior

In general, (C.9) can be computed in closed form if $w(\theta)$ is the conjugate prior of $P(a|\theta)$. When $P(a|\theta)$ is an exponential (one-sided Laplacian), the conjugate prior is the Gamma distribution,

$$w(\theta|\kappa, \beta) = \Gamma(\kappa)^{-1} \theta^{\kappa-1} \beta^\kappa e^{-\beta\theta}, \quad \theta \in \mathbb{R}^+,$$

where κ and β are its *shape* and *scale* parameters respectively. Plugging this in (C.9) we obtain the *Mixture of exponentials* model (MOE), which has the following form (see Appendix 7 for the full derivation),

$$Q_{\text{MOE}}(a|\beta, \kappa) = \kappa \beta^\kappa (a + \beta)^{-(\kappa+1)}, \quad a \in \mathbb{R}^+. \quad (\text{C.10})$$

With some abuse of notation, we will also denote the symmetric distribution on a as MOE,

$$Q_{\text{MOE}}(a|\beta, \kappa) = \frac{1}{2} \kappa \beta^\kappa (|a| + \beta)^{-(\kappa+1)}, \quad a \in \mathbb{R}. \quad (\text{C.11})$$

Although the resulting prior has two parameters to deal with instead of one, we know from universal coding theory that, in principle, any choice of κ and β will give us a model whose codelength regret is asymptotically small.

Furthermore, being IID models, each coefficient of \mathbf{a} itself is modeled as a mixture of exponentials, which makes the resulting model over \mathbf{a} very well suited to the most flexible scenario where the “underlying” θ can be different for each a_j . In Section 5.2 we will

show that a single MOE distribution can fit each of the K rows of \mathbf{A} better than K separate Laplacian distributions fine-tuned to these rows, with a total of K parameters to be estimated. Thus, not only we can deal with one single unknown θ , but we can actually achieve maximum flexibility with only two parameters (κ and β). This property is particular of the mixture models, and does not apply to the other universal models presented.

Finally, if desired, both κ and β can be easily estimated using the method of moments (see Appendix 7). Given sample estimates of the first and second non-central moments, $\hat{\mu}_1 = \frac{1}{n} \sum_{j=1}^n |a_j|$ and $\hat{\mu}_2 = \frac{1}{n} \sum_{j=1}^n |a_j|^2$, we have that

$$\hat{\kappa} = 2(\hat{\mu}_2 - \hat{\mu}_1^2)/(\hat{\mu}_2 - 2\hat{\mu}_1^2) \quad \text{and} \quad \hat{\beta} = (\hat{\kappa} - 1)\hat{\mu}_1. \quad (\text{C.12})$$

When the MOE prior is plugged into (C.5) instead of the standard Laplacian, the following new sparse coding formulation is obtained,

$$a_j = \arg \min_{\mathbf{a}} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \lambda_{\text{MOE}} \sum_{k=1}^K \log(|a_k| + \beta), \quad (\text{C.13})$$

where $\lambda_{\text{MOE}} = 2\sigma^2(\kappa + 1)$. An example of the MOE regularizer, and the thresholding function it induces, is shown in Figure C.2 (center column) for $\kappa = 2.5, \beta = 0.05$. Smooth, differentiable non-convex regularizers such as the one in (C.13) have become a mainstream robust alternative to the ℓ_1 norm in statistics [35, 14]. Furthermore, it has been shown that the use of such regularizers in regression leads to consistent estimators which are able to identify the relevant variables in a regression model (oracle property) [35]. This is not always the case for the ℓ_1 regularizer, as was proved in [14]. The MOE regularizer has also been recently proposed in the context of compressive sensing [36], where it is conjectured to be better than the ℓ_1 -term at recovering sparse signals in compressive sensing applications.⁵ This conjecture was partially confirmed recently for non-convex regularizers of the

⁵In [36], the logarithmic regularizer arises from approximating the ℓ_0 pseudo-norm as an ℓ_1 -normalized

form $\psi(\mathbf{a}) = \|\mathbf{a}\|_r$ with $0 < r < 1$ in [37, 38], and for a more general family of non-convex regularizers including the one in (C.13) in [39]. In all cases, it was shown that the conditions on the sensing matrix (here \mathbf{D}) can be significantly relaxed to guarantee exact recovery if non-convex regularizers are used instead of the ℓ_1 norm, provided that the exact solution to the non-convex optimization problem can be computed. In practice, this regularizer is being used with success in a number of applications here and in [40, 41].⁶ Our experimental results in Section 5 provide further evidence on the benefits of the use of non-convex regularizers, leading to a much improved recovery accuracy of sparse coefficients compared to ℓ_1 and ℓ_0 . We also show in Section 5 that the MOE prior is much more accurate than the standard Laplacian to model the distribution of reconstruction coefficients drawn from a large database of image patches. We also show in Section 5 how these improvements lead to better results in applications such as image estimation and classification.

3.2 The Jeffreys prior

The Jeffreys prior for a parametric model class $\mathcal{M} = \{P(a|\theta), \theta \in \Theta\}$, is defined as

$$w(\theta) = \frac{\sqrt{|I(\theta)|}}{\int_{\Theta} \sqrt{|I(\xi)|} d\xi}, \quad \theta \in \Theta, \quad (\text{C.14})$$

where $|I(\theta)|$ is the determinant of the *Fisher information matrix*

$$I(\theta) = \left\{ E_{P(a|\tilde{\theta})} \left[-\frac{\partial^2}{\partial \tilde{\theta}^2} \log P(a|\tilde{\theta}) \right] \right\} \Big|_{\tilde{\theta}=\theta}. \quad (\text{C.15})$$

The Jeffreys prior is well known in Bayesian theory due to three important properties: it virtually eliminates the hyper-parameters of the model, it is invariant to the original element-wise sum, without the insight and theoretical foundation here reported.

⁶While these works support the use of such non-convex regularizers, none of them formally derives them using the universal coding framework as in this paper.

parametrization of the distribution, and it is a “non-informative prior,” meaning that it represents well the lack of prior information on the unknown parameter θ [42]. It turns out that, for quite different reasons, the Jeffreys prior is also of paramount importance in the theory of universal coding. For instance, it has been shown in [43] that the worst case regret of the mixture code obtained using the Jeffreys prior approaches that of the NML as the number of samples n grows. Thus, by using Jeffreys, one can attain the minimum worst case regret asymptotically, while retaining the advantages of a mixture (not needing hindsight of \mathbf{a}), which in our case means to be able to use it as a model for computing \mathbf{a} via sparse coding.

For the exponential distribution we have that $I(\theta) = \frac{1}{\theta^2}$. Clearly, if we let $\Theta = (0, \infty)$, the integral in (C.14) evaluates to ∞ . Therefore, in order to obtain a proper integral, we need to exclude 0 and ∞ from Θ (note that this was not needed for the conjugate prior). We choose to define $\Theta = [\theta_1, \theta_2]$, $0 < \theta_1 < \theta_2 < \infty$, leading to $w(\theta) = \frac{1}{\ln(\theta_2/\theta_1)} \frac{1}{\theta}$, $\theta \in [\theta_1, \theta_2]$. The resulting mixture, after being symmetrized around 0, has the following form (see Appendix 7):

$$Q_{\text{JOE}}(a|\theta_1, \theta_2) = \frac{1}{2\ln(\theta_2/\theta_1)} \frac{1}{|a|} \left(e^{-\theta_1|a|} - e^{-\theta_2|a|} \right), \quad a \in \mathbb{R}^+. \quad (\text{C.16})$$

We refer to this prior as a *Jeffreys mixture of exponentials* (JOE), and again overload this acronym to refer to the symmetric case as well. Note that although Q_{JOE} is not defined for $a = 0$, its limit when $a \rightarrow 0$ is finite and evaluates to $\frac{\theta_2 - \theta_1}{2\ln(\theta_2/\theta_1)}$. Thus, by defining $Q_{\text{JOE}}(0) = \frac{\theta_2 - \theta_1}{2\ln(\theta_2/\theta_1)}$, we obtain a prior that is well defined and continuous for all $a \in \mathbb{R}$. When plugged into (C.5), we get the JOE-based sparse coding formulation,

$$\min_{\mathbf{a}} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \lambda_{\text{JOE}} \sum_{k=1}^K \{\log|a_k| - \log(e^{-\theta_1|a_k|} - e^{-\theta_2|a_k|})\}, \quad (\text{C.17})$$

where, according to the convention just defined for $Q_{\text{JOE}}(0)$, we define $\psi_{\text{JOE}}(0) := \log(\theta_2 -$

θ_1). According to the MAP interpretation we have that $\lambda_{\text{JOE}} = 2\sigma^2$, coming from the Gaussian assumption on the approximation error as explained in Section 2.1.

As with MOE, the JOE-based regularizer, $\psi_{\text{JOE}}(\cdot) = -\log Q_{\text{JOE}}(\cdot)$, is continuous and differentiable in \mathbb{R}^+ , and its derivative converges to a finite value at zero, $\lim_{a \rightarrow 0} \psi'_{\text{JOE}}(a) = \frac{\theta_2^2 - \theta_1^2}{\theta_2 - \theta_1}$. As we will see later in Section 4, these properties are important to guarantee the convergence of sparse coding algorithms using non-convex priors. Note from (C.17) that we can rewrite the JOE regularizer as

$$\psi_{\text{JOE}}(a_k) = \log |a_k| - \log e^{-\theta_1 |a_k|} (1 - e^{-(\theta_2 - \theta_1) |a_k|}) = \theta_1 |a_k| + \log |a_k| - \log(1 - e^{-(\theta_2 - \theta_1) |a_k|}),$$

so that for sufficiently large $|a_k|$, $\log(1 - e^{-(\theta_2 - \theta_1) |a_k|}) \approx 0$, $\theta_1 |a_k| \gg \log |a_k|$, and we have that $\psi_{\text{JOE}}(|a_k|) \approx \theta_1 |a_k|$. Thus, for large $|a_k|$, the JOE regularizer behaves like ℓ_1 with $\lambda' = 2\sigma^2 \theta_1$. In terms of the probability model, this means that the tails of the JOE mixture behave like a Laplacian with $\theta = \theta_1$, with the region where this happens determined by the value of $\theta_2 - \theta_1$. The fact that the non-convex region of $\psi_{\text{JOE}}(\cdot)$ is confined to a neighborhood around 0 could help to avoid falling in bad local minima during the optimization (see Section 4 for more details on the optimization aspects). Finally, although having Laplacian tails means that the estimated \mathbf{a} will be biased [35], the sharper peak at 0 allows us to perform a more aggressive thresholding of small values, without excessively clipping large coefficients, which leads to the typical over-smoothing of signals recovered using an ℓ_1 regularizer. See Figure C.2 (rightmost column) for an example regularizer based on JOE with parameters $\theta_1 = 20$, $\theta_2 = 100$, and the thresholding function it induces.

The JOE regularizer has two hyper-parameters (θ_1, θ_2) which define Θ and that, in principle, need to be tuned. One possibility is to choose θ_1 and θ_2 based on the physical properties of the data to be modeled, so that the possible values of θ never fall outside of the range $[\theta_1, \theta_2]$. For example, in modeling patches from grayscale images with a limited

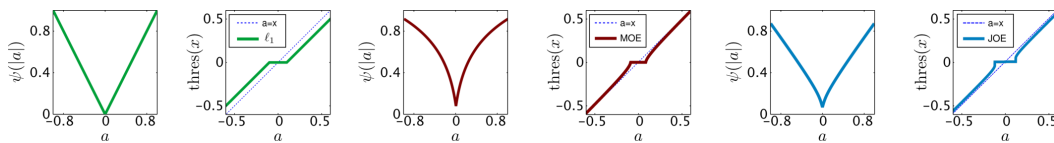


Figure C.2: Left to right: ℓ_1 (green), MOE (red) and JOE (blue) regularizers and their corresponding thresholding functions $\text{thres}(x) := \arg \min_a \{(x - a)^2 + \lambda \psi(|a|)\}$. The unbiasedness of MOE is due to the fact that large coefficients are not shrunk by the thresholding function. Also, although the JOE regularizer is biased, the shrinkage of large coefficients can be much smaller than the one applied to small coefficients.

dynamic range of $[0, 255]$ in a DCT basis, the maximum variance of the coefficients can never exceed 128^2 . The same is true for the minimum variance, which is defined by the quantization noise.

Having said this, in practice it is advantageous to adjust $[\theta_1, \theta_2]$ to the data at hand. In this case, although no closed form solutions exist for estimating $[\theta_1, \theta_2]$ using MLE or the method of moments, standard optimization techniques can be easily applied to obtain them. See Appendix 7 for details.

3.3 The conditional Jeffreys

A recent approach to deal with the case when the integral over Θ in the Jeffreys prior is improper, is the *conditional Jeffreys* [33, Chapter 11]. The idea is to construct a proper prior, based on the improper Jeffreys prior and the first few n_0 samples of \mathbf{a} , $(a_1, a_2, \dots, a_{n_0})$, and then use it for the remaining data. The key observation is that although the normalizing integral $\int \sqrt{I(\theta)} d\theta$ in the Jeffreys prior is improper, the unnormalized prior $w(\theta) = \sqrt{I(\theta)}$ can be used as a measure to weight $P(a_1, a_2, \dots, a_{n_0} | \theta)$,

$$w(\theta) = \frac{P(a_1, a_2, \dots, a_{n_0} | \theta) \sqrt{I(\theta)}}{\int_{\Theta} P(a_1, a_2, \dots, a_{n_0} | \xi) \sqrt{I(\xi)} d\xi}. \quad (\text{C.18})$$

It turns out that the integral in (C.18) usually becomes proper for small n_0 in the order

of $\dim(\Theta)$. In our case we have that for any $n_0 \geq 1$, the resulting prior is a $\text{Gamma}(\kappa_0, \beta_0)$ distribution with $\kappa_0 := n_0$ and $\beta_0 := \sum_{j=1}^{n_0} a_j$ (see Appendix 7 for details). Therefore, using the conditional Jeffreys prior in the mixture leads to a particular instance of MOE , which we denote by CMOE (although the functional form is identical to MOE), where the Gamma parameters κ and β are automatically selected from the data. This may explain in part why the Gamma prior performs so well in practice, as we will see in Section 5.

Furthermore, we observe that the value of β obtained with this approach (β_0) coincides with the one estimated using the method of moments for MOE if the κ in MOE is fixed to $\kappa = \kappa_0 + 1 = n_0 + 1$. Indeed, if computed from n_0 samples, the method of moments for MOE gives $\beta = (\kappa - 1)\mu_1$, with $\mu_1 = \frac{1}{n_0} \sum a_j$, which gives us $\beta = \frac{n_0+1-1}{n_0} \sum a_j = \beta_0$. It turns out in practice that the value of κ estimated using the method of moments gives a value between 2 and 3 for the type of data that we deal with (see Section 5), which is just above the minimum acceptable value for the CMOE prior to be defined, which is $n_0 = 1$. This justifies our choice of $n_0 = 2$ when applying CMOE in practice.

As n_0 becomes large, so does $\kappa_0 = n_0$, and the Gamma prior $w(\theta)$ obtained with this method converges to a Kronecker delta at the mean value of the Gamma distribution, $\delta_{\kappa_0/\beta_0}(\cdot)$. Consequently, when $w(\theta) \approx \delta_{\kappa_0/\beta_0}(\theta)$, the mixture $\int_{\Theta} P(a|\theta)w(\theta)d\theta$ will be close to $P(a|\kappa_0/\beta_0)$. Moreover, from the definition of κ_0 and β_0 we have that κ_0/β_0 is exactly the MLE of θ for the Laplacian distribution. Thus, for large n_0 , the conditional Jeffreys method approaches the MLE Laplacian model.

Although from a universal coding point of view this is not a problem, for large n_0 the conditional Jeffreys model will lose its flexibility to deal with the case when different coefficients in \mathbf{A} have different underlying θ . On the other hand, a small n_0 can lead to a prior $w(\theta)$ that is overfitted to the local properties of the first samples, which for non-stationary data such as image patches, can be problematic. Ultimately, n_0 defines a trade-off between the degree of flexibility and the accuracy of the resulting model.

4 Optimization and implementation details

All of the mixture models discussed so far yield non-convex regularizers, rendering the sparse coding problem non-convex in \mathbf{a} . It turns out however that these regularizers satisfy certain conditions which make the resulting sparse coding optimization well suited to be approximated using a sequence of successive convex sparse coding problems, a technique known as *Local Linear Approximation* (LLA) [44] (see also [41, 45] for alternative optimization techniques for such non-convex sparse coding problems). In a nutshell, suppose we need to obtain an approximate solution to

$$\mathbf{a}_j = \arg \min_{\mathbf{a}} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \sum_{k=1}^K \psi(|a_k|), \quad (\text{C.19})$$

where $\psi(\cdot)$ is a non-convex function over \mathbb{R}^+ . At each LLA iteration, we compute $\mathbf{a}_j^{(t+1)}$ by doing a first order expansion of $\psi(\cdot)$ around the K elements of the current estimate $a_{kj}^{(t)}$,

$$\tilde{\psi}_k^{(t)}(|a|) = \psi(|a_{kj}^{(t)}|) + \psi'(|a_{kj}^{(t)}|) \left(|a| - |a_{kj}^{(t)}| \right) = \psi'(|a_{kj}^{(t)}|)|a| + c_k,$$

and solving the convex weighted ℓ_1 problem that results after discarding the constant terms c_k ,

$$\begin{aligned} \mathbf{a}_j^{(t+1)} &= \arg \min_{\mathbf{a}} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \sum_{k=1}^K \tilde{\psi}_k^{(t)}(|a_k|) \\ &= \arg \min_{\mathbf{a}} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \sum_{k=1}^K \psi'(|a_{kj}^{(t)}|)|a_k| = \arg \min_{\mathbf{a}} \|\mathbf{x}_j - \mathbf{D}\mathbf{a}\|_2^2 + \sum_{k=1}^K \lambda_k^{(t)} |a_k| \end{aligned} \quad (\text{C.20})$$

where we have defined $\lambda_k^{(t)} := \lambda \psi'(|a_{kj}^{(t)}|)$. If $\psi'(\cdot)$ is continuous in $(0, +\infty)$, and right-continuous and finite at 0, then the LLA algorithm converges to a stationary point of (C.19) [14]. These conditions are met for both the MOE and JOE regularizers. Although, for the JOE

prior, the derivative $\psi'(\cdot)$ is not defined at 0, it converges to the limit $\frac{\theta_2^2 - \theta_1^2}{2(\theta_2 - \theta_1)}$ when $|a| \rightarrow 0$, which is well defined for $\theta_2 \neq \theta_1$. If $\theta_2 = \theta_1$, the JOE mixing function is a Kronecker delta and the prior becomes a Laplacian with parameter $\theta = \theta_1 = \theta_2$. Therefore we have that for all of the mixture models studied, the LLA method converges to a stationary point. In practice, we have observed that 5 iterations are enough to converge. Thus, the cost of sparse coding, with the proposed non-convex regularizers, is at most 5 times that of a single ℓ_1 sparse coding, and could be less in practice if warm restarts are used to begin each iteration.

Of course we need a starting point $\mathbf{a}_j^{(0)}$, and, being a non-convex problem, this choice will influence the approximation that we obtain. One reasonable choice, used in this work, is to define $a_{kj}^{(0)} = a_0$, $k = 1, \dots, K, j = 1, \dots, N$, where a_0 is a scalar so that $\psi'(a_0) = E_w[\theta]$, that is, so that the first sparse coding corresponds to a Laplacian regularizer whose parameter is the average value of θ as given by the mixing prior $w(\theta)$.

Finally, note that although the discussion here has revolved around the Lagrangian formulation to sparse coding of (C.4), this technique is also applicable to the constrained formulation of sparse-coding given by Equation (C.1) for a fixed dictionary \mathbf{D} .

Expected approximation error: Since we are solving a convex approximation to the actual target optimization problem, it is of interest to know how good this approximation is in terms of the original cost function. To give an idea of this, after an approximate solution \mathbf{a} is obtained, we compute the expected value of the difference between the true and approximate regularization term values. The expectation is taken, naturally, in terms of the assumed distribution of the coefficients in \mathbf{a} . Since the regularizers are separable, we can compute the error in a separable way as an expectation over each k -th coefficient, $\zeta_q(a_k) = E_{v \sim q} [\tilde{\psi}_k(v) - \psi(v)]$, where $\tilde{\psi}_k(\cdot)$ is the approximation of $\psi_k(\cdot)$ around the final

estimate of a_k . For the case of $q = \text{MOE}$, the expression obtained is (see Appendix)

$$\zeta_{\text{MOE}}(a_k, \kappa, \beta) = E_{v \sim \text{MOE}(\kappa, \beta)} [\tilde{\psi}_k(v) - \psi(v)] = \log(a_k + \beta) + \frac{1}{a_k + \beta} \left[a_k + \frac{\beta}{\kappa - 1} \right] - \log \beta - \frac{1}{\kappa}.$$

In the MOE case, for κ and β fixed, the minimum of ζ_{MOE} occurs when $a_k = \frac{\beta}{\kappa - 1} = \mu(\beta, \kappa)$.

We also have $\zeta_{\text{MOE}}(0) = (\kappa - 1)^{-1} - \kappa^{-1}$.

The function $\zeta_q(\cdot)$ can be evaluated on each coefficient of \mathbf{A} to give an idea of its quality. For example, in the experiments from Section 5, we obtained an average value of 0.16, which lies between $\zeta_{\text{MOE}}(0) = 0.19$ and $\min_a \zeta_{\text{MOE}}(a) = 0.09$. Depending on the experiment, this represents 6% to 7% of the total sparse coding cost function value, showing the efficiency of the proposed optimization.

Comments on parameter estimation: All the universal models presented so far, with the exception of the conditional Jeffreys, depend on hyper-parameters which in principle should be tuned for optimal performance (remember that they do not influence the universality of the model). If tuning is needed, it is important to remember that the proposed universal models are intended for reconstruction coefficients of *clean data*, and thus their hyper-parameters should be computed from statistics of clean data, or either by compensating the distortion in the statistics caused by noise (see for example [46]). Finally, note that when \mathbf{D} is linearly dependent and $\text{rank}(\mathbf{D}) = \mathbb{R}^M$, the coefficients matrix \mathbf{A} resulting from an exact reconstruction of \mathbf{X} will have many zeroes which are not properly explained by any continuous distribution such as a Laplacian. We sidestep this issue by computing the statistics only from the non-zero coefficients in \mathbf{A} . Dealing properly with the case $P(a = 0) > 0$ is beyond the scope of this work.

5 Experimental results

In the following experiments, the testing data \mathbf{X} are 8×8 patches drawn from the Pascal VOC2006 *testing* subset,⁷ which are high quality 640×480 RGB images with 8 bits per channel. For the experiments, we converted the 2600 images to grayscale by averaging the channels, and scaled the dynamic range to lie in the $[0, 1]$ interval. Similar results to those shown here are also obtained for other patch sizes.

5.1 Dictionary learning

For the experiments that follow, unless otherwise stated, we use a “global” overcomplete dictionary \mathbf{D} with $K = 4M = 256$ atoms trained on the full VOC2006 *training* subset using the method described in [47, 48], which seeks to minimize the following cost during training,⁸

$$\min_{\mathbf{D}, \mathbf{A}} \frac{1}{N} \sum_{j=1}^N \left\{ \|\mathbf{x}_j - \mathbf{D} \mathbf{a}_j\|_2^2 + \lambda \psi(\mathbf{a}_j) \right\} + \mu \|\mathbf{D}^T \mathbf{D}\|_F^2, \quad (\text{C.21})$$

where $\|\cdot\|_F$ denotes Frobenius norm. The additional term, $\mu \|\mathbf{D}^T \mathbf{D}\|_F^2$, encourages *incoherence* in the learned dictionary, that is, it forces the atoms to be as orthogonal as possible. Dictionaries with lower coherence are well known to have several theoretical advantages such as improved ability to recover sparse signals [5, 49], and faster and better convergence to the solution of the sparse coding problems (C.1) and (C.3) [50]. Furthermore, in [47] it was shown that adding incoherence leads to improvements in a variety of sparse modeling applications, including the ones discussed below.

We used MOE as the regularizer in (C.21), with $\lambda = 0.1$ and $\mu = 1$, both chosen empirically. See [1, 8, 47] for details on the optimization of (C.3) and (C.21).

⁷<http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html\#VOC2006>

⁸While we could have used off-the-shelf dictionaries such as DCT in order to test our universal sparse coding framework, it is important to use dictionaries that lead to the state-of-the-art results in order to show the additional potential improvement of our proposed regularizers.

5.2 MOE as a prior for sparse coding coefficients

We begin by comparing the performance of the Laplacian and MOE priors for fitting a single global distribution to the whole matrix \mathbf{A} . We compute \mathbf{A} using (C.1) with $\epsilon \approx 0$ and then, following the discussion in Section 4, restrict our study to the nonzero elements of \mathbf{A} .

The empirical distribution of \mathbf{A} is plotted in Figure C.3(a), along with the best fitting Laplacian, MOE, JOE, and a particularly good example of the conditional Jeffreys (CMOE) distributions.⁹ The MLE for the Laplacian fit is $\hat{\theta} = N_1 / \|\mathbf{A}\|_1 = 27.2$ (here N_1 is the number of nonzero elements in \mathbf{A}). For MOE, using (C.12), we obtained $\kappa = 2.8$ and $\beta = 0.07$. For JOE, $\theta_1 = 2.4$ and $\theta_2 = 371.4$. According to the discussion in Section 3.3, we used the value $\kappa = 2.8$ obtained using the method of moments for MOE as a hint for choosing $n_0 = 2$ ($\kappa_0 = n_0 + 1 = 3 \approx 2.8$), yielding $\beta_0 = 0.07$, which coincides with the β obtained using the method of moments. As observed in Figure C.3(a), in all cases the proposed mixture models fit the data better, significantly better for both Gamma-based mixtures, MOE and CMOE, and slightly better for JOE. This is further confirmed by the Kullback-Leibler divergence (KLD) obtained in each case. Note that JOE fails to significantly improve on the Laplacian mode due to the excessively large estimated range $[\theta_1, \theta_2]$. In this sense, it is clear that the JOE model is very sensitive to its hyper-parameters, and a better and more robust estimation would be needed for it to be useful in practice.

Given these results, hereafter we concentrate on the best case which is the MOE prior (which, as detailed above, can be derived from the conditional Jeffreys as well, thus representing both approaches).

From Figure C.1(e) we know that the optimal $\hat{\theta}$ varies locally across different regions, thus, we expect the mixture models to perform well also on a per-atom basis. This is confirmed in Figure C.3(b), where we show, for each row $\mathbf{a}^k, k = 1, \dots, K$, the difference in KLD

⁹To compute the empirical distribution, we quantized the elements of \mathbf{A} uniformly in steps of 2^{-8} , which for the amount of data available, gives us enough detail and at the same time reliable statistics for all the quantized values.

between the globally fitted MOE distribution and the best per-atom fitted MOE, the globally fitted Laplacian, and the per-atom fitted Laplacians respectively. As can be observed, the KLD obtained with the *global* MOE is significantly smaller than the global Laplacian in all cases, and even the *per-atom* Laplacians in most of the cases. This shows that MOE, with only two parameters (which can be easily estimated, as detailed in the text), is a much better model than K Laplacians (requiring K critical parameters) fitted specifically to the coefficients associated to each atom. Whether these modeling improvements have a practical impact is explored in the next experiments.

5.3 Recovery of noisy sparse signals

Here we compare the active set recovery properties of the MOE prior, with those of the ℓ_1 -based one, on data for which the sparsity assumption $|\mathcal{A}_j| \leq L$ holds exactly for all j . To this end, we obtain sparse approximations to each sample \mathbf{x}_j using the ℓ_0 -based Orthogonal Matching Pursuit algorithm (OMP) on \mathbf{D} [20], and record the resulting active sets \mathcal{A}_j as ground truth. The data is then contaminated with additive Gaussian noise of variance σ and the recovery is performed by solving (C.1) for \mathbf{A} with $\epsilon = CM\sigma^2$ and either the ℓ_1 or the MOE-based regularizer for $\psi(\cdot)$. We use $C = 1.32$, which is a standard value in denoising applications (see for example [9]).

For each sample j , we measure the error of each method in recovering the active set as the Hamming distance between the true and estimated support of the corresponding reconstruction coefficients. The accuracy of the method is then given as the percentage of the samples for which this error falls below a certain threshold T . Results are shown in Figure C.3(c) for $L = (5, 10)$ and $T = (2, 4)$ respectively, for various values of σ . Note the very significant improvement obtained with the proposed model.

Given the estimated active set \mathcal{A}_j , the estimated clean patch is obtained by projecting \mathbf{x}_j onto the subspace defined by the atoms that are active according to \mathcal{A}_j , using least squares

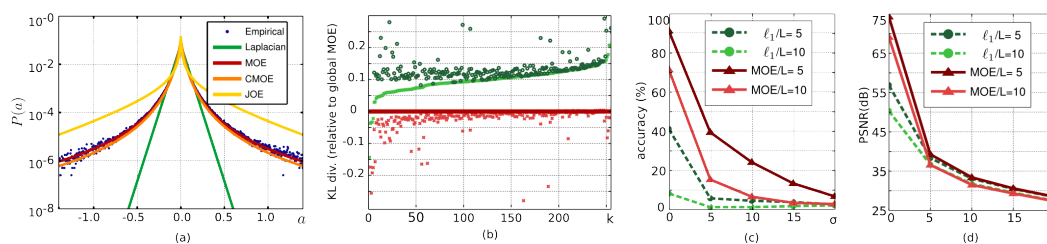


Figure C.3: (a) Empirical distribution of the coefficients in \mathbf{A} for image patches (blue dots), best fitting Laplacian (green), MOE (red), CMOE (orange) and JOE (yellow) distributions. The Laplacian (KLD=0.17 bits) is clearly not fitting the tails properly, and is not sufficiently peaked at zero either. The two models based on a Gamma prior, MOE (KLD=0.01 bits) and CMOE (KLD=0.01 bits), provide an almost perfect fit. The fitted JOE (KLD=0.14) is the most sharply peaked at 0, but does not fit the tails as tight as desired. As a reference, the entropy of the empirical distribution is $H = 3.00$ bits. (b) KLD for the best fitting global Laplacian (dark green), per-atom Laplacian (light green), global MOE (dark red) and per-atom MOE (light red), relative to the KLD between the globally fitted MOE distribution and the empirical distribution. The horizontal axis represents the indexes of each atom, $k = 1, \dots, K$, ordered according to the difference in KLD between the global MOE and the per-atom Laplacian model. Note how the global MOE outperforms both the global and per-atom Laplacian models in all but the first 4 cases. (c) active set recovery accuracy of ℓ_1 and MOE, as defined in Section 5.3, for $L = 5$ and $L = 10$, as a function of σ . The improvement of MOE over ℓ_1 is a factor of 5 to 9. (d) PSNR of the recovered sparse signals with respect to the true signals. In this case significant improvements can be observed at the high SNR range, specially for highly sparse ($L = 5$) signals. The performance of both methods is practically the same for $\sigma \geq 10$.

(which is the standard procedure for denoising once the active set is determined). We then measure the PSNR of the estimated patches with respect to the true ones. The results are shown in Figure C.3(d), again for various values of σ . As can be observed, the MOE-based recovery is significantly better, specially in the high SNR range. Notoriously, the more accurate active set recovery of MOE does not seem to improve the denoising performance in this case. However, as we will see next, it does make a difference when denoising real life signals, as well as for classification tasks.

5.4 Recovery of real signals with simulated noise

This experiment is an analogue to the previous one, when the data are the original natural image patches (without forcing exact sparsity). Since for this case the sparsity assumption is only approximate, and no ground truth is available for the active sets, we compare the different methods in terms of their denoising performance.

A critical strategy in image denoising is the use of overlapping patches, where for each pixel in the image a patch is extracted with that pixel as its center. The patches are denoised independently as M -dimensional signals and then recombined into the final denoised images by simple averaging. Although this consistently improves the final result in all cases, the improvement is very different depending on the method used to denoise the individual patches. Therefore, we now compare the denoising performance of each method at two levels: individual patches and final image.

To denoise each image, the global dictionary described in Section 5.1 is further adapted to the noisy image patches using (C.21) for a few iterations, and used to encode the noisy patches via (C.2) with $\epsilon = CM\sigma^2$. We repeated the experiment for two learning variants (ℓ_1 and MOE regularizers), and two coding variants ((C.2) with the regularizer used for learning, and ℓ_0 via OMP). The four variants were applied to the standard images Barbara, Boats, Lena, Man and Peppers, and the results summarized in Table C.1. We show sample



Figure C.4: Sample image denoising results. Top: Barbara, $\sigma = 30$. Bottom: Boats, $\sigma = 40$. From left to right: noisy, ℓ_1/OMP , ℓ_1/ℓ_1 , MOE/MOE. The reconstruction obtained with the proposed model is more accurate, as evidenced by a better reconstruction of the texture in Barbara, and sharp edges in Boats, and does not produce the artifacts seen in both the ℓ_1 and ℓ_0 reconstructions, which appear as black/white speckles all over Barbara, and ringing on the edges in Boats.

results in Figure C.4. Although the quantitative improvements seen in Table C.1 are small compared to ℓ_1 , there is a significant improvement at the visual level, as can be seen in Figure C.4. In all cases the PSNR obtained coincides or surpasses the ones reported in [1].¹⁰

5.5 Zooming

As an example of signal recovery in the absence of noise, we took the previous set of images, plus a particularly challenging one (Tools), and subsampled them to half each side. We then simulated a zooming effect by upsampling them and estimating each of the 75% missing pixels (see e.g., [51] and references therein). We use a technique similar to the one used in [52]. The image is first interpolated and then deconvoluted using a Wiener filter. The

¹⁰Note that in [1], the denoised image is finally blended with the noisy image using an empirical weight, providing an extra improvement to the final PSNR in some cases. The results in C.1 are already better without this extra step.

learning coding	$\sigma = 10$					$\sigma = 20$					$\sigma = 30$				
	ℓ_1		MOE		[1]	ℓ_1		MOE		[1]	ℓ_1		MOE		[1]
	ℓ_0	ℓ_1	ℓ_0	MOE		ℓ_0	ℓ_1	ℓ_0	MOE		ℓ_0	ℓ_1	ℓ_0	MOE	
barbara	30.4/ 34.4	31.2 /33.8	30.5/ 34.4	30.9/ 34.4	34.4	26.5/30.6	26.9/30.2	26.8/30.7	27.0 / 30.9	30.8	24.5/28.2	24.8/28.2	24.8/28.3	24.9 / 28.5	28.4
boat	30.4/33.7	30.9 /33.4	30.5/33.7	30.8/ 33.8	33.7	26.9/30.2	27.2/30.1	27.1/30.3	27.3 / 30.4	30.3	25.0/28.1	25.2/28.2	25.3/28.2	25.4 / 28.3	28.2
lena	31.8/35.5	32.4 /35.1	32.1/35.6	32.3/ 35.6	35.5	28.3/32.3	28.6/32.0	28.7/32.3	28.8 / 32.4	32.4	26.4/30.1	26.6/30.2	26.7/30.3	26.8 / 30.4	30.3
peppers	31.6/34.8	32.1 /34.6	31.8/ 34.9	32.0/ 34.9	34.8	28.3/31.9	28.7 /31.8	28.6/31.9	28.7 / 32.0	31.9	26.3/29.8	26.6/ 29.9	26.6/ 29.9	26.7 / 29.9	-
man	29.6/33.0	30.6 /32.9	29.7/33.0	30.2/ 33.1	32.8	25.8/28.8	26.3 /28.9	26.0/28.9	26.2/ 29.0	28.8	23.9/26.5	24.2 / 26.8	24.1/26.6	24.2 / 26.7	26.5
AVERAGE	30.7/34.2	31.4 /33.9	30.8/34.2	31.1/ 34.3	34.1	27.0/30.6	27.4/30.4	27.3/30.6	27.5 / 30.8	30.6	25.1/28.3	25.4/ 28.5	25.4/28.4	25.5 / 28.5	28.4

Table C.1: Denoising results: in each table, each column shows the denoising performance of a learning+coding combination. Results are shown in pairs, where the left number is the PSNR between the clean and recovered individual patches, and the right number is the PSNR between the clean and recovered images. Best results are in bold. The proposed MOE produces better final results over both the ℓ_0 and ℓ_1 ones in all cases, and at patch level for all $\sigma > 10$. Note that the average values reported are the PSNR of the average MSE, and not the PSNR average.

image	cubic	ℓ_0	ℓ_1	MOE
barbara	25.0	25.6	25.5	25.6
boat	28.9	29.8	29.8	29.9
lena	32.7	33.8	33.8	33.9
peppers	32.0	33.4	33.4	33.4
man	28.4	29.4	29.3	29.4
tools	21.0	22.3	22.2	22.3
AVER	22.8	24.0	24.0	24.1

Figure C.5: Zooming results. Left to right: summary, Tools image, detail of zooming results for the framed region, top to bottom and left to right: cubic, ℓ_0 , ℓ_1 , MOE. As can be seen, the MOE result is as sharp as ℓ_0 but produces less artifacts. This is reflected in the 0.1dB overall improvement obtained with MOE, as seen in the leftmost summary table.

deconvoluted image has artifacts that we treat as noise in the reconstruction. However, since there is no real noise, we do not perform averaging of the patches, using only the center pixel of $\hat{\mathbf{x}}_j$ to fill in the missing pixel at j . The results are summarized in Figure C.5, where we again observe that using MOE instead of ℓ_0 and ℓ_1 improves the results.

5.6 Classification with universal sparse models

In this section we apply our proposed universal models to a classification problem where each sample \mathbf{x}_j is to be assigned a class label $y_j = 1, \dots, c$, which serves as an index to the set of possible classes, $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_c\}$. We follow the procedure of [48], where the classifier assigns each sample \mathbf{x}_j by means of the maximum a posteriori criterion (C.5) with the term $-\log P(\mathbf{a})$ corresponding to the assumed prior, and the dictionaries representing each class are learned from training samples using (C.21) with the corresponding regularizer $\psi(\mathbf{a}) = -\log P(\mathbf{a})$. Each experiment is repeated for the baseline Laplacian model, implied in the ℓ_1 regularizer, and the universal model MOE, and the results are then compared. In this case we expect that the more accurate prior model for the coefficients will result in an improved likelihood estimation, which in turn should improve the accuracy of the system.

We begin with a classic texture classification problem, where patches have to be identified as belonging to one out of a number of possible textures. In this case we experimented with samples of $c = 2$ and $c = 3$ textures drawn at random from the Brodatz database,¹¹ the ones actually used shown in Figure C.6. In each case the experiment was repeated 10 times. In each repetition, a dictionary of $K = 300$ atoms was learned from all 16×16 patches of the leftmost halves of each sample texture. We then classified the patches from the rightmost halves of the texture samples. For the $c = 2$ we obtained an average error rate of 5.13% using ℓ_1 against 4.12% when using MOE, which represents a reduction of 20% in classification error. For $c = 3$ the average error rate obtained was 13.54% using

¹¹<http://www.ux.uis.no/~tranden/brodatz.html>

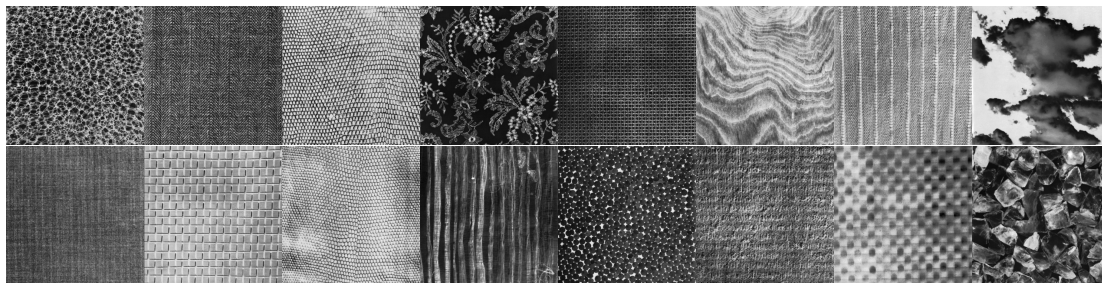


Figure C.6: Textures used in the texture classification example.

ℓ_1 and 11.48% using MOE, which is 15% lower. Thus, using the universal model instead of ℓ_1 yields a significant improvement in this case (see for example [8] for other results in classification of Brodatz textures).

The second sample problem presented is the Graz'02 bike detection problem,¹² where each pixel of each testing image has to be classified as either background or as part of a bike. In the Graz'02 dataset, each of the pixels can belong to one of two classes: bike or background. On each of the training images (which by convention are the first 150 even-numbered images), we are given a mask that tells us whether each pixel belongs to a bike or to the background. We then train a dictionary for bike patches and another for background patches. Patches that contain pixels from both classes are assigned to the class corresponding to the majority of their pixels.

In Figure C.7 we show the *precision vs. recall curves* obtained with the detection framework when either the ℓ_1 or the MOE regularizers were used in the system. As can be seen, the MOE-based model outperforms the ℓ_1 in this classification task as well, giving a better precision for all recall values.

In the above experiments, the parameters for the ℓ_1 prior (λ), the MOE model (λ_{MOE}) and the incoherence term (μ) were all adjusted by cross validation. The only exception is the MOE parameter β , which was chosen based on the fitting experiment as $\beta = 0.07$.

¹²<http://lear.inrialpes.fr/people/marszalek/data/ig02/>

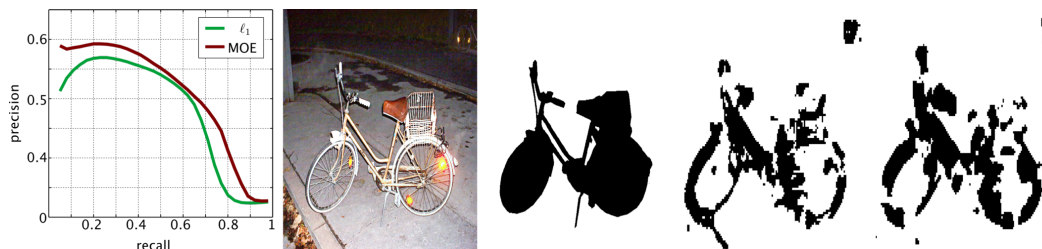


Figure C.7: Classification results. Left to right: precision vs. recall curve, a sample image from the Graz'02 dataset, its ground truth, and the corresponding estimated maps obtained with ℓ_1 and MOE for a fixed threshold. The precision vs. recall curve shows that the mixture model gives a better precision in all cases. In the example, the classification obtained with MOE yields less false positives and more true positives than the one obtained with ℓ_1 .

6 Concluding remarks

A framework for designing sparse modeling priors was introduced in this work, using tools from universal coding, which formalizes sparse coding and modeling from a MDL perspective. The priors obtained lead to models with both theoretical and practical advantages over the traditional ℓ_0 and ℓ_1 -based ones. In all derived cases, the designed non-convex problems are suitable to be efficiently (approximately) solved via a few iterations of (weighted) ℓ_1 subproblems. We also showed that these priors are able to fit the empirical distribution of sparse codes of image patches significantly better than the traditional IID Laplacian model, and even the non-identically distributed independent Laplacian model where a different Laplacian parameter is adjusted to the coefficients associated to each atom, thus showing the flexibility and accuracy of these proposed models. The additional flexibility, furthermore, comes at a small cost of only 2 parameters that can be easily and efficiently tuned (either (κ, β) in the MOE model, or (θ_1, θ_2) in the JOE model), instead of K (dictionary size), as in weighted ℓ_1 models. The additional accuracy of the proposed models was shown to have significant practical impact in active set recovery of sparse signals, image denoising, and classification applications. Compared to the Bayesian approach, we avoid the potential burden of solving several sampled sparse problems, or being forced to use a conjugate

prior for computational reasons (although in our case, *a fortiori*, the conjugate prior does provide us with a good model). Overall, as demonstrated in this paper, the introduction of information theory tools can lead to formally addressing critical aspects of sparse modeling.

Future work in this direction includes the design of priors that take into account the nonzero mass at $a = 0$ that appears in overcomplete models, and online learning of the model parameters from noisy data, following for example the technique in [46].

Acknowledgments

Work partially supported by NGA, ONR, ARO, NSF, NSSEFF, and FUNDACIBA-ANTEL. We wish to thank Julien Mairal for providing us with his fast sparse modeling toolbox, SPAMS.¹³ We also thank Federico Lecumberry for his participation on the incoherent dictionary learning method, and helpful comments.

7 Appendix: Derivation of the MOE model

In this case we have $P(a|\theta) = \theta e^{-\theta a}$ and $w(\theta|\kappa, \beta) = \frac{1}{\Gamma(\kappa)} \theta^{\kappa-1} \beta^\kappa e^{-\beta\theta}$, which, when plugged into (C.9), gives

$$Q(a|\beta, \kappa) = \int_{\theta=0}^{\infty} \theta e^{-\theta a} \frac{1}{\Gamma(\kappa)} \theta^{\kappa-1} \beta^\kappa e^{-\beta\theta} d\theta = \frac{\beta^\kappa}{\Gamma(\kappa)} \int_{\theta=0}^{\infty} e^{-\theta(a+\beta)} \theta^\kappa d\theta.$$

After the change of variables $u := (a + \beta)\theta$ ($u(0) = 0$, $u(\infty) = \infty$), the integral can be written as

$$\begin{aligned} Q(a|\beta, \kappa) &= \frac{\beta^\kappa}{\Gamma(\kappa)} \int_{\theta=0}^{\infty} e^{-u} \left(\frac{u}{a + \beta} \right)^\kappa \frac{du}{a + \beta} = \frac{\beta^\kappa}{\Gamma(\kappa)} (a + \beta)^{-(\kappa+1)} \int_{\theta=0}^{\infty} e^{-u} u^\kappa du \\ &= \frac{\beta^\kappa}{\Gamma(\kappa)} (a + \beta)^{-(\kappa+1)} \Gamma(\kappa + 1) = \frac{\beta^\kappa}{\Gamma(\kappa)} (a + \beta)^{-(\kappa+1)} \kappa \Gamma(\kappa), \end{aligned}$$

¹³<http://www.di.ens.fr/willow/SPAMS/>

obtaining $Q(a|\beta, \kappa) = \kappa\beta^\kappa(a + \beta)^{-(\kappa+1)}$, since the integral on the second line is precisely the definition of $\Gamma(\kappa + 1)$. The symmetrization is obtained by substituting a by $|a|$ and dividing the normalization constant by two, $Q(|a||\beta, \kappa) = 0.5\kappa\beta^\kappa(|a| + \beta)^{-(\kappa+1)}$.

The mean of the MOE distribution (which is defined only for $\kappa > 1$) can be easily computed using integration by parts,

$$\mu(\beta, \kappa) = \kappa\beta^\kappa \int_0^\infty \frac{u}{(u + \beta)^{(\kappa+1)}} du = \kappa\beta \left[-\frac{u}{\kappa(u + \beta)^\kappa} \Big|_0^\infty + \frac{1}{\kappa} \int_0^\infty \frac{du}{(u + \beta)^\kappa} \right] = \frac{\beta}{\kappa - 1}.$$

In the same way, it is easy to see that the non-central moments of order i are $\mu_i = \frac{\beta}{\binom{\kappa-1}{i}}$.

The MLE estimates of κ and β can be obtained using any nonlinear optimization technique such as Newton method, using for example the estimates obtained with the method of moments as a starting point. In practice, however, we have not observed any significant improvement in using the MLE estimates over the moments-based ones.

Expected approximation error in cost function

As mentioned in the optimization section, the LLA approximates the MOE regularizer as a weighted ℓ_1 . Here we develop an expression for the expected error between the true function and the approximate convex one, where the expectation is taken (naturally) with respect to the MOE distribution. Given the value of the current iterate $a^{(t)} = a_0$, (assumed

positive, since the function and its approximation are symmetric), the approximated regularizer is $\psi^{(t)}(a) = \log(a_0 + \beta) + \frac{1}{|a_0| + \beta}(a - a_0)$. We have

$$\begin{aligned} E_{a \sim \text{MOE}(\kappa, \beta)} [\psi^{(t)}(a) - \psi(a)] &= \\ &= \int_0^\infty \frac{\kappa \beta^\kappa}{(a + \kappa)^{\kappa+1}} \left[\log(|a_0 + \beta) + \frac{1}{a_0 + \beta}(a - a_0) - \log(a + \beta) \right] da \\ &= \log(a_0 + \beta) + \frac{a_0}{a_0 + \beta} + \frac{\kappa \beta^\kappa}{a_0 + \beta} \int_0^\infty \frac{a}{(a + \beta)^{\kappa+1}} da - \kappa \beta^\kappa \int_0^\infty \frac{\log(a + \beta)}{(a + \beta)^{\kappa+1}} da \\ &= \log(a_0 + \beta) + \frac{a_0}{a_0 + \beta} + \frac{\beta}{(a_0 + \beta)(\kappa - 1)} - \log \beta - \frac{1}{\kappa}. \end{aligned}$$

Derivation of the constrained Jeffreys (JOE) model

In the case of the exponential distribution, the Fisher Information Matrix in (C.15) evaluates to

$$I(\theta) = \left\{ E_{P(\cdot|\tilde{\theta})} \left[\frac{\partial^2}{\partial \tilde{\theta}^2} (-\log \theta + \theta \log a) \right] \right\} \Big|_{\tilde{\theta}=\theta} = \left\{ E_{P(\cdot|\tilde{\theta})} \left[\frac{1}{\tilde{\theta}^2} \right] \right\} \Big|_{\tilde{\theta}=\theta} = \frac{1}{\theta^2}.$$

By plugging this result into (C.14) with $\Theta = [\theta_1, \theta_2]$, $0 < \theta_1 < \theta_2 < \infty$ we obtain $w(\theta) = \frac{1}{\ln(\theta_2/\theta_1)} \frac{1}{\theta}$. We now derive the (one-sided) JOE probability density function by plugging this $w(\theta)$ in (C.9),

$$Q(a) = \int_{\theta_1}^{\theta_2} \theta e^{-\theta a} \frac{1}{\ln(\theta_2/\theta_1)} \frac{d\theta}{\theta} = \frac{1}{\ln(\theta_2/\theta_1)} \int_{\theta_1}^{\theta_2} e^{-\theta a} d\theta = \frac{1}{\ln(\theta_2/\theta_1)} \frac{1}{a} (e^{-\theta_1 a} - e^{-\theta_2 a}).$$

Although $Q(a)$ cannot be evaluated at $a = 0$, the limit for $a \rightarrow 0$ exists and is finite, so we can just define $Q(0)$ as this limit, which is

$$\lim_{a \rightarrow 0} Q(a) = \lim_{a \rightarrow 0} \frac{1}{\ln(\theta_2/\theta_1) a} [1 - \theta_1 a + o(a^2) - (1 - \theta_2 a + o(a^2))] = \frac{\theta_2 - \theta_1}{\ln(\theta_2/\theta_1)}.$$

Again, if desired, parameter estimation can be done for example using maximum likelihood (via nonlinear optimization), or using the method of moments. However, in this case, the method of moments does not provide a closed form solution for (θ_1, θ_2) . The non-central moments of order i are

$$\begin{aligned}\mu_i &= \int_0^{\infty+} \frac{a^i}{\ln(\theta_2/\theta_1)} \frac{1}{a} [e^{-\theta_1 a} - e^{-\theta_2 a}] da \\ &= \frac{1}{\ln(\theta_2/\theta_1)} \left\{ \int_0^{\infty+} a^{i-1} e^{-\theta_1 a} da - \int_0^{\infty+} a^{i-1} e^{-\theta_2 a} da \right\}.\end{aligned}\quad (\text{C.22})$$

For $i = 1$, both integrals in (C.22) are trivially evaluated, yielding $\mu_1 = \frac{1}{\ln(\theta_2/\theta_1)}(\theta_1^{-1} - \theta_2^{-1})$.

For $i > 1$, these integrals can be solved using integration by parts:

$$\begin{aligned}\mu_i^+ &= \int_0^{\infty+} a^{i-1} e^{-\theta_1 a} da = a^{i-1} \frac{1}{(-\theta_1)} e^{-\theta_1 a} \Big|_0^{\infty+} - \frac{1}{(-\theta_1)} (i-1) \int_0^{\infty+} a^{i-2} e^{-\theta_1 a} da \\ \mu_i^- &= \int_0^{\infty+} a^{i-1} e^{-\theta_2 a} da = a^{i-1} \frac{1}{(-\theta_2)} e^{-\theta_2 a} \Big|_0^{\infty+} - \frac{1}{(-\theta_2)} (i-1) \int_0^{\infty+} a^{i-2} e^{-\theta_2 a} da,\end{aligned}$$

where the first term in the right hand side of both equations evaluates to 0 for $i > 1$. Therefore, for $i > 1$ we obtain the recursions $\mu_i^+ = \frac{i-1}{\theta_1} \mu_{i-1}^+$, $\mu_i^- = \frac{i-1}{\theta_2} \mu_{i-1}^-$, which, combined with the result for $i = 1$, give the final expression for all the moments of order $i > 0$

$$\mu_i = \frac{(i-1)!}{\ln(\theta_2/\theta_1)} \left(\frac{1}{\theta_1^i} - \frac{1}{\theta_2^i} \right), \quad i = 1, 2, \dots$$

In particular, for $i = 1$ and $i = 2$ we have

$$\theta_1 = (\ln(\theta_2/\theta_1)\mu_1 + \theta_2^{-1})^{-1} \quad \theta_2 = (\ln(\theta_2/\theta_1)\mu_2 + \theta_1^{-2})^{-1},$$

which, when combined, give us

$$\theta_1 = \frac{2\mu_1}{\mu_2 + \ln(\theta_2/\theta_1)\mu_1^2}, \quad \theta_2 = \frac{2\mu_1}{\mu_2 - \ln(\theta_2/\theta_1)\mu_1^2}. \quad (\text{C.23})$$

One possibility is to solve the nonlinear equation $\theta_2/\theta_1 = \frac{\mu_2 + \ln(\theta_2/\theta_1)\mu_1^2}{\mu_2 - \ln(\theta_2/\theta_1)\mu_1^2}$ for $u = \theta_1/\theta_2$ by finding the roots of the nonlinear equation $u = \frac{\mu_2 + \ln u \mu_1^2}{\mu_2 - \ln u \mu_1^2}$ and choosing one of them based on some side information. Another possibility is to simply fix the ratio θ_2/θ_1 beforehand and solve for θ_1 and θ_2 using (C.23).

Derivation of the conditional Jeffreys (CMOE) model

The conditional Jeffreys method defines a proper prior $w(\theta)$ by assuming that n_0 samples from the data to be modeled \mathbf{a} were already observed. Plugging the Fisher information for the exponential distribution, $I(\theta) = \theta^{-2}$, into (C.18) we obtain

$$w(\theta) = \frac{P(a^{n_0}|\theta)\theta^{-1}}{\int_{\Theta} P(a^{n_0}|\xi)\xi^{-1}d\xi} = \frac{(\prod_{j=1}^{n_0} \theta e^{-\theta a_j})\theta^{-1}}{\int_0^{+\infty} (\prod_{j=1}^{n_0} \xi e^{-\xi a_j})\xi^{-1}d\xi} = \frac{\theta^{n_0-1} e^{-\theta \sum_{j=1}^{n_0} a_j}}{\int_0^{+\infty} \xi^{n_0-1} e^{-\xi \sum_{j=1}^{n_0} a_j} d\xi}.$$

Denoting $S_0 = \sum_{j=1}^{n_0} a_j$ and performing the change of variables $u := S_0 \xi$ we obtain

$$w(\theta) = \frac{\theta^{n_0-1} e^{-S_0 \theta}}{S_0^{-n_0} \int_0^{+\infty} u^{n_0-1} e^{-u} du} = \frac{S_0^{n_0} \theta^{n_0-1} e^{-S_0 \theta}}{\Gamma(n_0)},$$

where the last equation derives from the definition of the Gamma function, $\Gamma(n_0)$. We see that the resulting prior $w(\theta)$ is a Gamma distribution $\text{Gamma}(\kappa_0, \beta_0)$ with $\kappa_0 = n_0$ and $\beta_0 = S_0 = \sum_{j=1}^{n_0} a_j$.

8 Supplementary material

Proposition 2. Let \mathbf{D} be a dictionary of normalized atoms, that is, $\|\mathbf{d}_k\|_2 = 1, k = 1, \dots, p$. Let $\mathbf{G} = \mathbf{D}^T \mathbf{D}$ be its Gram matrix. Denote by $\rho(\mathbf{D})$ the spectral norm of \mathbf{D} , that is, $\rho(\mathbf{D}) = \sqrt{\|\mathbf{G}\|_2}$. Define the cumulative mutual coherence of \mathbf{D} as

$$\bar{\mu}(\mathbf{D}) = \max_k \sum_{r \neq k} |\mathbf{d}_r^T \mathbf{d}_k|.$$

Then we have that

$$\rho(\mathbf{D}) \leq \sqrt{1 + \bar{\mu}(\mathbf{D})}.$$

Proof. Let $\lambda(\mathbf{G})$ denote the set of eigenvalues of \mathbf{G} (which are positive by the definition of \mathbf{G}). By definition, we have $\|\mathbf{G}\|_2 := \lambda_{\max} := \max\{\lambda(\mathbf{G})\}$. Using the Gerschgorin Circle Theorem [53, pp. 320–321], we have that $\lambda(\mathbf{G}) \subseteq \cup_{k=1}^K \mathcal{C}_k$, where

$$\mathcal{C}_k = \left\{ y : |y - g_{kk}| \leq \sum_{r \neq k} |g_{kr}| \right\} \quad (\text{C.24})$$

and $g_{kr} = \mathbf{d}_k^T \mathbf{d}_r$ are the elements of \mathbf{G} . Since the atoms are normalized, we have that $g_{kk} = 1, \forall k$. We also have by definition of $\bar{\mu}(\mathbf{D})$ that $\sum_{r \neq k} |g_{kr}| \leq \bar{\mu}(\mathbf{D})$. Plugging these two values in (C.24) we obtain that $\lambda_{\max} \leq 1 + \bar{\mu}(\mathbf{D})$. Finally, using the definition of $\rho(\mathbf{D})$ we conclude that

$$\rho(\mathbf{D}) = \sqrt{\lambda_{\max}} \leq \sqrt{1 + \bar{\mu}(\mathbf{D})}.$$

□

References

- [1] M. Aharon, M. Elad, and A. Bruckstein. The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Trans. SP*, 54(11):4311–4322, Nov. 2006.
- [2] K. Engan, S. Aase, and J. Husoy. Multi-frame compression: Theory and design. *Signal Processing*, 80(10):2121–2140, Oct. 2000.
- [3] B. Olshausen and D. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.
- [4] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.
- [5] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. on Pure and Applied Mathematics*, 57:1413–1457, 2004.
- [6] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.
- [7] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Trans. PAMI*, 27(6):957–968, 2005.
- [8] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Adv. NIPS*, volume 21, Dec. 2009.
- [9] J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. *SIAM MMS*, 7(1):214–241, April 2008.

- [10] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML*, pages 759–766, June 2007.
- [11] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, Feb. 2009.
- [12] R. Giryes, Y. Eldar, and M. Elad. Automatic parameter setting for iterative shrinkage methods. In *IEEE 25-th Convention of Electronics and Electrical Engineers in Israel (IEEEI'08)*, Dec. 2008.
- [13] M. Figueiredo. Adaptive sparseness using Jeffreys prior. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Adv. NIPS*, pages 697–704. MIT Press, Dec. 2001.
- [14] H. Zou. The adaptive LASSO and its oracle properties. *Journal Am. Stat. Assoc.*, 101:1418–1429, 2006.
- [15] J. Rissanen. Universal coding, information, prediction and estimation. *IEEE Trans. IT*, 30(4), July 1984.
- [16] R. Coifman and M. Wickenhauser. Entropy-based algorithms for best basis selection. *IEEE Trans. IT*, 38:713–718, 1992.
- [17] P. Moulin and J. Liu. Analysis of multiresolution image denoising schemes using generalized-Gaussian and complexity priors. *IEEE Trans. IT*, April 1999.
- [18] N. Saito. Simultaneous noise suppression and signal compression using a library of orthonormal bases and the MDL criterion. In E. Foufoula-Georgiou and P. Kumar, editors, *Wavelets in Geophysics*, pages 299–324. New York: Academic, 1994.
- [19] E. J. Candès. Compressive sampling. *Proc. of the International Congress of Mathematicians*, 3, Aug. 2006.

- [20] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Trans. SP*, 41(12):3397–3415, 1993.
- [21] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2 edition, Feb. 2009.
- [22] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.
- [23] E. Lam and J. Goodman. A mathematical analysis of the DCT coefficient distributions for images. *IEEE Trans. IP*, 9(10):1661–1666, 2000.
- [24] T. Cover and J. Thomas. *Elements of information theory*. John Wiley and Sons, Inc., 2 edition, 2006.
- [25] J. Rissanen. *Stochastic complexity in statistical inquiry*. Singapore: World Scientific, 1992.
- [26] M. Everingham, A. Zisserman, C. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. <http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf>.
- [27] S. Ji, Y. Xue, and L. Carin. Bayesian compressive sensing. *IEEE Trans. SP*, 56(6):2346–2356, 2008.
- [28] M. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning*, 1:211–244, 2001.
- [29] D. Wipf and B. Rao. An empirical bayesian strategy for solving the simultaneous sparse approximation problem. *IEEE Trans. IP*, 55(7-2):3704–3716, 2007.
- [30] D. Wipf, J. Palmer, and B. Rao. Perspectives on sparse bayesian learning. In *Adv. NIPS*, Dec. 2003.

- [31] N. Merhav and M. Feder. Universal prediction. *IEEE Trans. IT*, 44(6):2124–2147, Oct. 1998.
- [32] Y. Shtarkov. Universal sequential coding of single messages. *Probl. Inform. Transm.*, 23(3):3–17, July 1987.
- [33] P. Grünwald. *The Minimum Description Length Principle*. MIT Press, June 2007.
- [34] G. Schwartz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.
- [35] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal Am. Stat. Assoc.*, 96(456):1348–1360, Dec. 2001.
- [36] E. J. Candès, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *J. Fourier Anal. Appl.*, 14(5):877–905, Dec. 2008.
- [37] R. Saab, R. Chartrand, and O. Yilmaz. Stable sparse approximation via nonconvex optimization. In *ICASSP*, April 2008.
- [38] S. Foucart and M. Lai. Sparsest solutions of underdetermined linear systems via ℓ_q -minimization for $0 < q \leq 1$. *Applied and Computational Harmonic Analysis*, 3(26):395–407, 2009.
- [39] J. Trzasko and A. Manduca. Relaxed conditions for sparse signal recovery with general concave priors. *IEEE Trans. SP*, 57(11):4347–4354, 2009.
- [40] R. Chartrand. Fast algorithms for nonconvex compressive sensing: MRI reconstruction from very few data. In *IEEE ISBI*, June 2009.
- [41] J. Trzasko and A. Manduca. Highly undersampled magnetic resonance image reconstruction via homotopic ℓ_0 -minimization. *IEEE Trans. MI*, 28(1):106–121, Jan. 2009.

- [42] J. Bernardo and A. Smith. *Bayesian Theory*. Wiley, 1994.
- [43] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Trans. IT*, 44(6):2743–2760, 1998.
- [44] H. Zou and R. Li. One-step sparse estimates in nonconcave penalized likelihood models. *Annals of Statistics*, 36(4):1509–1533, 2008.
- [45] G. Gasso, A. Rakotomamonjy, and S. Canu. Recovering sparse signals with non-convex penalties and DC programming. *IEEE Trans. SP*, 57(12):4686–4698, 2009.
- [46] G. Motta, E. Ordentlich, I. Ramirez, G. Seroussi, and M. Weinberger. The DUDE framework for grayscale image denoising. Technical report, HP laboratories, 2009. <http://www.hpl.hp.com/techreports/2009/HPL-2009-252.html>.
- [47] I. Ramirez, F. Lecumberry, and G. Sapiro. Universal priors for sparse modeling. In *CAMSAP*, Dec. 2009.
- [48] I. Ramírez, P. Sprechmann, and G. Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *CVPR*, June 2010.
- [49] J. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. IT*, 50(10):2231–2242, Oct. 2004.
- [50] M. Elad. Optimized projections for compressed-sensing. *IEEE Trans. SP*, 55(12):5695–5702, Dec. 2007.
- [51] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity. Preprint arXiv:1006.3056.
- [52] R. Neelamani, H. Choi, and R. Baraniuk. Forward: Fourier-wavelet regularized deconvolution for ill-conditioned systems. *IEEE Trans. SP*, 52(2):418–433, 2004.

- [53] G. Golub and C. van Loan. *Matrix Computations*. JHU Press, 3rd edition, 1996.

D

An MDL framework for sparse coding and dictionary learning

Ignacio Ramírez and Guillermo Sapiro

Department of Electrical and Computer Engineering, University of Minnesota.

The power of sparse signal modeling with learned over-complete dictionaries has been demonstrated in a variety of applications and fields, from signal processing to statistical inference and machine learning. However, the statistical properties of these models, such as under-fitting or over-fitting *given* sets of data, are still not well characterized in the literature. As a result, the success of sparse modeling depends on hand-tuning critical parameters for each data and application. This work aims at addressing this by providing a practical and objective characterization of sparse models by means of the Minimum Description Length (MDL) principle – a well established information-theoretic approach to model selection in statistical inference. The resulting framework derives a family of efficient sparse coding and dictionary learning algorithms which, by virtue of the MDL principle, are completely parameter free. Furthermore, such framework allows to incorporate additional prior information to existing models, such as Markovian dependencies, or to define completely new problem formulations, including in the matrix analysis area, in a natural way. These virtues will be demonstrated with parameter-free algorithms for the classic image denoising and classification problems, and for low-rank matrix recovery in video applications.

1 Introduction

A *sparse model* is one in which signals of a given type $\mathbf{y} \in \mathbb{R}^m$ can be represented accurately as sparse linear combinations of the columns (atoms) of a learned dictionary $\mathbf{D} \in \mathbb{R}^{m \times p}$,

$\mathbf{y} = \mathbf{D}\mathbf{a} + \mathbf{e}$, where by accurate we mean that $\|\mathbf{e}\| \ll \|\mathbf{y}\|$ (in some norm), and by sparse we mean that the number of non-zero elements in \mathbf{a} , denoted by $\|\mathbf{a}\|_0$, is small compared to its dimension p . These concepts will be formalized in the next section.

Such models, especially when \mathbf{D} is learned from training samples, are by now a well established tool in a variety of fields and applications, see [1, 2, 3] for recent reviews.

When sparsity is a modeling device and not an hypothesis about the nature of the analyzed signals, parameters such as the *desired* sparsity in the solutions, or the size p of the dictionaries to be learned, play a critical role in the effectiveness of sparse models for the data and tasks at hand. However, lacking theoretical guidelines for such parameters, published applications based on learned sparse models often rely on either cross-validation or ad-hoc methods for determining such critical parameters (an exception for example being the Bayesian approach, e.g., [4]). Clearly, such techniques can be impractical and/or ineffective in many cases. This in turn hinders the further application of such models to new types of data and applications, or their evolution into different, possibly more sophisticated, models.

At the bottom of the aforementioned problem lie fundamental questions such as: *How rich or complex is a sparse model? How does this depend on the required sparsity of the solutions, or the size of the dictionaries? What is the best model for a given data class and a given task?*

The general problem of answering such questions and, in particular, the latter, is known as *model selection*. Popular model selection techniques such as Akaike's Information Criterion (AIC) [5], Bayes Information Criterion (BIC) [6], and the Minimum Description Length principle (MDL) [7, 8, 9], work by building a cost function which balances a measure of *goodness of fit* with one of *model complexity*, and search for a model that minimizes such cost. In this sense, these tools can be regarded as practical implementations of the Occam's razor principle, which states that, given two (equally accurate) descriptions for a given

phenomenon, the simpler one is usually the best.

In the Minimum Description Length principle, given a family or model class \mathcal{M} of candidate models indexed by a parameter M , and a data sample \mathbf{y} , the best model $\hat{M} \in \mathcal{M}$ is the one that can be used to describe \mathbf{y} completely (including the parameters M themselves) with the fewest number of bits,

$$\hat{M} = \arg \min_{M \in \mathcal{M}} L(\mathbf{y}, M), \quad (\text{D.1})$$

where $L(\mathbf{y}, M)$ is a *codelength assignment function* which defines the theoretical codelength required to describe (\mathbf{y}, M) *uniquely*, and which is a key component of any MDL-based framework. The underlying idea of MDL is that *compressibility is a good indirect way of measuring the ability of a model to capture regularity from the data*. Common practice in MDL uses the *Ideal Shannon Codelength Assignment* [10, Chapter 5] to define $L(\mathbf{y}, M)$ in terms of a *probability assignment* $P(\mathbf{y}, M)$ as $L(\mathbf{y}, M) = -\log P(\mathbf{y}, M)$ (all logarithms will be assumed on base 2 hereafter). In this way, the problem of choosing $L(\cdot)$ becomes one of choosing a suitable probability model for (\mathbf{y}, M) . Note here how MDL considers probability models not as a statement about the true nature of the data, but only as a modeling tool. If we now write $P(\mathbf{y}, M) = P(\mathbf{y}|M)P(M)$, we obtain the more familiar *penalized likelihood* form,

$$\hat{M} = \arg \min_{M \in \mathcal{M}} -\log P(\mathbf{y}|M) - \log P(M), \quad (\text{D.2})$$

with $-\log P(M)$ representing the model complexity, or *model cost*, term.

The use of MDL for sparse signal modeling has been explored for example in the context of wavelet-based denoising (where $M = \mathbf{a} \in \mathbb{R}^m$, $p = m$ and $\mathbf{D} \in \mathbb{R}^{m \times m}$ is *fixed*) of images corrupted by additive white Gaussian noise (AWGN) [11, 12, 13, 14, 15]. In [11, 12, 13], the data is described using (D.2) with $-\log P(\mathbf{y}|\mathbf{a})$ assumed to be *solely due to noise*, and an

$L(\mathbf{a})$ term which exploits sparsity,

$$\hat{\mathbf{a}} = \arg \min_{\mathbf{a} \in \mathcal{M}} \frac{1}{2\sigma_e^2} \|\mathbf{y} - \mathbf{D}\mathbf{a}\|_2^2 + L(\mathbf{a}). \quad (\text{D.3})$$

Here the first term corresponds to the ideal codelength, up to a constant, of an IID Gaussian sequence of zero mean and known variance σ_e^2 . The difference between [11, 12, 13] lies in the definition of $L(\mathbf{a})$. The line of work [14, 15] follows the modern MDL approach by using sophisticated tools from coding theory, the so called *one-part universal codes*, which encodes (\mathbf{y}, \mathbf{a}) jointly, and reduces the arbitrariness in defining $L(\mathbf{a})$. However, such tools can only be applied for certain choices of $P(\mathbf{y}|\mathbf{a})$ and $P(\mathbf{a})$. In the case of [14, 15], the choice is to use continuous Gaussian models for both. As Gaussian models are *not well suited to the typically observed statistical properties of such data*, the performance of the resulting denoisers for example is very poor compared to the current state-of-the-art.

The present work extends and/or improves on the aforementioned work in the following ways:¹

- MDL-based sparse coding is extended to the case of *non-orthonormal, possibly over-complete and learned dictionaries* \mathbf{D} . As we will see in Section 5, this extension, critical to deal with modern, very successful sparse modeling approaches, poses not only new design problems but also significant computational challenges compared to the orthonormal case.
- Efficient codelengths (probability distributions) for the different components to encode (error, coefficients, dictionary) are obtained by *applying universal coding schemes to priors that are suited to the typically observed statistics of such data*.
- As a particular point of the above item, systematic model-fit deviations are naturally

¹This paper extends preliminary results reported in [16]. In particular, new dictionary learning algorithms are developed which include ℓ_1 atom regularization, forward and backward dictionary size adaptation. We also develop a new model for the low-rank matrix approximation problem.

taken into account in $P(\mathbf{y}|\mathbf{a})$. The resulting fitting terms fall into the category of robust estimators (see [17]), thus marrying robust statistics with information theory and with sparse modeling (dictionary learning).

- We comply with the basic MDL sanity check, meaning, that *the theoretical codelengths obtained are smaller than a “raw” description of the data*. We do so by including quantization in our models, and treating its effect rigorously.
- Dictionary learning within the MDL framework allows us to *optimize both the number of atoms p , as well as their structure*, resulting in a natural and objective form of regularization for \mathbf{D} .
- Structure is naturally added to the sparse models in the form of Markovian dependencies between adjacent data samples. We also show an extension of the model to the problem of low-rank matrix completion.

As a result of the above features, we obtain for the first time an MDL-based, parameter-free framework for signal modeling that is able to yield state-of-the-art results.

At the theoretical level, this brings us a step closer to the fundamental understanding of *learned* sparse models and brings a different perspective, that of MDL, into the sparse modeling world.

The remainder of this paper is organized as follows. Sparse models, and the associated notation, are described in detail in Section 2. Section 3 introduces MDL, and its application to sparse models. In Section 4 we present the probability models used to assign codelengths to different parts of the encoded data, while sections 5 and 6 describe the actual sparse coding and dictionary learning algorithms developed. Experimental results follow in Section 7, and the paper is concluded in Section 8.

2 Background on sparse modeling

Assume we are given n m -dimensional data samples ordered as columns of a matrix $\mathbf{Y} = [\mathbf{y}_1 | \mathbf{y}_2 | \dots | \mathbf{y}_n] \in \mathbb{R}^{m \times n}$. Consider a linear model for \mathbf{Y} , $\mathbf{Y} = \mathbf{D}\mathbf{A} + \mathbf{E}$, where $\mathbf{D} = [\mathbf{d}_1 | \mathbf{d}_2 | \dots | \mathbf{d}_p]$ is an $m \times p$ dictionary consisting of p atoms, $\mathbf{A} = [\mathbf{a}_1 | \mathbf{a}_2 | \dots | \mathbf{a}_n] \in \mathbb{R}^{p \times n}$ is a matrix of coefficients where each j -th column \mathbf{a}_j specifies the linear combination of columns of \mathbf{D} that approximates \mathbf{y}_j , and $\mathbf{E} = [\mathbf{e}_1 | \mathbf{e}_2 | \dots | \mathbf{e}_n] \in \mathbb{R}^{m \times n}$ is a matrix of approximation errors.

We define the support, or active set, of a vector $\mathbf{a} \in \mathbb{R}^p$ as $\text{supp}(\mathbf{a}) = \{k : \mathbf{a}_k \neq 0\}$. Let $\Gamma = \text{supp}(\mathbf{a})$. We also represent the support of \mathbf{a} as a binary vector $\mathbf{z} \in \{0, 1\}^p$ such that $z_i = 1$ for $i \in \Gamma$, and 0 otherwise. We refer to the sub-vector in $\mathbb{R}^{|\Gamma|}$ of non-zero elements of \mathbf{a} as either $\mathbf{a}_{[\Gamma]}$ or $\mathbf{a}_{[\mathbf{z}]}$. Both conventions are extended to refer to sets of columns of matrices, for example, $\mathbf{D}_{[\Gamma]}$ is the matrix formed by the $|\Gamma|$ columns of \mathbf{D} indexed by Γ . We will use the pseudo-norm $\|\mathbf{a}\|_0 := |\Gamma| = \sum \mathbf{z}$ to denote the number of non-zero elements of \mathbf{a} . We say that the model is *sparse* if we can achieve $\|\mathbf{e}_j\|_2 \ll \|\mathbf{y}_j\|_2$ and $\|\mathbf{a}\|_0 \ll p$ simultaneously for all or most $j = 1, \dots, n$.

The result of quantizing a real-valued variable y to precision δ is denoted by $[y]_\delta$. This notation is extended to denote element-wise quantization of vector (e.g., $[\mathbf{e}]$) and matrix operands (e.g., $[\mathbf{E}]$).

2.1 Sparse coding

One possible form of expressing the *sparse coding problem* is given by

$$\hat{\mathbf{a}}_j = \arg \min_{\mathbf{u} \in \mathbb{R}^p} \|\mathbf{y}_j - \mathbf{D}\mathbf{u}\|_2 \quad \text{s.t.} \quad \|\mathbf{u}\|_0 \leq \gamma, \quad (\text{D.4})$$

where $\gamma \ll p$ indicates the desired *sparse level* of the solution. Since problem (D.4) is non-convex and NP-hard, approximate solutions are sought. This is done either by using greedy methods such as Matching Pursuit (MP) [18], or by solving a convex approximation

to (D.4), commonly known as the *lasso* [19],

$$\hat{\mathbf{a}}_j = \arg \min_{\mathbf{u} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{u}\|_2 \quad \text{s.t.} \quad \|\mathbf{u}\|_1 \leq \tau. \quad (\text{D.5})$$

There exists a body of results showing that, under certain conditions on γ and \mathbf{D} , the problem (D.4) can be solved exactly via (D.5) or MP (see for example [1, 20]). In other cases, the objective is not to solve (D.4), but to guarantee some property of the estimated $\hat{\mathbf{a}}_j$. For example, in the above mentioned case of AWGN denoising in the wavelets domain, the parameter τ can be chosen so that the resulting estimators are universally optimal with respect to some class of signals [21]. However, if \mathbf{D} is arbitrary, no such choice exists. Also, if \mathbf{D} is orthonormal, the problem (D.5) admits a closed form solution obtained via the so-called *soft thresholding* [21]. However, again, for general \mathbf{D} , no such solution exists, and the search for efficient algorithms has been a hot topic recently, e.g., [22, 23, 24].

2.2 Dictionary learning

When \mathbf{D} is an optimization variable, we refer to the resulting problem as *dictionary learning*:

$$(\hat{\mathbf{A}}, \hat{\mathbf{D}}) = \arg \min_{\mathbf{A}, \mathbf{D}} \sum_{j=1}^n \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{a}_j\|_2^2 \quad \text{s.t.} \quad \|\mathbf{a}_j\|_r \leq \tau \forall j, \quad \|\mathbf{d}_k\|_2 \leq 1 \forall k, \quad (\text{D.6})$$

with $0 \leq r \leq 1$. The constraint $\|\mathbf{d}_k\|_2 \leq 1$, $k = 1, \dots, p$, is necessary to avoid an arbitrary decrease of the cost function by setting $\mathbf{D} \leftarrow \alpha \mathbf{D}$, $\mathbf{A} \leftarrow \frac{1}{\alpha} \mathbf{A}$, for any $\alpha > 1$. The cost function in (D.6) is non-convex in (\mathbf{A}, \mathbf{D}) , so that only local convergence can be guaranteed. This is usually achieved using alternate optimization in \mathbf{D} and \mathbf{A} . See for example [25, 26] and references therein.

2.3 Issues with traditional sparse models: a motivating example

Consider the K-SVD-based [25] sparse image restoration framework [27]. This is an ℓ_0 -based dictionary learning framework, which approximates (D.6) for the case $r = 0$ by alternate minimization. In the case of image denoising, the general procedure can be summarized as follows:

1. An initial, *global* dictionary \mathbf{D}_0 is learned using training samples for the class of data to be processed (in this case small patches of natural images). The user must supply a patch width w , a dictionary size p and a value for τ .
2. The noisy image is decomposed into overlapping $w \times w$ patches (one patch per pixel of the image), and its noisy patches are used to further adapt \mathbf{D} using the following *denoising* variant of (D.6),

$$(\hat{\mathbf{D}}, \hat{\mathbf{A}}) = \arg \min_{\mathbf{D}, \mathbf{A}} \sum_{j=1}^n \|\mathbf{a}_j\|_0, \quad \text{s.t.} \quad \frac{1}{2} \|\mathbf{y}_j - \mathbf{D}\mathbf{a}_j\|_2^2 \leq C\sigma^2, \quad \|\mathbf{d}_k\|_2 = 1, \quad k = 1, \dots, p. \quad (\text{D.7})$$

Here the user must further supply a constant C (in [27], it is 1.32), the noise variance σ^2 , and the number of iterations J of the optimization algorithm, which is usually kept small to avoid over-fitting (the algorithm is *not* allowed to converge).

3. The final image is constructed by assembling the patches in $\hat{\mathbf{Y}} = \hat{\mathbf{D}}\hat{\mathbf{A}}$ into the corresponding original positions of the image. The final pixel value at each location is an average of all the patches to which it belongs, plus a small fraction $0 \leq \lambda \leq 1$ of the original noisy pixels ($\lambda = 30/\sigma$ in [27]).

Despite the good results obtained for natural images, several aspects of this method are not satisfactory:

- Several parameters ($w, p, \tau, C, J, \lambda$) need to be tuned. *There is no interpretation, and therefore no justifiable choice for these parameters, other than maximizing the empirical performance of the algorithm (according to some metric, in this case PSNR) for the data at hand.*
- The effect of such parameters on the result is shadowed by the effects of later stages of the algorithm and their associated parameters (e.g. overlapping patch averaging). *There is no fundamental way to optimize each stage separately.*

As a partial remedy to the first problem, Bayesian sparse models were developed (e.g., [4]) where these parameters are assigned prior distributions which are then learned from the data. However, this approach still does not provide objective means to compare different models (with different priors, for example). Further, the Bayesian technique implies having to repeatedly solve possibly costly optimization problems, increasing the computational burden of the application.

As mentioned in the introduction, this work proposes to address the above practical issues, as well as to provide a new angle into dictionary learning, by means of the MDL principle for model selection. The details on how this is done are the subject of the following sections.

3 Sparse model selection and MDL

Given data \mathbf{Y} , a maximum support size γ and a dictionary size p , traditional sparse modeling provides means to estimate the best model $M = (\mathbf{A}, \mathbf{D})$ for \mathbf{Y} within the set $\mathcal{M}(\gamma, p)$ defined as

$$\mathcal{M}(\gamma, p) := \{(\mathbf{A}, \mathbf{D}) : \|\mathbf{a}_j\|_0 \leq \gamma, j = 1, \dots, n, \mathbf{D} \in \mathbb{R}^{m \times p}\}. \quad (\text{D.8})$$

We call such set a *sparse model class* with hyper-parameters (γ, p) . Such classes are nested in the following way: first, for a fixed dictionary size p we have $\mathcal{M}(\gamma - 1, p) \subset \mathcal{M}(\gamma, p)$. Also, for fixed γ , if we consider $\mathcal{M}(\gamma, p - 1)$ to be a particular case of $\mathcal{M}(\gamma, p)$ where the p -th atom is all-zeroes and $a_{pj} = 0, \forall j$, then we also have that $\mathcal{M}(\gamma, p - 1) \subset \mathcal{M}(\gamma, p)$.

If one wants to choose the best model among all possible classes $\mathcal{M}(\gamma, p)$, the problem becomes one of *model selection*. The general objective of model selection tools is to define an objective criterion for choosing such model. In particular, MDL model selection uses codelength as such criterion. More specifically, this means first computing the best model within each family as

$$(\mathbf{A}(\gamma, p), \mathbf{D}(\gamma, p)) = \arg \min \{L(\mathbf{Y}, \mathbf{A}, \mathbf{D}) : (\mathbf{A}, \mathbf{D}) \in \mathcal{M}(\gamma, p)\},$$

and then choosing $(\hat{\gamma}, \hat{p}) = \arg \min \{L(\mathbf{Y}, \mathbf{A}(\gamma, p), \mathbf{D}(\gamma, p)) : 0 \leq \gamma \leq p, p > 0\}$.

When \mathbf{D} is fixed, which is the case of sparse coding, the only model parameter is \mathbf{A} , and we have $p + 1$ possible classes, $\mathcal{M}(\gamma) = \{\mathbf{A} : \|\mathbf{a}_j\|_0 \leq \gamma, j = 1, \dots, n\}$, one for each $0 \leq \gamma \leq p$. If each data sample \mathbf{y}_j from \mathbf{Y} is encoded independently, then, as with traditional sparse coding (the framework can also be extended to *collaborative* models), the model selection problem can be broken into n sub-problems, one per sample, by redefining the model class accordingly as $\mathcal{M}(\gamma) = \{\mathbf{a} : \|\mathbf{a}\|_0 \leq \gamma\}$. Clearly, in the latter case, the optimum γ can vary from sample to sample.

Compared to the algorithm in Section 2.3, we have a *fundamental, intrinsic* measure of the quality of each model, the codelength $L(\mathbf{Y}, \mathbf{A}, \mathbf{D})$, to guide our search through the models, and which is unobscured from the effect of possible later stages of the application. In contrast, there is no obvious intrinsic measure of quality for models learned through (D.8), making comparisons between models learned for different parameters (patch width

w , regularization parameter τ , norm r , constants C, λ) possible only in terms of the observed results of the applications where they are embedded. The second advantage of this framework is that it allows to select, in a fundamental fashion, the best model parameters *automatically*, thus resulting in parameter-free algorithms.²

Such advantages will be of practical use only if the resulting computational algorithms are not orders of magnitude slower than the traditional ones, and efficient algorithms are a critical component of this framework, see Section 5.

3.1 A brief introduction to MDL

For clarity of the presentation, in this section we will consider \mathbf{D} fixed, and a single data sample \mathbf{y} to be encoded. The Minimum Description Length principle was pioneered by Rissanen [7] in what is called “early MDL,” and later refined by himself [8] and other authors to form what is today known as “modern MDL” (see [28] for an up-to-date extensive reference on the subject). The goal of MDL is to provide an objective criterion to select the model M , out of a family of competing models \mathcal{M} , that gives the best description of the *given* data \mathbf{y} . In this case of sparse coding with fixed dictionary we have $M = \mathbf{a}$.

The main idea of MDL is that, the best model for the data at hand is the one that is able to capture more *regularity* from it. The more regularity a model captures, the more succinct the description of the data will be under that model (by avoiding redundancy in the description). Therefore, MDL will select the best model as the one that produces the shortest (most efficient) description of the data, which in our case is given by $L(\mathbf{y}, \mathbf{a})$.

As mentioned in Section 1, MDL translates the problem of choosing a codelength function $L(\cdot)$ to one of choosing probability models by means of the ideal Shannon codelength

²For the case of image processing, the patch width w is also a relevant parameter that could be automatically learned with the same MDL-based framework presented here. However, since it is specific to image processing, and due to space constraints and for clarity of the exposition, it will not be considered as part of the model selection problem hereafter.

assignment $L(\mathbf{y}, \mathbf{a}) = -\log P(\mathbf{y}, \mathbf{a})$. It is common to extend such ideal codelength to continuous random variables x with probability density function $p(x)$ as $L(x) = -\log p(x)$, by assuming that they will be quantized with sufficient precision so that

$$P([x]_\delta) \approx p(x)\delta, \quad (\text{D.9})$$

and disregarding the constant term $-\log \delta$ in $L(x)$, as it is inconsequential for model selection. However, in our framework, the optimum quantization levels will often be large enough so that such approximations are no longer valid.

To produce a complete description of the data \mathbf{y} , the best model parameters \hat{M} used to encode \mathbf{y} need to be included in the description as well. If the only thing we know is that \hat{M} belongs to a given class \mathcal{M} , then the cost of this description will depend on how large and complex \mathcal{M} is. MDL will penalize more those models that come from larger (more complex) classes. This is summarized in one of the fundamental results underlying MDL [8], which establishes that the minimum number of bits required for encoding *any* data vector \mathbf{y} using a model from a class \mathcal{M} has the form $L_{\mathcal{M}}(\mathbf{y}) = \mathcal{L}_{\mathcal{M}}(\mathbf{y}) + \mathcal{C}(\mathcal{M})$, where $\mathcal{L}_{\mathcal{M}}(\mathbf{y})$ is called the *stochastic complexity*, which depends only on the particular instance of \mathbf{y} being encoded, and $\mathcal{C}(\mathcal{M})$ is an unavoidable *parametric complexity* term, which depends *solely* on the structure, geometry, etc., of the model class \mathcal{M} .

In the initial version of MDL [7], the parameter \hat{M} was first encoded separately using $L(\hat{M})$ bits, and then \mathbf{y} was described given \hat{M} using $L(\mathbf{y}|\hat{M})$ bits, so that the complete description of \mathbf{y} required $L(\mathbf{y}|\hat{M}) + L(\hat{M})$ bits. This is called a *two-parts code*. An asymptotic expression of this MDL was developed in [7] which is equivalent to the BIC criterion [6], only in the asymptotic regime. As we will see next, modern MDL departs significantly from this two-parts coding scheme.

3.2 Modern MDL and universal coding

The main difference between “early” [7] and “modern” [8, 9] MDL is the introduction of *universal codes* as the main building blocks for computing codelengths. In a nutshell, universal coding can be regarded as an extension of the original Shannon theory to the case where the probability model $P(\cdot)$ of the data to be encoded is not fully specified, but only known to belong to a certain class of candidate probability models \mathcal{M} (recall that classic Shannon theory assumes that $P(\cdot)$ is perfectly known). For example, \mathcal{M} can be a family of parametric distributions indexed by some parameter M . Akin to Shannon theory, for an encoding scheme to be called *universal*, the codelengths it produces need to be optimal, in some sense, with respect to the codelengths produced by all the models in \mathcal{M} .

Various universality criteria exist. For example, consider the *codelength redundancy* of a model $Q(\cdot)$, $\mathcal{R}(\mathbf{y}; Q) = -\log Q(\mathbf{y}) - [\arg \min_{P \in \mathcal{M}} -\log P(\mathbf{y})]$. In words, this is the codelength overhead obtained with $Q(\cdot)$ for describing an instance \mathbf{y} , compared to the best model in \mathcal{M} that could be picked for \mathbf{y} , *with hindsight* of \mathbf{y} . For example, if \mathcal{M} is a parametric family, such model is given by the maximum likelihood (ML) estimator of M . A model $Q(\cdot)$ is called *minimax universal*, if it minimizes the *worst case redundancy*, $\mathcal{R}(Q) = \arg \max_{\mathbf{y} \in \mathbb{R}^m} \mathcal{R}(\mathbf{y}; Q)$. One of the main techniques in universal coding is *one-part coding*, where the data \mathbf{y} and the best class parameter \hat{M} are encoded jointly. Such codes are used in the line of work of “MDL denoising” due to Rissanen and his collaborators [14, 15]. However, applying one-part codes at this level restricts the probability models to be used.³ As a consequence, the results obtained with this approach in such works are not competitive with the state-of-the-art. Therefore, in this work, we maintain a two-parts encoding scheme (or three parts, if \mathbf{D} is to be encoded as well), where we separately describe \mathbf{a} , \mathbf{D} , and \mathbf{y} given (\mathbf{a}, \mathbf{D}) . We will however use universal codes to describe each of these parts as

³In particular, those used in [14, 15] are based on the Normalized Maximum Likelihood (NML) universal model [29], which requires closed-form MLE estimators for its evaluation, something that cannot be obtained for example with a Laplacian prior on \mathbf{a} and non-orthogonal dictionaries.

efficiently as possible. Details on this are given in the next section.

4 Encoding scheme

We now define the models and encoding schemes used to describe each of the parts that comprise a sparse model for a data sample \mathbf{y} ; that is, the dictionary \mathbf{D} , the coefficients \mathbf{a} , and the approximation error $\mathbf{e} = \mathbf{y} - \mathbf{D}\mathbf{a}$ (which can include both the noise and the model deviation), which can be regarded as the conditional description of \mathbf{y} given the model parameters (\mathbf{a}, \mathbf{D}) . The result will be a cost function $L(\mathbf{y})$ of the form (note that $\mathbf{y} = \mathbf{D}\mathbf{a} + \mathbf{e}$ can be fully recovered from $(\mathbf{e}, \mathbf{a}, \mathbf{D})$),

$$L(\mathbf{y}, \mathbf{a}, \mathbf{D}) = L(\mathbf{e}|\mathbf{a}, \mathbf{D}) + L(\mathbf{a}|\mathbf{D}) + L(\mathbf{D}).$$

While computing each of these parts, three main issues need to be dealt with:

1. **Define appropriate probability models.** Here, it is fundamental to incorporate as much prior information as possible, so that no cost is paid in learning (and thereby coding) already known statistical features of the data. Examples of such prior information include sparsity itself, invariance to certain transformations or symmetries, and (Markovian) dependencies between coefficients.
2. **Deal with unknown parameters.** We will use universal encoding strategies to encode data efficiently in terms of families of probability distributions.
3. **Model the effect of quantization.** All components, $\mathbf{e}, \mathbf{a}, \mathbf{D}$ need to be quantized to some precisions, respectively $\delta_e, \delta_a, \delta_d$, in order to obtain finite, realistic codelengths for describing \mathbf{y} (when the precision variable is obvious from the argument to which it is applied, we drop it to simplify the notation, for example, we will write $[e]_{\delta_e}$ as $[e]$). This quantization introduces several complications, such as optimization over

discrete domains, increase of sparsity by rounding to zero, increase of approximation error, and working with discrete probability distributions.

All such issues need to be considered with efficiency of computation in mind. The discussion will focus first on the traditional, single-signal case where each sample y is encoded separately from the rest. At the end of this section, we will also discuss the extension of this framework to a multi-signal case, which has several algorithmic and modeling advantages over the single-signal case, and which forms the basis for the dictionary learning algorithms described later.

4.1 Encoding the sparse coefficients

Probability model: Each coefficient in \mathbf{a} is modeled as the product of three (non-independent) random variables, $A = ZS(V + \delta_a)$, where $Z = 1$ implies $A \neq 0$, $S = \text{sgn}(A)$, and $V = \max\{|A| - \delta_a, 0\}$ is the absolute value of A corrected for the fact that $V \geq \delta_a$ when $Z = 1$.⁴

We model Z as a Bernoulli variable with $P(Z = 1) = \rho_a$. Conditioned on $Z = 0$, $S = V = 0$ with probability 1, so no encoding is needed.⁵ Conditioned on $Z = 1$, we assume $P(S = -1) = P(S = 1) = 1/2$, and V to be a (discretized) exponential, $\text{Exp}(\theta_a)$. With these choices, $P(SV|Z = 1)$ is a (discretized) Laplacian distribution, which is a standard model for transform (e.g., DCT, Wavelet) coefficients. This encoding scheme is depicted in Figure D.1(a,b). The resulting model is a particular case of the “spike and slab” model used in statistics (see [30] and references therein). A similar factorization of the sparse coefficients is used in the Bayesian framework as well [4].

Unknown parameters: According to the above model, the resulting encoding scheme for the coefficients (sparse code) is a three-parts code: $L(\mathbf{a}) = L(\mathbf{z}) + L(\mathbf{s}|\mathbf{z}) + L(\mathbf{a}|\mathbf{s}, \mathbf{z})$. The

⁴Note that it is necessary to encode S and V separately, instead of considering SV as one random variable, so that the sign of A can be recovered when $|V| = 0$.

⁵We can easily extend the proposed model beyond $S = V = 0$ and consider a distribution for S, V when $Z = 0$. This will naturally appear as part of the coding cost. This extends standard sparse coding to the case where the non-sparse component of the vector are not necessarily zero.

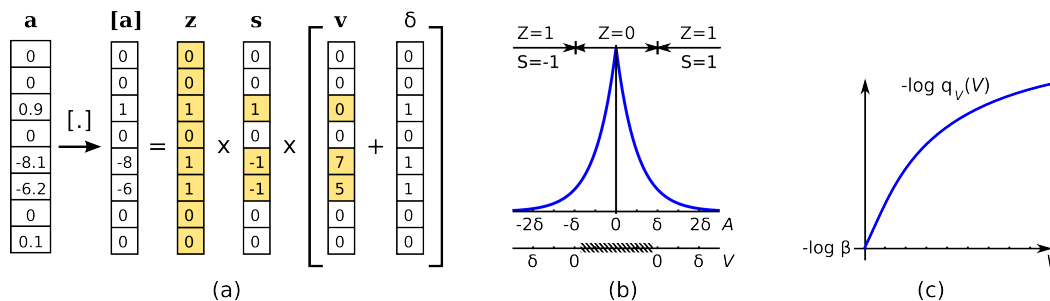


Figure D.1: Encoding of the sparse code. (a) After quantization (here $\delta_a = 1$), each coefficient a_k is decomposed into three variables, $z_k = \mathbf{1}(a_k)$, $s_k = \text{sgn}(a_k)$ and $v_k = \max\{|a_k| - \delta_a, 0\}$. These are respectively modeled by random variables $Z \sim \text{Ber}(\rho_a)$, $S \sim \text{Ber}(1/2)$, $V \sim \text{Exp}(\theta_a)$ (only the shaded numbers are actually encoded) (b) Scheme of the mapping from continuous coefficients (random variable A), into Z , S and V . (c) Ideal codelength for the MOE model for V , $-\log q_V(V; \kappa, \beta)$. This is a smooth, concave function.

support \mathbf{z} is described using the *enumerative two-parts code* [31], which first describes its size, $\|\mathbf{a}\|_0$, using $\log p$ bits, and then the particular arrangement of the ones in \mathbf{z} using $\log \binom{p}{\|\mathbf{a}\|_0}$ bits. The total codelength for coding \mathbf{z} is then $L(\mathbf{z}) = \log p + \log \binom{p}{\|\mathbf{a}\|_0}$. This is a universal encoding scheme, and as such is more efficient than those used previously in [11, 13]. Then, $L(\mathbf{s}|\mathbf{z}) = \|\mathbf{a}\|_0$ bits are needed to encode $\mathbf{s}_{[\mathbf{z}]}$, the actual signs of the non-zero coefficients. Finally, we need to encode the magnitudes of the $\|\mathbf{a}\|_0$ non-zero coefficients, $\mathbf{a}_{[\mathbf{z}]}$. We do so by considering it first as a sequence of exponentially-distributed continuous random variables, to which quantization is applied later. Since the parameter θ_a of the exponential is unknown,⁶ we use a universal model $q_V(\cdot)$ for the class of continuous exponential distributions instead. We obtain such universal model $q_V(V)$ via a convex mixture, one of the standard techniques for this,

$$q_V(V; \kappa_a, \beta_a) = \int_0^{+\infty} \Gamma(\theta; \kappa_a, \beta_a) \frac{\theta}{2} e^{-\theta|V|} d\theta, \quad (\text{D.10})$$

⁶This parameter is related to the sparsity level, and as discussed in Section 2.3, is usually assumed known or determined via cross-validation. Following [32], here we use tools from universal modeling, which permit to also automatically handle the non-stationarity of this parameter and its expected variability for different non-zero entries of \mathbf{a} .

where the mixing function $\Gamma(\theta; \kappa, \beta) = \Gamma(\kappa)^{-1} \theta^{\kappa-1} \beta^\kappa e^{-\beta\theta}$, is the Gamma density function of (non-informative) shape and scale parameters κ and β . With this choice, (D.10) has a closed form expression, and the degenerate cases $\theta = 0$ and $\theta = \infty$ are given zero weight. The resulting *Mixture of Exponentials* (MOE) density function $q_V(V)$, is given by (see [32] for details),

$$q_V(V; \beta_a, \kappa_a) = \kappa_a \beta_a^{\kappa_a} (V + \beta_a)^{-(\kappa_a+1)}, V \in \mathbb{R}^+.$$

Note that the universality of this mixture model does not depend on the values of the parameters κ_a, β_a , and guided by [32], we set $\kappa_a = 3.0$ and $\beta_a = 50$. The ideal Shannon codelength for this density function distribution is given by $-\log q_V(V; \kappa_a, \beta_a) = -\log \kappa_a - \kappa_a \log \beta_a + (\kappa_a + 1) \log(V + \beta_a)$. This function, shown in Figure D.1(c), is non-convex, however continuous and differentiable for $V > 0$.

Quantization: On one hand, quantizing the coefficients to a finite precision δ_a increases the approximation/modeling error, from $\mathbf{y} - \mathbf{D}\mathbf{a}$ to $\mathbf{y} - \mathbf{D}[\mathbf{a}]_{\delta_a}$. This additional error, $\mathbf{D}(\mathbf{a} - [\mathbf{a}]_{\delta_a})$, will clearly increase with δ_a . On the other hand, larger δ_a will reduce the description length of the non-zero values of the coefficients, $[\mathbf{v}]_{\delta_a}$. In practice, for reasonable quantization steps, the error added by such quantization is negligible compared to the approximation error. For example, for describing natural images divided in patches of 8×8 pixels, our experiments indicate that there is no practical advantage in using a value smaller than $\delta_a = 16$. Consequently, our current algorithms do not attempt to optimize the code-length on this parameter, and we have kept this value fixed throughout all the experiments of Section 7.

4.2 Encoding the error

Probability model: Most sparse coding frameworks, including all the mentioned MDL-based ones, assume the error \mathbf{e} to be solely due to measurement noise, typically of the

AWGN type. However, \mathbf{e} actually contains a significant component which is due to a systematic deviation of the model from the clean data. Following this, we model the elements of \mathbf{e} as samples of an IID random variable E which is the linear combination of two independent variables, $E = \hat{E} + N$. Here $N \sim \mathcal{N}(0, \sigma_e^2)$ represents random measurement noise in \mathbf{y} . We assume the noise variance σ_e^2 known, as it can be easily and reliably estimated from the input data (for example, taking the minimum empirical variance over a sufficient number of sub-samples). The distribution of the second variable, $\hat{E} \sim \text{Lap}(0, \theta_e)$ is a Laplacian of unknown parameter θ_e , which represents the error component due to the model itself. The resulting continuous distribution $p_E(E)$, which we call ‘‘LG,’’ is the convolution of the distributions of both components (see [33] for details on the derivation),

$$\begin{aligned} p_E(E; \sigma_e^2, \theta_e) &= \int_{\zeta=-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma_e^2}} e^{-\frac{\zeta^2}{2\sigma_e^2}} \frac{1}{2\theta_e} e^{-\frac{|E-\zeta|}{\theta_e}} d\zeta \\ &= \frac{1}{4\theta_e} e^{\frac{\sigma_e^2}{2\theta_e^2}} \left[e^{E/\theta_e} \text{erfc}\left(\frac{E + \sigma_e^2/\theta_e}{\sqrt{2}\sigma_e}\right) + e^{-E/\theta_e} \text{erfc}\left(\frac{-E + \sigma_e^2/\theta_e}{\sqrt{2}\sigma_e}\right) \right], \quad (\text{D.11}) \end{aligned}$$

where $\text{erfc}(u) = \frac{2}{\sqrt{\pi}} \int_u^{+\infty} e^{-t^2} dt$ is the *complimentary Gauss error function*. The ideal code-length, $-\log p_E(E)$, is shown in Figure D.2(a) for various parameter values. This function is convex and differentiable on \mathbb{R} , which is nice for optimization purposes. Figure D.2(b) shows its derivative, or so called ‘‘influence function’’ in robust statistics. It can be verified that $-\log p_E(E)$ behaves like a Laplacian with parameter θ_e for large values of E . Further, since its derivative is bounded, the influence of outliers is diminished. In fact, $-\log p_E(E)$ is easily verified to be a ψ -type M-estimator, a family of functions used in robust statistics (see [17]). Thus, using this model, we obtain an information-theoretic robust estimator, which is consistent with the motivations leading to its use in our framework, and which has a significant practical impact in the experimental results.

Unknown parameters: Since θ_e is unknown, encoding \mathbf{e} efficiently calls for the use of universal codes. In this case, again, we employ a mixture model. Since the parameter θ_e

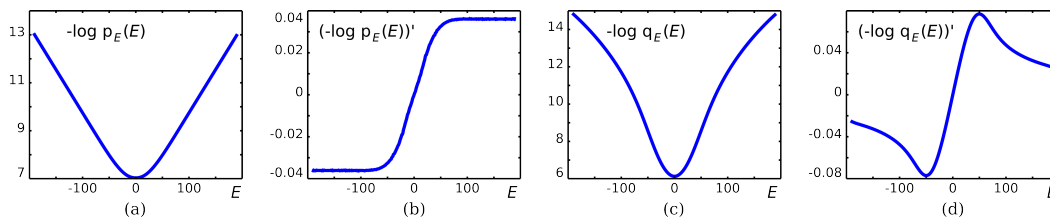


Figure D.2: Residual probability model. (a) Ideal codelength function of the “LG” distribution, $-\log p_E(E)$, (b) LG influence function, that is, $(-\log p_E(E))'$, (c) universal mixture for the LG model (MOEG), (d) MOEG influence function.

comes from the underlying Laplacian component, we again use a Gamma for the mixing function,

$$q_E(E; \sigma_e^2, \kappa_e, \beta_e) = \int_0^{+\infty} \Gamma(\theta; \kappa_e, \beta_e) p_E(E; \sigma_e^2, \theta) d\theta. \quad (\text{D.12})$$

We call this model MOEG. As with the MOE model, the universality of this model is guaranteed by the theory for the choice of its underlying mixing function, for any (non-informative) κ_e and β_e . In this case, we use $\kappa_e = 3.0$ and $\beta_e = \delta_e$. Also, we know from the discussion above that σ_e^2 can be easily and reliably estimated from the data. Thus, we can say that the model for E is parameter-free in this case as well. Figure D.2(c) shows the numerical evaluation of the ideal Shannon codelength $-\log q_E(E; \sigma_e^2, \kappa_e, \beta_e)$, which is non-convex. However, it is twice differentiable everywhere, again a desirable property for optimization purposes (more on this in sections 5 and 6). As with the LG distribution, $-\log q_E(E)$ is an ψ -type M-estimator, in this case, a *redescending* M-estimator, since its derivative (Figure D.2(d)) vanishes to 0 at ∞ . As such, $-\log q_E(E)$, derived from the universal model corresponding to $p_E(E)$, can reject outliers even more aggressively than $-\log p_E(E)$, again marrying robust statistics with information theory in a natural way.

Quantization: To losslessly encode finite-precision input data such as digital images, the quantization step of the error coefficients needs not be more than that of the data itself, δ_y ,

and we simply quantize the error coefficients uniformly with step $\delta_e = \delta_y$. For example, for 8-bit digital images, we set $\delta_e = \delta_y = 1$.

4.3 Model for the dictionary

Probability model: Dictionary learning practice shows that learned atoms, unsurprisingly, present features that are similar to those of the original data. For example, the piecewise smoothness of small image patches is to be expected in the atoms of learned dictionaries for such data. This prior information, often neglected in dictionary learning algorithms, needs to be taken into account for encoding such atoms efficiently.

We embody such information in the form of *predictability*. This is, we will encode an atom $\mathbf{d} \in \mathbb{R}^m$ as a sequence of causal prediction residuals, $\mathbf{b} \in \mathbb{R}^m$, $b_{i+1} = d_{i+1} - \tilde{d}_{i+1}(d_1, d_2, \dots, d_i)$, $1 \leq i < m$, a function of the previously encoded elements in \mathbf{d} . In particular, if we restrict \tilde{d}_{i+1} to be a linear function, the residual vector can be written as $\mathbf{b} = \mathbf{W}\mathbf{d}$, where $\mathbf{W} \in \mathbb{R}^{m \times m}$ is lower triangular due to the causality constraint (this aspect has important efficiency consequences in the algorithms to be developed in Section 6). This is depicted in Figure D.3, along with the specific prediction scheme that we adopted for the image processing examples in Section 7. In this case we consider an atom \mathbf{d} to be an $\sqrt{m} \times \sqrt{m}$ image patch, and use a causal bi-linear predictor where the prediction of each pixel in the dictionary atom is given by `north_pixel + west_pixel - northwest_pixel`.

As a general model for linear prediction residuals, we assume \mathbf{b} to be a sequence of IID Laplacian samples of parameter θ_d . In principle, θ_d is also unknown. However, describing \mathbf{D} is only meaningful for dictionary learning purposes, and, in that case, \mathbf{D} is updated iteratively, so that when computing an iterate $\mathbf{D}^{(t)}$ of \mathbf{D} , we can use $\mathbf{D}^{(t-1)}$ to estimate and fix θ_d via ML (more on this θ_d later in Section 6). Thus, we consider θ_d to be known.

Quantization: When \mathbf{A} is fixed during a dictionary learning iteration (which consists of an alternate descent between \mathbf{D} and \mathbf{A}), we can view (\mathbf{A}, \mathbf{Y}) as n input-output training

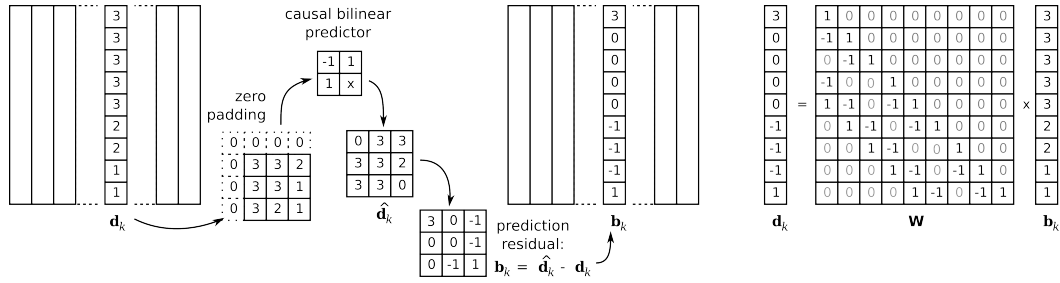


Figure D.3: Prediction scheme used for learning natural image patches dictionaries (in this example, 3×3 patches, and $m = 9$). An atom \mathbf{d}_k is arranged as a 3×3 patch, and a causal bi-linear predictor (shown as a 2×2 template) with zero-padding (pixels outside of the patch are assumed 0) is applied to it, producing a predicted atom $\hat{\mathbf{d}}_k$ and a residual $\mathbf{b}_k = \mathbf{d}_k - \hat{\mathbf{d}}_k$. The previous operation can be written as $\mathbf{b}_k = \mathbf{W}\mathbf{d}_k$, with $\mathbf{W} \in \mathbb{R}^{9 \times 9}$ the linear mapping from atom to prediction residuals corresponding to this example.

pairs, and \mathbf{D} as the ML estimator of the linear coefficients describing such mapping via $\mathbf{Y} = \mathbf{D}\mathbf{A} + \mathbf{E}$. Based on this, we use the quantization step $\delta_d = 1/\sqrt{n}$, which is an optimal step for encoding the ML parameter in two-part codes, as described in [8, Theorem 1].

Computation: Computing $L(\mathbf{D})$ is only relevant for learning purposes. In general, since $\|\mathbf{d}_k\|_2 \leq 1$, and $\|\mathbf{d}_k\|_2 \leq \sqrt{m}\|\mathbf{d}_k\|_1$, we have that $\hat{\theta}_d = (pm)^{-1} \sum_k \|\mathbf{d}_k\|_1 \leq (p\sqrt{m})^{-1} \ll \delta_d = \sqrt{n}$, and the error of using the approximation (D.9) is not significant,

$$L(\mathbf{D}) = \sum_{k=1}^p L(\mathbf{d}_k) \approx \sum_{k=1}^p \{-\log p(\mathbf{W}\mathbf{d}_k; \theta_d) - m \log \delta_d\} = \theta_d \sum_{k=1}^p \|\mathbf{W}\mathbf{d}_k\|_1 + \frac{mp}{2} \log n + c, \quad (\text{D.13})$$

where $p(\mathbf{W}\mathbf{d}_k)$ is the IID Laplacian distribution over the k -th atom prediction residual vector $\mathbf{W}\mathbf{d}_k$, and c is a fixed constant. For p fixed (we will later see how to learn the dictionary size p as well), the above expression is simply an ℓ_1 penalty on the atom prediction residual coefficients. As we will see in Section 6, this allows us to use efficient convex optimization tools to update the atoms.

4.4 Extension to sequential (collaborative) coding

One natural assumption that we can often make on the set of data samples \mathbf{Y} is that, besides all being sparsely representable in terms of the learned dictionary \mathbf{D} , they share other statistical properties. For example, we can assume that the underlying unknown model parameters, $\theta_e, \rho_a, \theta_a, \theta_d$, are the same for all columns of the sparse data decomposition (\mathbf{E}, \mathbf{A}) .

Under such assumption, if we encode each column of \mathbf{Y} sequentially, we can learn statistical information from the ones already encoded and apply it to estimate the unknown parameters of the distributions used for encoding the following ones. The general idea is depicted in Figure D.4(a). Concretely, suppose we have already encoded $j - 1$ samples. We can then use $[\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{(j-1)}]$ to estimate θ_e , and $[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_{(j-1)}]$ to estimate θ_a and ρ_a , and “plug-in” these parameters to encode the j -th sample. This justifies the name of this encoding method, which is known in the coding literature as *sequential plug-in* encoding. This encoding strategy has several advantages: 1) For common parameter estimators such as ML, this method can be shown to be universal; 2) Since all distribution parameters are fixed (pre-estimated) when encoding the j -th sample, we can use the “original,” non-universal distributions assumed for modeling \mathbf{e}_j (LG) and \mathbf{a}_j (Laplacian), which have closed forms and are usually faster to compute (together with (D.9)) than their universal mixture counterparts; 3) Furthermore, these original distributions are convex, so that in this case, given a fixed support, we are able to exactly minimize the codelength over the non-zero coefficient values; 4) With many samples available for parameter estimation, we can potentially afford more complex models.

Residual: We estimate θ_e in two steps. First, since the random variable E is an independent sum of two random variables, $E = \hat{E} + N$, we have that $\text{var}(E) = \text{var}(\hat{E}) + \text{var}(N) = \text{var}(\hat{E}) + \sigma_e^2$. Now, since \hat{E} is Laplacian, we have that $\text{var}(\hat{E}) = 2\theta_e^2$. Combining both equations we have that $\theta_e = 0.5\sqrt{\text{var}(\hat{E}) - \sigma_e^2}$. With the noise variance σ_e^2 assumed known, and using

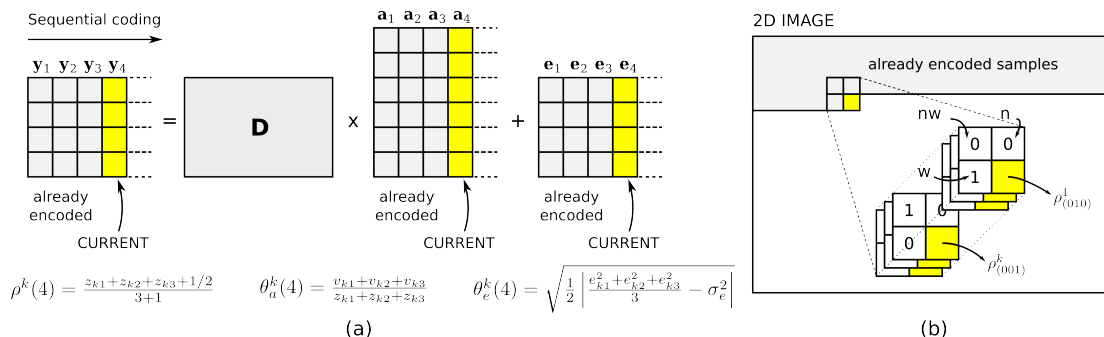


Figure D.4: Collaborative encoding scheme. (a) In this example, 3 samples have already been encoded, and we are about to encode sample 4. The formulas for estimating the various model parameters are shown for $j = 4$, in particular those for the error and the coefficients associated to the k -th atom (the k -th row of \mathbf{A}). (b) Markov model for the coefficients support matrix \mathbf{Z} . Here, a sample patch \mathbf{y} is about to be encoded. Here the first atom was only used by the pixel to the west, so that the Markov state for modeling z_1 is $(n, w, nw) = (0, 1, 0)$, and $P(z_1 = 1) = \rho_{(0,1,0)}^1$. As for the k -th atom, only the nw pixel has used it, so that the Markov state for z_k is $(0, 0, 1)$, that is, $P(z_k = 1) = \rho_{(0,0,1)}^k$.

the standard unbiased variance estimator, $\hat{\text{var}}(\hat{E}) = (p(j-1))^{-1} \|\mathbf{E}_{[1, \dots, (j-1)]}\|_F^2$, we obtain

$$\hat{\theta}_e = 0.5 \sqrt{\max\{(p(j-1))^{-1} \|\mathbf{E}_{[1, \dots, (j-1)]}\|_F^2 - \sigma_e^2, 0\}},$$

where the maximization guarantees that $\hat{\theta}_e \in \mathbb{R}^+$.

Coefficients: In the case of \mathbf{a} , we have in principle two unknown parameters, the probability of an element being non-zero, ρ_a , and the scale parameter of the Laplacian governing the non-zero values, θ_a (both previously handled with universal models). Here, however, we extend the model, drawing from the well known empirical fact that coefficients associated to different atoms can have very different statistics, both in frequency and variance. This is typical of DCT coefficients for example (see [34]), and has been consistently observed for learned dictionaries as well [32]. Therefore, we will consider a separate set of parameters (ρ_a^k, θ_a^k) for each row k of \mathbf{A} , \mathbf{a}^k . We update such parameters from the coefficients observed in the respective row for the already-computed samples,

$(a_{k1}, a_{k2}, \dots, a_{k(j-1)})$, and encode each k -th coefficient in \mathbf{a}_j (more specifically, in \mathbf{z}_j , and \mathbf{v}_j), as the j -th sample of the respective row. Concretely, let $n_1^k = \sum_{j'=1}^{(j-1)} z_{kj'}$ be the number of non-zero coefficients observed so far in the k -th row. For ρ_a^k , we use the Krichevsky-Trofimov (KT) estimator [35],

$$\hat{\rho}_a^k = \frac{n_1^k + 0.5}{j}, \quad (\text{D.14})$$

which is a universal plug-in encoding scheme for Bernoulli sequences of unknown parameter. For encoding v_{kj} , we apply the ML estimator for the exponential family to the non-zero coefficients observed so far in the k -th row. Recalling that $v_{kj} = \max\{|a_{kj'}| - \delta_a, 0\}$, the resulting estimator is given by

$$\hat{\theta}_a^k = \frac{\sum_{j'=1}^{(j-1)} \max\{|a_{kj'}| - \delta_a, 0\}}{n_1^k}.$$

Markovian dependencies: In many applications, spatially/temporally adjacent samples are statistically dependent. For example, we may assume that an atom is more likely to occur for a sample j if it has been used by, say, the $(j-1)$ -th sample (see also [36]). In that case, we may consider different estimations of ρ^k depending on the value of $z_{k(j-1)}$, $\rho_1^k = P(z_{kj} = 1 | z_{k(j-1)} = 1)$, and $\rho_0^k = P(z_{kj} = 1 | z_{k(j-1)} = 0)$. In particular, for the image processing results of Section 7, we use a Markovian model which depends on three previous samples, corresponding to the (causal) neighboring west, north, and northwest patches of the one being encoded. Thus, for each atom k we will have 8 possible parameters, $\rho_{(n,w,nw)}^k$, $(n, w, nw) \in \{0, 1\}^3$, where each value of (n, w, nw) indicates a possible *Markov state* in which a sample may occur. This is depicted in Figure D.4(b). For each state (n, w, nw) , we estimate $\rho_{(n,w,nw)}^k$ using (D.14), with the average taken over the samples which occur in the same state (n, w, nw) .

5 MDL based sparse coding

For the encoding problem, \mathbf{D} is fixed (it has already been learned), and we consider encoding a single data sample \mathbf{y} . The model selection problem here is that of choosing the model (indexed by the sparse code \mathbf{a}) among all the models belonging to the nested family of model classes $\mathcal{M}(\gamma) = \{\mathbf{a} \in \mathbb{R}^p, \|\mathbf{a}\|_0 \leq \gamma\}, \gamma = 0, \dots, p$, that yields the smallest code-length for describing \mathbf{y} . In principle, this calls for finding the best model $\mathbf{a}(\gamma)$ within each model class $\mathcal{M}(\gamma)$, and then selecting $\hat{\mathbf{a}} = \arg \min_{0 \leq \gamma \leq p} L(\mathbf{y}, \mathbf{a}(\gamma))$. However, in order to be computationally efficient, and as with most sparse coding and model selection algorithms, several simplifications and approximations are needed. Let us first consider the problem of finding $\mathbf{a}(\gamma)$,

$$\begin{aligned}
 \mathbf{a}(\gamma) &:= \arg \min_{\mathbf{a} \in \mathcal{M}(\gamma)} L(\mathbf{y}, \mathbf{a}) \\
 &= \arg \min_{\mathbf{a} \in \mathbb{R}^p} -\log P_E(\mathbf{y} - \mathbf{D}\mathbf{a}) - \log P(\mathbf{z}) - \log P(\mathbf{s}|\mathbf{z}) - \log P_V(\mathbf{a}|\mathbf{s}, \mathbf{z}) \\
 &= \arg \min_{\mathbf{a} \in \mathbb{R}^p} -\log P_E(\mathbf{y} - \mathbf{D}\mathbf{a}) - \log \binom{p}{\|\mathbf{a}\|_0} + \|\mathbf{a}\|_0 - \log P_V(\mathbf{a}) \quad \text{s.t.} \quad \|\mathbf{a}\|_0 \leq \gamma.
 \end{aligned}
 \tag{D.15}$$

For quantized \mathbf{a} , this is an optimization problem over a discrete, infinite domain, with a non-convex (in the continuous domain) constraint, and a non-differentiable cost function in \mathbf{a} . Based on the literature on sparse coding, at least two alternatives can be devised at this point. One way is to use a pursuit technique, e.g., [18]. Another option is to use a convex relaxation of the code-length function, e.g., [37]. For the sake of brevity, here we will describe an algorithm loosely based on the first alternative. Details on the convex relaxation method for MDL-based sparse coding will be published elsewhere.

The pursuit-like algorithm, which we call COdelength-Minimizing Pursuit Algorithm

(COMPA), is summarized in Algorithm 3. This is a non-greedy cross-breed between Matching Pursuit (MP) and Forward Stepwise Selection (FSS) [38]. As with those methods, COMPA starts with the empty solution $\mathbf{a}^{(0)} = \mathbf{0}$, and updates the value of one single coefficient at each iteration. Then, given the current correlation $\mathbf{g}^{(t)} = \mathbf{D}\mathbf{e}^{(t)}$ between the dictionary atoms and the current residual, each k -th coefficient in $\mathbf{a}^{(t)}$ is tentatively incremented (or decremented) by $\Delta_k = \left[g_k^{(t)} \right]$, and a candidate codelength \hat{L}_k is computed in each case. The coefficient that produces the smallest $\hat{L}(\mathbf{y}, \mathbf{a})$ is updated to produce $\mathbf{a}^{(t+1)}$.

The logic behind this procedure is that the codelength cost of adding a new coefficient to the support is usually very high, so that adding a new coefficient only makes sense if its contribution is high enough to produce some noticeable effect in the other parts of the codelength. A variant of this algorithm was also implemented where, for each candidate k , the value of the increment Δ_k was refined in order to minimize \hat{L}_k . However, this variant turned out to be significantly slower, and the compression gains were below 0.01 bits per sample (uncompressed codelength is 8 bits per sample). Assuming that $L^{(t)}$ is unimodal, the algorithm stops if the codelength of a new iterate is larger than the previous one. To assess the validity of this assumption, we also implemented a variant which stops, as MP or FSS, when the residual-coefficients correlation $\|\mathbf{g}^{(t)}\|_\infty$ is no longer significant, which typically requires many more iterations. With this variant we obtained a negligible improvement of 0.004 bits per sample, while increasing the computational cost about three times due to the extra iterations required.

6 MDL based dictionary learning

Given that our sparse coding algorithm in Section 5 can select the best support size γ for each sample in \mathbf{Y} , the definition of the model class $\mathcal{M}(\gamma, p)$ given in Section 3, which assumes the same γ for all samples in \mathbf{Y} , is no longer appropriate (we could of course

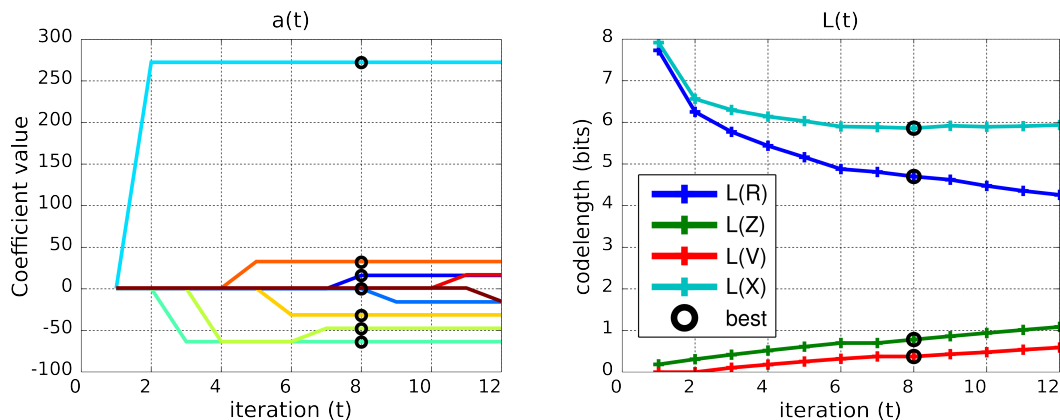


Figure D.5: Typical evolution of the COMPA algorithm. (a) coefficients. (b) codelength. The best iterate (code) is marked with a black circle. Also note that describing the support ($L(Z)$) actually takes more bits than describing the non-zero values ($L(V)$).

add 0-weight coefficients to make γ equal for all data). Instead, for dictionary learning, we consider the model class family $\mathcal{M}(p) = \{(\mathbf{A}, \mathbf{D}), \mathbf{D} \in \mathbb{R}^{m \times p}, \mathbf{a}_j \in \mathcal{M}(\gamma; \mathbf{D}), j = 1, \dots, n\}$, where $\mathcal{M}(\gamma; \mathbf{D})$ is the model class family of sparse codes based on a fixed dictionary \mathbf{D} defined in Section 5, with the dependency on \mathbf{D} made explicit. It is easy to see that the model classes $\mathcal{M}(p)$ are nested. We now need to solve

$$(\mathbf{A}(p), \mathbf{D}(p)) = \arg \min_{(\mathbf{A}, \mathbf{D}) \in \mathcal{M}(p)} L(\mathbf{E}, \mathbf{A}, \mathbf{D}), \quad (\text{D.16})$$

for $p=0, 1, \dots$, and then choose $(\hat{\mathbf{A}}, \hat{\mathbf{D}}) = (\mathbf{A}(\hat{p}), \mathbf{D}(\hat{p}))$ with the optimal dictionary size

$$\hat{p} = \arg \min_p \{L(\mathbf{E}, \mathbf{A}(p), \mathbf{D}(p)) : p=0, 1, \dots\}.$$

As with sparse coding, here we exploit the nested nature of the model classes to speed up the model selection. For this, we propose a forward-selection algorithm, described in Algorithm 4, which starts from $\mathcal{M}(0)$ (the empty dictionary), and then approximates the best model in $\mathcal{M}(p+1)$ by adding a new atom to the dictionary computed for $\mathcal{M}(p)$ and

Algorithm 3: COdelength Minimizing Pursuit Algorithm (COMPA)

Input: Data sample \mathbf{y} , dictionary \mathbf{D}
Output: $\hat{\mathbf{a}}, \hat{\mathbf{e}}$
initialize $t \leftarrow 0; \mathbf{a}^{(0)} \leftarrow \mathbf{0}; \mathbf{e} \leftarrow \mathbf{y}; L^{(0)} \leftarrow L(\mathbf{y}, \mathbf{0}); \mathbf{g}^{(t)} \leftarrow \mathbf{D}^T \mathbf{e}^{(t)}$; // $\mathbf{g}^{(t)}$ correlation of current residual with the dictionary
repeat
 for $k \leftarrow 1, 2, \dots, p$ **do**
 $\Delta_k \leftarrow [\mathbf{g}_k^{(t)}]_{\delta_a}$; // step Δ_k is correlation, quantized to prec. δ_a
 $\tilde{\mathbf{L}}_k \leftarrow L([\mathbf{e} - \Delta_k \mathbf{d}_k]_{\delta_e}, \mathbf{a} + \Delta_k \omega_k)$; // $\omega_k = k$ -th canonical vec. of \mathbb{R}^p
 end
 $L^{(t+1)} \leftarrow \min\{\tilde{L}_k : k = 1, \dots, p\}$; // update coefficients vector
 $\mathbf{a}^{(t+1)} \leftarrow \mathbf{a}^{(t)} + \Delta_k \omega_k$; // update correlation
 $\mathbf{g}^{(t+1)} \leftarrow \mathbf{g}^{(t)} - \Delta_k \mathbf{d}_k$; // update correlation
 $t \leftarrow t + 1$;
until $L^{(t)} \geq L^{(t-1)}$;
 $\hat{\mathbf{a}} \leftarrow \mathbf{a}^{(t-1)}$;
 $\hat{\mathbf{e}} \leftarrow [\mathbf{y} - \mathbf{D}\hat{\mathbf{a}}]$;
STOP;

Algorithm 4: MDL-based dictionary learning via forward selection.

Input: Data \mathbf{Y}
Output: $(\hat{\mathbf{A}}, \hat{\mathbf{D}})$
initialize $p \leftarrow 0; \mathbf{A}(0) \leftarrow \emptyset; \mathbf{D}(0) \leftarrow \emptyset; \mathbf{E}(0) \leftarrow \mathbf{Y}; L(0) \leftarrow L(\mathbf{E}(0), \mathbf{A}(0), \mathbf{D}(0))$;
repeat
 $\tilde{\mathbf{d}} \leftarrow \mathbf{u}_1, \mathbf{U}\Sigma\mathbf{V}^T = \mathbf{E}(p)$ // Initial value of new atom is the left-eigenvector associated to the largest singular value of $\mathbf{E}^{(t)}$.
 $\mathbf{D}^0 \leftarrow [\mathbf{D}(p) | \tilde{\mathbf{d}}]$ // Initial dictionary for optimization below.
 $(\mathbf{A}(p+1), \mathbf{D}(p+1)) \leftarrow \arg \min_{(\mathbf{A}, \mathbf{D}) \in \mathcal{M}(p+1)} L(\mathbf{E}, \mathbf{A}, \mathbf{D})$ // Optimize dict. via Algorithm 5
 $p \leftarrow p + 1$;
 $L(p) \leftarrow L(\mathbf{E}(p), \mathbf{A}(p), \mathbf{D}(p))$;
until $L(p) \geq L(p-1)$;
 $\hat{\mathbf{A}} \leftarrow \mathbf{A}(p-1); \hat{\mathbf{D}} \leftarrow \mathbf{D}(p-1)$;

then invoking Algorithm 5, which is discussed in depth in the next subsection.

A backward-selection algorithm was also developed which first learns the model for $\mathcal{M}(p_{\max})$ via (5), where p_{\max} is a given maximum dictionary size, and then prunes the less frequently used atoms until no further decrease in codelength is observed. This algorithm allows us to provide especially-constructed initial dictionaries for Algorithm (5), e.g., an (overcomplete) DCT frame, which can be critical for finding good local minima of the non-convex problem (D.16). We do this for example to learn a dictionary for the whole class of natural images, see Section 7.

Algorithm 5: MDL-based dictionary learning for a given size p

Input: Data \mathbf{Y} , initial dictionary \mathbf{D}^0 , multiplier λ, η
Output: Local-optimum $(\hat{\mathbf{A}}, \hat{\mathbf{D}})$
initialize $\mathbf{D}^{(0)} = \mathbf{D}^0, t = 1$;
repeat
 for $j = 1, \dots, n$ **do**
 $\mathbf{a}_j^{(t)} \leftarrow \arg \min_{\mathbf{a}} L(\mathbf{e}, \mathbf{a}, \mathbf{D}^{(t-1)})$;
 end
 Update plug-in parameters: $\theta_\epsilon, \{(\theta_a^k, \rho_a^k), k = 1, \dots, p\}, \theta_d$;
 $\mathbf{D}^{(t)} \leftarrow \arg \min_{\mathbf{D}} L(\mathbf{E}, \mathbf{A}, \mathbf{D})$;
 $t \leftarrow t + 1$;
until $\frac{\|\mathbf{D}^{(t)} - \mathbf{D}^{(t-1)}\|_2}{\|\mathbf{D}^{(t)}\|_2} \leq \epsilon$;

6.1 Optimizing the dictionary for fixed p

For fixed p , and given an initial \mathbf{D} , Algorithm 5 adapts the atoms of \mathbf{D} to fit the training data \mathbf{Y} . At the high level, our algorithm is very similar to the traditional approach of alternate minimization over (\mathbf{A}, \mathbf{D}) . However, there are a number of important differences, namely: 1) The cost function minimized is now the cumulative codelength of describing \mathbf{Y} , $L(\mathbf{E}, \mathbf{A}, \mathbf{D})$; 2) Minimizing over \mathbf{A} is done sample by sample following Section 5; 3) Since \mathbf{D} needs to be described as well, it has an associated codelength (see Section 4.3), resulting in regularized dictionary update, described below; 4) in a cross-breed between Expectation-Maximization, and plug-in estimation, we estimate the model parameters for the current iterate $(\mathbf{E}^{(t)}, \mathbf{A}^{(t)}, \mathbf{D}^{(t)})$, from the accumulated statistics of previous iterates $\{(\mathbf{E}^{(t')}, \mathbf{A}^{(t')}, \mathbf{D}^{(t')}), t' = 1, \dots, t - 1\}$. At the end of the learning process, these parameters are “saved” as part of the learned model and can be used for modeling future data along with \mathbf{D} .

At the t -th iteration of the alternate minimization between \mathbf{D} and \mathbf{A} , with $\mathbf{A}^{(t)}$ just computed and kept fixed, the dictionary step consists of solving the sub-problem

$$\mathbf{D}^{(t)} = \arg \min_{\mathbf{D} \in \mathbb{R}^{m \times p}} L(\mathbf{Y}, \mathbf{A}^{(t)}, \mathbf{D}) = \arg \min_{\mathbf{D} \in \mathbb{R}^{m \times p}} L(\mathbf{Y}|\mathbf{A}^{(t)}, \mathbf{D}) + L(\mathbf{D}).$$

According to Section 4.3, we have $L(\mathbf{D}) = \frac{1}{\theta_d^{(t)}} \sum_{k=1}^p \|\mathbf{W}\mathbf{d}_k\|_1$, where

$$\theta_d^{(t)} = \frac{1}{mp} \sum_{k=1}^p \sum_{i=1}^m |d_{ik}^{(t-1)}|$$

is the Laplacian MLE of θ_d based on $\mathbf{D}^{(t-1)}$. Correspondingly, the data fitting term, via (D.9) and disregarding the constant terms, is given by $L(\mathbf{Y}|\mathbf{A}^{(t)}, \mathbf{D}) = L(\mathbf{Y} - \mathbf{D}\mathbf{A}^{(t)}|\theta_e^{(t)}, \sigma_e^2) = \sum_{j=1}^n \sum_{i=1}^m -\log LG(y_{ij} - (\mathbf{D}\mathbf{A}^{(t)})_{ij}; \theta_e^{(t)}, \sigma_e^2)$, where $\theta_e^{(t)}$ is the estimator of θ_e given $\mathbf{E} = \mathbf{Y} - \mathbf{D}^{(t-1)}\mathbf{A}^{(t)}$ (see Section 4.4) and σ_e^2 is assumed known. The problem can now be written as,

$$\mathbf{D}^{(t)} = \arg \min_{\mathbf{D}} L(\mathbf{Y} - \mathbf{D}\mathbf{A}^{(t)}|\theta_e^{(t)}, \sigma_e^2) + \theta_d^{(t)} \sum_{k=1}^p \|\mathbf{W}\mathbf{d}_k\|_1. \quad (\text{D.17})$$

For general \mathbf{W} , the optimization of (D.17) is challenging since none of the above terms are separable, in particular, the non-differentiable ℓ_1 term. However, since \mathbf{W} is easily invertible (as described in Section 4.3, it is lower triangular with 1's in the diagonal), we can perform a change of variables and solve the equivalent problem in the *prediction residual matrix* $\mathbf{U} = \mathbf{W}\mathbf{D}$ instead,

$$\hat{\mathbf{U}} = \arg \min_{\mathbf{U}} L(\mathbf{Y} - \mathbf{W}^{-1}\mathbf{U}\mathbf{A}^{(t)}|\theta_e^{(t)}, \sigma_e^2) + \theta_d^{(t)} \sum_{k=1}^p \|\mathbf{U}_k\|_1. \quad (\text{D.18})$$

Since the regularization term in (D.18) is decoupled in the elements of \mathbf{U} , and $L(\mathbf{Y} - \mathbf{W}^{-1}\mathbf{U}\mathbf{A}^{(t)}|\theta_e^{(t)}, \sigma_e^2)$ is convex and differentiable in \mathbf{U} (see Figure D.2(a)), (D.18) can be efficiently solved using the existing techniques for separable non-differentiable regularization terms. In our case, we employ the backtracking variant of FISTA [23], focusing on an efficient numerical evaluation of each step.

7 Experimental results

7.1 Coding performance

The first experiment in this section assesses the ability of our coding scheme to actually produce a compressed description of the data, in this case 8-bit gray-scale images. To this end, a dictionary \mathbf{D} was learned using the backward-selection algorithm, for the training samples from the Pascal’06 image database,⁷ converted to 8-bit gray-scale images and decomposed into 8×8 patches. The initial dictionary was an overcomplete DCT frame with $p = 256$. The resulting global dictionary \mathbf{D} has $p = 250$ atoms. We then encoded the testing samples from the same database, obtaining an average codelength of 4.1 bits per pixel (bpp), confirming the ability of our model to produce a compressed description of the data.

7.2 Learning performance

We compare the performance of the forward and backward dictionary learning algorithms proposed in Section 6 by applying each method to learn a dictionary for the standard “Boats” image (taken from the SIPI database,⁸ along with “Lena,” “Barbara” and “Peppers” used in the following experiments), and then measuring the final codelength and computation time. For the backward case, the initial dictionary is the global dictionary learned in the previous experiment. As for the forward method, we also include a faster “partial update” variant which performs a few (10) iterations of Algorithm 5 after adding a new atom, instead of allowing it to converge. The results were a compression level of 5.13bpp at a computational cost of 3900s for the backward method, 5.19bpp requiring 800s for the convergent forward method, and 5.22bpp requiring 150s for the partial forward method (the running times were measured for a parallelized C++ implementation running on an

⁷<http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html>

⁸<http://sipi.usc.edu/database/database.php?volume=misc&image=38\#top>

Athlon Phenom II X6 at 2.6GHz). In summary, all three methods reach similar, significant, compression levels. Slightly better results are obtained with the backward method, at the cost of a significant increase in computational time. On the other hand, the partial forward variant is significantly faster than the other two, yielding similar codelengths.

7.3 Denoising of natural images

The task in this case is to estimate a clean image from an observed noisy version whose pixels are corrupted by AWGN of known variance σ_e^2 . Here \mathbf{Y} contains all (overlapping) 8×8 patches from the noisy image. The denoising algorithm proceeds in two stages. In the first one, a dictionary \mathbf{D} is learned from the noisy image patches \mathbf{Y} . We use the backward selection algorithm since it allows us to use the global dictionary as the starting point, a common practice in this type of problems, [25, 27]. Secondly, the clean patches are estimated as sparse combinations of atoms from \mathbf{D} . In our case, the second stage admits two variants. The first one is a rate-distortion (RD) procedure akin to the traditional method used for example in [25], where each clean sample $\hat{\mathbf{y}}_j$ is estimated using a distortion-constrained formulation. In our case, we minimize the codelength (or “rate”) of describing \mathbf{y}_j up to a prescribed distortion proportional to the noise level, $\hat{\mathbf{y}}_j = \mathbf{D}\hat{\mathbf{a}}_j$, $\hat{\mathbf{a}}_j = \arg \min_{\mathbf{u}} L(\mathbf{u}) \quad \text{s.t.} \quad \|\mathbf{y}_j - \mathbf{D}\mathbf{u}\|_2 \leq C\sigma_N^2$. Here we use $C = 1.0$. The second variant, coined “post-thresholding” (PT) is more consistent with the learning phase, and is truly parameter-free, since the estimation derives from the same codelength minimization procedure used for learning the dictionary \mathbf{D} . In this case we obtain an initial estimate $\tilde{\mathbf{y}}_j = \mathbf{D}\tilde{\mathbf{a}}_j$, $\tilde{\mathbf{a}}_j = \arg \min_{\mathbf{u}} L(\mathbf{u}) + L(\mathbf{y}_j|\mathbf{u})$. However, according to the model developed in Section 4.2, the encoding residual $\tilde{\mathbf{e}} = \mathbf{y}_j - \tilde{\mathbf{y}}_j$ may contain a significant portion of clean data due to modeling errors. We can then think of $\tilde{\mathbf{e}}$ as clean data corrupted by noise of variance σ_e^2 . To extract the clean portion, we solve another codelength-minimization sub-problem, this time with a Gaussian prior for the error, and a Laplacian prior for the clean

part, $\bar{\mathbf{e}}_j = \arg \min_{\mathbf{u}} \frac{1}{\sigma_e^2} \|\tilde{\mathbf{e}}_j - \mathbf{u}\|_2 + \frac{1}{\hat{\theta}_e} \|\mathbf{u}\|_1$, where $\hat{\theta}_e = \sqrt{0.5 \max\{0, \text{var}(\tilde{\mathbf{e}}_j) - \sigma_e^2\}}$, following Section 4.4. We then compute the final estimate as $\hat{\mathbf{y}}_j = \tilde{\mathbf{y}}_j + \bar{\mathbf{e}}_j$. In either variant, the model used for $L(\mathbf{a})$ includes the Markovian dependency between the occurrence of each atom in a patch and its previously-encoded neighbors, as described in Section 4.4.

Denoising performance is summarized in Figure D.6, along with a detail of the result for $\sigma_e = 10$ for the “Boats” image in Figure D.6. In all cases, there is a 1 to 5 dB improvement over the best MDL-based results in [15], thus showing the relevance of overcoming the limitations in previous MDL applications to sparse coding. Both the RD and PT methods yield results which are comparable to those of [25], which depend significantly on several carefully tuned parameters.⁹ While the RD variant performs better than PT in terms of PSNR, PT is faster and tends to produce less artifacts than RD, thus resulting in more visually pleasant images than RD. This, which can be clearly seen in Figure D.6, occurs in all other cases as well. Including the Markov dependency in $L(\mathbf{a})$ produced an average improvement of up to 0.2dB.

7.4 Texture mosaic segmentation

Here we are given c images with sample textures, and a target mosaic of textures,¹⁰ and the task is to assign each pixel in the mosaic to one of the textures. Again, all images are decomposed into overlapping patches. This time a different dictionary is learned for each texture using patches from corresponding training images. In order to capture the texture patterns, a patch width $w = 16$ was used. Then, each patch in the mosaic is encoded using all available dictionaries, and its center pixel is assigned to the class which produced the shortest description length for that patch.

This seemingly natural procedure results in a success rate of 77%, which is consistent

⁹To the best of our knowledge, these results, as well as those in [25], are among the best that can be obtained for gray-scale images without using multi-scale and/or spatial aggregation of patches as in [39, 40].

¹⁰Taken from <http://www.ux.uis.no/~tranden/>.

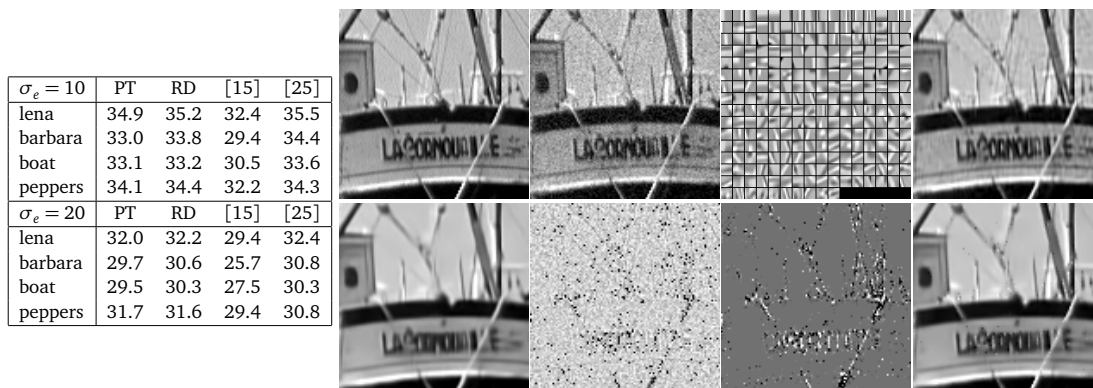


Figure D.6: Denoising results. Left table: denoising performance, in PSNR, of K-SVD [25], MDL denoising [15], and the Post-Thresholding (PT) and Rate-Distortion (RD) denoising variants. Images, top row: clean “Boats”, noisy version, learned dictionary for this image (final $p = 248$), image recovered using RD. Images, bottom row: image reconstructed from the initial estimation \hat{y}_j obtained in the PT method, its residual, portion of residual that was added back, final PT estimation.

with the second picture of Figure D.7. The problem is that this procedure is inconsistent with the learning formulation, because each dictionary is adapted to minimize the *average* codelength of describing each patch in the respective texture. Therefore, good results can only be expected if the decision is made for groups of patches simultaneously, that is, by considering the cumulative codelength of a set of patches. We implement this by deciding on each patch on the basis of comparing the average codelength obtained with each dictionary for encoding that patch and all patches in a circular neighborhood with a radius of 20 pixels. The success rate in this case is 95.3%, which is comparable to the state-of-the art for this type of problems (see for example [41], which learns sparse models for explicitly maximizing the success rate). The Markovian model improved our results by 1%.

7.5 Low-rank matrix approximation

The low-rank matrix approximation family of problems (see [42] for a review) can be seen as an extension to the problem of sparse coding where sparsity is substituted by matrix rank. Concretely, the task is to recover a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ from an incomplete and/or

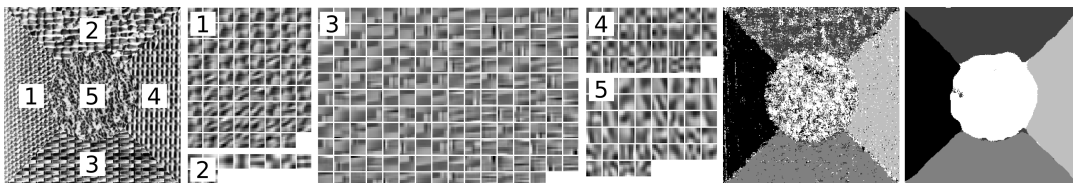


Figure D.7: Left to right: Texture mosaic, dictionaries learned for each class (note the automatically learned different sizes), patch-wise codelength-based classification map –each shade of gray corresponds to a texture class – (77.0% success rate), classification map obtained by averaging the codelength over a neighborhood of patches (95.4% success rate).

corrupted observation \mathbf{Y} , under the assumption that the rank of \mathbf{A} , $\text{rank}(\mathbf{A})$, is small. As with sparse coding, $\text{rank}(\mathbf{A})$ is relaxed using the ℓ_1 equivalent for matrix rank, which is the nuclear norm, $\|\mathbf{A}\|_* := \sum_i \sigma_i(\mathbf{A})$, where $\sigma_i(\mathbf{A})$ is the i -th singular value of \mathbf{A} . It has been shown in [42] that, under certain assumptions on $\text{rank}(\mathbf{A})$, the following estimation function is able to recover \mathbf{A} from a noisy observation \mathbf{Y} , and with a significant fraction of its coefficients arbitrarily corrupted,

$$\hat{\mathbf{A}} = \arg \min_{\mathbf{W}} \|\mathbf{W}\|_* + \lambda \|\mathbf{Y} - \mathbf{W}\|_1, \quad \lambda = 1/\sqrt{\max\{m, n\}}. \quad (\text{D.19})$$

A common proof of concept is to use this framework for robust background estimation in camera surveillance video sequences [43], and we apply our proposed framework for the same application.

To perform our MDL-based model selection within this formulation, we solve (D.19) for increasing values of λ , obtaining a low-rank approximation to \mathbf{A} , $(\mathbf{A}(\lambda), \mathbf{E}(\lambda) = \mathbf{Y} - \mathbf{A}(\lambda))$, which we encode using the universal models described in Section 4. We modified the algorithm described in [44] to allow for warm restarts, using the solution for the previous λ as a starting point for the next λ for faster convergence.

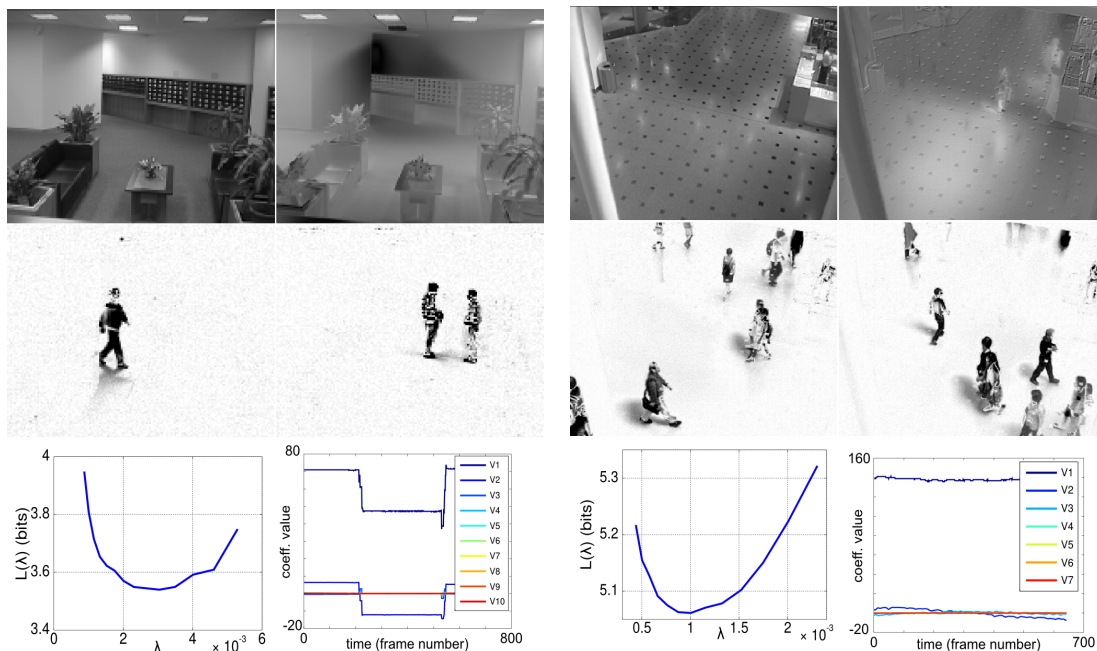
Consistently with the ℓ_1 fitting term of (D.19), we encode the non-zero values of $\mathbf{E}(\lambda)$ as a Laplacian sequence of unknown parameter. To exploit the potential sparsity in $\mathbf{E}(\lambda)$,

the locations of the non-zero values are encoded, as in 4.1, using an enumerative two-parts code for Bernoulli sequences of unknown parameter. To exploit low-rank in the encoding, the matrix $\mathbf{A}(\lambda)$ is encoded via its reduced SVD decomposition $\mathbf{A}(\lambda) = \mathbf{U}(\lambda)\Sigma(\lambda)\mathbf{V}(\lambda)^\top$. For $\text{rank}(\mathbf{A}(\lambda)) = r$, we have that $\mathbf{U}(\lambda) \in \mathbb{R}^{m \times r}$ are the left-eigenvectors, $\Sigma \in \mathbb{R}^{r \times r}$ is the diagonal matrix whose diagonal are the non-zero singular values of $\mathbf{A}(\lambda)$, and $\mathbf{V}(\lambda) \in \mathbb{R}^{r \times n}$ are the right-eigenvectors of $\mathbf{A}(\lambda)$. Each column of \mathbf{U} is encoded (in this video example) as a smooth image via a causal bilinear predictor identical to the one used for predictive coding of \mathbf{D} in 4.3, using a Laplacian model for the prediction residuals. Each column of \mathbf{V} is encoded as a smooth one-dimensional sequence, using a zero order predictor (the predicted value for the next coefficient is the previous coefficient value), with a Laplacian prior on the prediction residuals. Finally, the values of Σ , which can be arbitrary, are quantized and encoded using the universal code for integers [45].

The encoding method is very simple, with all unknown parameters encoded using a two-parts code, and codelengths for the discretized Laplacian pre-computed in look-up tables. Quantization for this case is as follows: the codelength associated with the r non-zero singular values is negligible, and we minimize unwanted distortion encoding them with high precision ($1e-16$). As for the columns of \mathbf{U} and \mathbf{V} , they all have unit norm, so that the average magnitude of their elements are close to $\sqrt{1/m}$ and $\sqrt{1/n}$ respectively. Based on this, our algorithm encodes the data with $\delta_u = Q/\sqrt{m}$ as the precision for encoding \mathbf{U} , and $\delta_v = Q/\sqrt{n}$ for \mathbf{V} , for several values of Q in $(0, 1)$, keeping the one producing the smallest codelength. The MDL-based estimation algorithm then chooses the model for which the codelength $L(\mathbf{Y}; \lambda) = L(\mathbf{U}(\lambda)) + L(\Sigma(\lambda)) + L(\mathbf{V}(\lambda))$ is minimized.

As in [43], here we show results for two sequences taken from [46]: “Lobby” (Figure D.8(a)), and “ShoppingMall” (Figure D.8(b)). Full videos can be viewed at <http://www.tc.umn.edu/~nacho/lowrank/>.

In both cases, the recovered backgrounds are very accurate. In particular, for the Lobby



(a) Results for “Lobby” sequence, featuring a room with lights that are switched off and on. The rank of the approximation for this case is rank = 10. The moment where the lights are turned off is clearly seen here as the “square pulse” in the middle of the first two right-eigenvectors (bottom-right). Also note how \mathbf{u}_2 (top-right) compensates for changes in shadows.

(b) Results for “ShoppingMall”, a fixed camera looking at a crowded hall. In this case, the rank of the approximation decomposition is rank = 7. Here, the first left-eigenvector models the background, whereas the rest tend to capture people that stood still for a while. Here we see the “phantom” of two such persons in the second left-eigenvector (top-right).

Figure D.8: Low-rank approximation results. Both figures show the first two left-eigenvectors as 2D images at the top, two sample frames from the approximation error sequences in the middle, which should contain the people that were removed from the videos, and the curve $L(\lambda)$ and the right-eigenvalues, scaled by Σ (representing the “activity” of each left-eigenvector along time), at the bottom.

sequence, the selected model captures just the eigenvectors needed to recover the background along with its lighting changes, including corrections for local shadows, leaving out only the people passing by.

8 Concluding remarks

We have presented an MDL-based sparse modeling framework, which automatically adapts to the inherent complexity of the data at hand using codelength as a metric.

The framework features a sparse coding algorithm and automatic tuning of the sparsity level on a per-sample basis, including a sequential collaborative variant which adapts the model parameters as it processes new samples, and two dictionary learning variants which learn the size of the dictionaries from the data. In all cases, the information-theoretic formulation led to robust coding and learning formulations, including novel robust metrics for the fitting term (LG and MOEG), and robust ℓ_1 -based dictionary regularization term. This formulation also allowed us to easily incorporate more prior information into the coding/learning process, such as Markovian spatial dependencies, by means of simple modifications to the probability models used.

As a result, the framework can be applied out-of-the-box to very different applications, from image denoising to low-rank matrix approximation, obtaining competitive results in all the cases presented, with minimal interaction from the user.

9 Supplementary material

9.1 MDL-based sparse coding via convex relaxation

Here we present an alternative method for performing efficient sparse model selection in the sparse coding problem of choosing the best \mathbf{a} for a given data sample \mathbf{y} from the nested

Algorithm 6: COdelength-Minimizing COntinuation ALgorithm (COMICAL)

Input: Data sample \mathbf{y} , dictionary \mathbf{D} , $\Delta_\lambda > 0$; $h > 0$
Output: The sparse code for \mathbf{y} , \mathbf{a}
initialize $\lambda = \min\{\|\mathbf{D}^\top \mathbf{y}\|_\infty, h\}$;
initialize $\mathbf{e}(\lambda) \leftarrow \mathbf{y}$; $\mathbf{z}(\lambda) \leftarrow \mathbf{0}$; $\gamma \leftarrow 0$; $\mathbf{a}(\lambda) \leftarrow \mathbf{0}$; $L_0 \leftarrow +\infty$;
while $\lambda \geq 0$ **do**
 repeat
 $\lambda \leftarrow \lambda - \Delta_\lambda$;
 $\tilde{\mathbf{a}} \leftarrow \arg \min_{\mathbf{u} \in \mathbb{R}^p} \mathcal{H}(\mathbf{y} - \mathbf{D}\mathbf{u}; h) + \lambda \|\mathbf{u}\|_1$. $\tilde{\mathbf{z}} \leftarrow \text{supp}(\tilde{\mathbf{a}})$;
 until $\tilde{\mathbf{z}} > \gamma$;
 $\gamma \leftarrow \sum \tilde{\mathbf{z}}$;
 $\mathbf{a}(\gamma)_{[\tilde{\mathbf{z}}^c]} \leftarrow \mathbf{0}$;
 $\mathbf{a}(\gamma)_{[\tilde{\mathbf{z}}]} \leftarrow \left[(\mathbf{D}_{[\tilde{\mathbf{z}}]})^\dagger \mathbf{y} \right]_{\delta_a}$;
 $\mathbf{e}(\gamma) \leftarrow [\mathbf{y} - \mathbf{D}\mathbf{a}(\gamma)]_{\delta_e}$;
 $\mathbf{z}(\gamma) \leftarrow \tilde{\mathbf{z}}$; // Note: actual support may be smaller due to quantization.
 $L(\gamma) \leftarrow L(\mathbf{y}(\gamma), \mathbf{a}(\gamma))$;
 if $L(\gamma) < L_0$ **then**
 $L_0 \leftarrow L(\gamma)$;
 else
 STOP ;
 end
end

class of possible sparse codes based on a fixed dictionary \mathbf{D} ,

$$\mathcal{M}_{\mathbf{D}} := \bigcup_{\gamma=0}^p \mathcal{M}_{\mathbf{D}}(\gamma), \quad \mathcal{M}_{\mathbf{D}}(\gamma) := \{\mathbf{a} \in \mathbb{R}^p : \|\mathbf{a}\|_0 \leq \gamma\}.$$

This method, called COdelength-MINimizing COntinuation ALgorithm (COMICAL), is detailed in Algorithm 6. The idea is to estimate the best code $\mathbf{a}(\gamma)$ from $\mathcal{M}_{\mathbf{D}}(\gamma)$ sequentially for $\gamma = 0, \dots, p$ by solving a convex approximation of the codelength function $L(\mathbf{y}, \mathbf{a})$ (here \mathbf{D} is ignored, as it is fixed). Disregarding quantization and constant terms, this has the following form,

$$L(\mathbf{y}, \mathbf{a}) = -\log p(\mathbf{e}) + \log \binom{p}{\|\mathbf{a}\|_0} + \|\mathbf{a}\|_0 - \log p(\mathbf{a}_{[\mathbf{z}]}) - \|\mathbf{a}\|_0 \log \delta_a, \quad (\text{D.20})$$

where the second term corresponds to encoding the support \mathbf{z} , the third term to encoding the sign \mathbf{s} , and the last two to encoding the non-zero values of \mathbf{a} . To obtain an approximate minimizer $\hat{\mathbf{a}}$ of the non-convex codelength function (D.20) we proceed in two stages. First, we obtain a sequence of candidate supports of increasing size $\{\mathbf{z}(\gamma) : \gamma = 1, \dots, p\}$ from which we compute the candidate codes from each class $\mathcal{M}(\gamma)$, $\{\mathbf{a}(\gamma) : \gamma = 1, \dots, p\}$ by finding the values of the non-zero coefficients, $\mathbf{a}_{\mathbf{z}(\gamma)}(\gamma)$.

For the sub-problem of selecting the set of optimal supports, we replace all occurrences of the ℓ_0 pseudo-norm, including the term $\log \binom{p}{\|\mathbf{a}\|_0}$, by a convex approximation $\lambda \|\mathbf{a}\|_1$, and approximate the ideal codelength function $-\log p(\mathbf{e})$ with the *Huber loss function*,

$$\mathcal{H}(e; h) := \begin{cases} \frac{1}{2}e^2 & , |e| < h \\ h|e| - \frac{h^2}{2} & , |e| \geq h \end{cases} ,$$

which is a good simple approximation of the LG codelength function (see Figure D.9), and $-\log p(\mathbf{a}|\mathbf{z}, \mathbf{s})$ with the ℓ_1 norm of \mathbf{a} , which then collapses with the other two ℓ_1 norms into $\lambda \|\mathbf{a}\|_1$. We extend the Huber loss function $\mathcal{H}(\mathbf{e}; h)$ to vectors $\mathbf{e} \in \mathbb{R}^m$ as $\mathcal{H}(\mathbf{e}; h) := \sum_{i=1}^m \mathcal{H}(e_i; h)$. The resulting convex problem has the following form,

$$\arg \min_{\mathbf{u} \in \mathbb{R}^p} \mathcal{H}(\mathbf{y} - \mathbf{D}\mathbf{u}; h) + \lambda \|\mathbf{u}\|_1 . \quad (\text{D.21})$$

In order to obtain a sequence of solutions $\mathbf{a}(\gamma)$ for increasing γ , we modulate the value of the parameter λ from the value λ_{\max} that produces the solution $\mathbf{a}(0) = \mathbf{0}$ (which is easily obtained using the KKT conditions of the problem, see below) down to 0. The Huber parameter h is kept fixed to a value of $h = 5\sigma_e$ to avoid performing the search over a two-dimensional parameter space. This value was chosen by inspecting the shape of the LG codelength function and its best Huber approximation (see Figure D.9) for the range

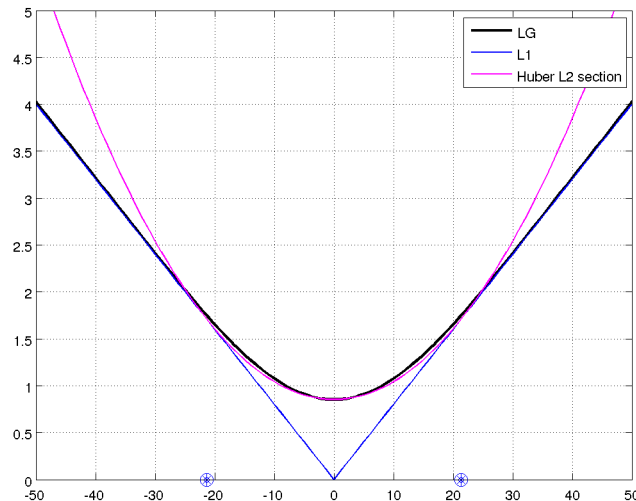


Figure D.9: Huber approximation of the LG codelength function. Here we show the ℓ_1 and ℓ_2 regions of the Huber loss function that better fits the LG codelength shown here. The regions are delimited by the value of the Huber parameter h marked here as blue starred dots on the horizontal axis.

of values of θ_e and σ_e typically observed (for example, from the output of the pursuit-based COMPA algorithm described previously). Problem (D.21) is a minor variation of the Lasso problem, that we solve with a minor variation of Coordinate Descent (CD) [22]. The fact that the Huber function $\mathcal{H}(\cdot; h)$ is smooth everywhere is crucial here to guarantee convergence of CD to the global optimum of (D.21) for each value of λ (see [47]). In fact, the only difference between CD and our algorithm is the thresholding function induced by the Huber- ℓ_1 scalar function,

$$\mathcal{S}_{\mathcal{H}}(u; h, \lambda) := \operatorname{argmin} \mathcal{H}(u; h) + \lambda|u|. \quad (\text{D.22})$$

It turns out that the solution to this problem coincides with the soft-thresholding operator [21] if $h > \lambda$, and otherwise is *identically zero*,

$$\mathcal{S}_{\mathcal{H}}(u; h, \lambda) = \begin{cases} \max\{0, |u| - \lambda\} & , \lambda < h \\ 0 & , \text{otherwise} \end{cases}$$

Therefore, the only difference between CD and our algorithm is the value of λ_{\max} , which is $\lambda_{\max}^{\text{Huber-}\ell_1} = \min\{\lambda_{\max}^{\ell_2-\ell_1}, h\}$.

The efficiency of our algorithm derives from the use of a continuation strategy, meaning that the problem is solved repeatedly for small decrements of λ , and the solution of the problem for the previous value of λ is used as a starting point (a *warm restart*) for solving the next problem. For sufficiently small decrements of λ , two consecutive solutions are very close in space, so that convergence can be achieved very quickly (even only one iteration of CD may be enough).

Given the estimated support $\mathbf{z}(\gamma)$, the non-zero values of $\mathbf{a}(\gamma)$, $\mathbf{a}_{[\mathbf{z}(\gamma)]}(\gamma)$ are computed as the quantized orthogonal projection of the data vector \mathbf{y} onto the active dictionary atoms $\mathbf{D}_{[\mathbf{z}(\gamma)]}$,

$$\mathbf{a}_{[\mathbf{z}(\gamma)]}(\gamma) = \left[\left(\mathbf{D}_{[\mathbf{z}]}^{\top} \mathbf{D}_{[\mathbf{z}]} \right)^{-1} \mathbf{D}_{[\mathbf{z}]}^{\top} \mathbf{y} \right]_{\delta_a}. \quad (\text{D.23})$$

Note that, in principle, given the fixed support $\mathbf{z}(\gamma)$, finding a local minimum (or global, if $-\log p(\mathbf{e})$ and $-\log p(\mathbf{a}_{[\mathbf{z}]})$ are convex), is now possible. However, we avoid this as performing such full optimization for each $\gamma = 0, \dots, p$ is computationally too expensive, and the potential codelength gains are small, as it is $L(\mathbf{z})$ that usually dominates the description length of $L(\mathbf{a}) = L(\mathbf{z}) + L(\mathbf{s}) + L(\mathbf{a}_{[\mathbf{z}]})$ (see Figure D.5).

9.2 ℓ_1 regularized dictionary update

Algorithm 7 details the FISTA-based dictionary update algorithm that we propose for solving the ℓ_1 -regularized dictionary update problem. There, we define $L'(\mathbf{E}; \theta_e, \sigma_e^2)$ as the gradient of $L(\mathbf{E}; \theta_e, \sigma_e^2)$ with respect to \mathbf{E} , that is, $\{L'(\mathbf{E}; \theta_e, \sigma_e^2)\}_{ij} = L'(e_{ij}; \theta_e, \sigma_e^2)$ (see Figure D.2(f)).

Algorithm 7: Dictionary update (FISTA)

Input: $\alpha_0 > 0$; $0 < \beta < 1$; $\mathbf{D}^{(0)} \in \mathbb{R}^{m \times p}$; Parameters from previous iteration: $\theta_d, \theta_e, \sigma_e^2$

Output: Dictionary \mathbf{D}

Initialize $\mathbf{U}^{(0)} \leftarrow \mathbf{W}\mathbf{D}^{(0)}$; $t \leftarrow 1$; $\mathbf{V}^{(t)} \leftarrow \mathbf{U}^{(0)}$; $\tau^{(t)} \leftarrow 1$;

while $\alpha > \alpha_{\min}$ **do**

 // Backtracking (find valid α):

$\alpha \leftarrow \alpha_0$;

$\mathbf{Z} \leftarrow \mathcal{S}_{\theta_e}(\mathbf{V}^{(t)} - (\mathbf{P}^{-1})^\top L'(\mathbf{Y} - \mathbf{P}^{-1}\mathbf{V}^{(t)}\mathbf{A}; \theta_e, \sigma_e^2) \mathbf{A}^\top)$;

$F \leftarrow L(\mathbf{Y} - \mathbf{P}^{-1}\mathbf{Z}\mathbf{A}; \theta_e, \sigma_e^2) + \theta_e \sum_{k=1}^p \|\mathbf{Z}_k\|_1$;

$G_\alpha \leftarrow L(\mathbf{Y} - \mathbf{P}^{-1}\mathbf{V}\mathbf{A}; \theta_e, \sigma_e^2) + \langle \mathbf{Z} - \mathbf{V}, \nabla f(\mathbf{V}) \rangle + \frac{\alpha}{2} \|\mathbf{Z} - \mathbf{V}\|_F^2 + \mathbf{g}(\mathbf{Z})$;

while $\alpha > \alpha_{\min}$ AND $F > G_\alpha$ **do**

$\alpha \leftarrow \alpha\beta$;

$\mathbf{Z} \leftarrow \mathcal{S}_{\theta_e}(\mathbf{V}^{(t)} - \alpha(\mathbf{P}^{-1})^\top L'(\mathbf{Y} - \mathbf{P}^{-1}\mathbf{V}^{(t)}\mathbf{A}; \theta_e, \sigma_e^2) \mathbf{A}^\top)$;

$F \leftarrow L(\mathbf{Y} - \mathbf{P}^{-1}\mathbf{Z}\mathbf{A}; \theta_e, \sigma_e^2) + \theta_e \sum_{k=1}^p \|\mathbf{Z}_k\|_1$;

$G_\alpha \leftarrow L(\mathbf{Y} - \mathbf{P}^{-1}\mathbf{V}\mathbf{A}; \theta_e, \sigma_e^2) + \langle \mathbf{Z} - \mathbf{V}, \nabla f(\mathbf{V}) \rangle + \frac{\alpha}{2} \|\mathbf{Z} - \mathbf{V}\|_F^2 + \mathbf{g}(\mathbf{Z})$;

end

$\mathbf{U}^{(k)} \leftarrow \mathbf{Z}$ // Assign new iterate.

 // Update auxiliary point.

$\tau^{(t+1)} \leftarrow \frac{1 + \sqrt{1 + 4\tau^{(t)}}}{2}$;

$\mathbf{V}^{(t+1)} \leftarrow \mathbf{U}^{(t)} + \left(\frac{\tau^{(t)} - 1}{\tau^{(t+1)}}\right) (\mathbf{U}^{(t)} - \mathbf{U}^{(t-1)})$;

$t \leftarrow t + 1$;

end

$\hat{\mathbf{D}} \leftarrow \mathbf{W}^{-1}\mathbf{U}^{(t)}$;

STOP;

Practical speedup: although the ℓ_1 -regularized model is statistically more accurate, it also makes the dictionary update algorithm significantly slower than its traditional non-regularized counterparts. To remedy this situation, we optionally perform a few iterations

using a ℓ_2 -regularized model instead,

$$\hat{\mathbf{D}} = \arg \min_{\mathbf{D}} \frac{1}{2} \|\mathbf{Y} - \mathbf{DA}\|_F^2 + \sum_{k=1}^p \frac{\sigma_d^2}{2} \|\mathbf{P}\mathbf{d}_k\|_2^2, \quad (\text{D.24})$$

which can be efficiently solved using block-coordinate descent in the dictionary atoms, each atom \mathbf{d}_k updated via scaled projected gradient, which consists of the following steps,

$$\begin{aligned} i) \quad \mathbf{u} &\leftarrow \frac{1}{(\mathbf{AA}^\top)_{kk} + \sigma_d^2 \text{diagdiag}(\mathbf{P}^\top \mathbf{P})} (\mathbf{DA}^{(t)}((\mathbf{A}^{(t)})^\top)_k - \mathbf{Y}((\mathbf{A}^{(t)})^\top)_k) + \mathbf{P}^\top \mathbf{P}\mathbf{d}_k^{(t)} \\ ii) \quad \mathbf{d}_k^{(t+1)} &\leftarrow \frac{1}{\min\{1, \|\mathbf{u}\|_2\}} \mathbf{u}, \end{aligned}$$

where $\mathbf{A}^{(t)}$ is the matrix of current sparse coefficients, $\mathbf{d}_k^{(t)}$ is the current k -th atom, and $\mathbf{d}_k^{(t+1)}$ is the updated atom.

References

- [1] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, Feb. 2009.
- [2] R. Rubinstein, A. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, June 2010.
- [3] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, June 2010.
- [4] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin. Nonparametric Bayesian dictionary learning for analysis of noisy and incomplete images. submitted to *IEEE Trans. Image Processing*, 2011.

- [5] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.
- [6] G. Schwartz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.
- [7] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
- [8] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Trans. IT*, 30(4):629–636, 1984.
- [9] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Trans. IT*, 44(6):2743–2760, 1998.
- [10] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 2 edition, 2006.
- [11] N. Saito. Simultaneous noise suppression and signal compression using a library of orthonormal bases and the MDL criterion. In E. Foufoula-Georgiou and P. Kumar, editors, *Wavelets in Geophysics*, pages 299–324. New York: Academic, 1994.
- [12] H. Krim and J.-C. Pesquet. On the statistics of best bases criteria. Technical report, MIT LIDS, 1995.
- [13] P. Moulin and J. Liu. Analysis of multiresolution image denoising schemes using generalized-Gaussian and complexity priors. *IEEE Trans. IT*, April 1999.
- [14] J. Rissanen. MDL denoising. *IEEE Trans. IT*, 46(7):2537–2543, 2000.
- [15] T. Roos, P. Myllymäki, and J. Rissanen. MDL denoising revisited. *IEEE Trans. SP*, 57(9):3347–3360, 2009.

- [16] I. Ramírez and G. Sapiro. Sparse coding and dictionary learning based on the MDL principle. In *Proc. of ICASSP 2011, Prague, Czech. Republic*, May 2011.
- [17] P. J. Huber. Robust estimation of a location parameter. *Annals of Statistics*, 53:73–101, 1964.
- [18] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Trans. SP*, 41(12):3397–3415, 1993.
- [19] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.
- [20] E. J. Candès. Compressive sampling. *Proc. of the International Congress of Mathematicians*, 3, Aug. 2006.
- [21] D. L. Donoho and I.M Johnston. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, September 1994.
- [22] J. Friedman, H. Hastie, and R. Tibshirani. Regularized paths for generalized linear models via coordinate descent. *Journal of Stat. Soft.*, 33(1), 2008.
- [23] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.
- [24] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.
- [25] M. Aharon, M. Elad, and A. Bruckstein. The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Trans. SP*, 54(11):4311–4322, Nov. 2006.
- [26] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

- [27] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Trans. IP*, 17(1):53–69, Jan. 2008.
- [28] P. Grünwald. *The Minimum Description Length Principle*. MIT Press, June 2007.
- [29] Y. Shtarkov. Universal sequential coding of single messages. *Probl. Inform. Transm.*, 23(3):3–17, July 1987.
- [30] H. Ishwaran and J. S. Rao. Spike and slab variable selection: Frequentist and Bayesian strategies. *Annals of Statistics*, 33(2):730–773, 2005.
- [31] T. M. Cover. Enumerative source encoding. *IEEE Trans. IT*, 19(1):73–77, 1973.
- [32] I. Ramírez and G. Sapiro. Universal regularizers for robust sparse coding and modeling. Submitted. Preprint available in <http://arxiv.org/abs/1003.2941> arXiv:1003.2941v2 [cs.IT], August 2010.
- [33] G. Motta, E. Ordentlich, I. Ramirez, G. Seroussi, and M. Weinberger. The iDUDE framework for grayscale image denoising. *IEEE Trans. IP*, 20(1):1–21, Jan. 2011.
- [34] E. Lam and J. Goodman. A mathematical analysis of the DCT coefficient distributions for images. *IEEE Trans. IP*, 9(10):1661–1666, 2000.
- [35] R. E. Krichevsky and V. K. Trofimov. The performance of universal encoding. *IEEE Trans. IT*, 27(2):199–207, 1981.
- [36] M. Zhou, H. Yang, G. Sapiro, D. Dunson, and L. Carin. Dependent hierarchical beta process for image interpolation and denoising. In *AISTATS*, 2011.
- [37] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

- [38] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2nd edition, Feb. 2009.
- [39] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3d transform-domain collaborative filtering. *IEEE Trans. IP*, 16(8):2080–2095, August 2007.
- [40] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *ICCV*, pages 2272–2279, Oct. 2009.
- [41] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *Proc. IEEE CVPR*, Anchorage, USA, June 2008.
- [42] E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3), May 2011.
- [43] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. In *Adv. NIPS*, Dec. 2009.
- [44] Z. Lin, M. Chen, and Y. Ma. The Augmented Lagrange Multiplier Method for exact recovery of corrupted low-rank matrices. <http://arxiv.org/abs/1009.5055>.
- [45] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.
- [46] L. Li, W. Huang, I. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Trans. IP*, 13(11):1459–1472, 2004.
- [47] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Opt. Theory and App.*, 109(3):475–494, June 2001.

E

Low-rank data modeling via the Minimum Description Length principle

Ignacio Ramírez and Guillermo Sapiro

Department of Electrical and Computer Engineering, University of Minnesota

Robust low-rank matrix estimation is a topic of increasing interest, with promising applications in a variety of fields, from computer vision to data mining and recommender systems. Recent theoretical results establish the ability of such data models to recover the true underlying low-rank matrix when a large portion of the measured matrix is either missing or arbitrarily corrupted. However, if low rank is not a hypothesis about the true nature of the data, but a device for extracting regularity from it, no current guidelines exist for choosing the rank of the estimated matrix. In this work we address this problem by means of the Minimum Description Length (MDL) principle – a well established information-theoretic approach to statistical inference – as a guideline for selecting a model for the data at hand. We demonstrate the practical usefulness of our formal approach with results for complex background extraction in video sequences.

1 Introduction

The key to success in signal processing applications often depends on incorporating the right prior information about the data into the processing algorithms. In matrix estimation, low-rank is an all-time popular choice, with analysis tools such as Principal Component Analysis (PCA) dominating the field. However, PCA estimation is known to be non-robust,

and developing robust alternatives is an active research field (see [1] for a review on low-rank matrix estimation). In this work, we focus on a recent robust variant of PCA, coined RPCA [1], which assumes that the difference between the observed matrix \mathbf{Y} , and the true underlying data \mathbf{X} , is a sparse matrix \mathbf{E} whose non-zero entries are arbitrarily valued. It has been shown in [1] that \mathbf{X} (alternatively, \mathbf{E}) can be recovered exactly by means of a convex optimization problem involving the rank of \mathbf{Y} and the ℓ_1 norm of \mathbf{E} . The power of this approach has been recently demonstrated in a variety of applications, mainly computer vision (see [2] and <http://perception.csl.uiuc.edu/matrix-rank/applications.html> for examples).

However, when used as a pure data modeling tool, with no assumed “true” underlying signal, the rank of \mathbf{X} in a PCA/RPCA decomposition is a parameter to be tuned in order to achieve some desired goal. A typical case is *model selection* [3, Chapter 7], where one wants to select the *size* of the model (in this case rank of the approximation) in order to strike an optimal balance between the ability of the estimated model to generalize to new samples, and its ability to adapt itself to the observed data (the classic overfitting/underfitting trade-off in statistics). The main issue in model selection is how to formulate this balance as a cost function.

In this work, we address this issue via the Minimum Description Length (MDL) principle [4, 5].¹ MDL is a general methodology for assessing the ability of statistical models to capture regularity from data. The MDL principle can be regarded as a practical implementation of the Occam’s razor principle, which states that, given two descriptions for a given phenomenon, the shorter one is usually the best. In a nutshell, MDL equates “ability to capture regularity” with “ability to compress” the data, using *code length* or *compressibility* as the metric for measuring candidate models.

The resulting framework provides a robust, parameter-free low-rank matrix selection

¹While here we address the matrix formulation, the developed framework is applicable in general, including to sparse models, and such general formulation will be reported in our extended version of this work.

algorithm, capable of capturing relevant low-rank information in the data, as in video sequences from surveillance cameras in the illustrative application here reported. From a theoretical standpoint, this brings a new, information theoretical perspective into the problem of low-rank matrix completion. Another important feature of an MDL-based framework such as the one here presented is that new prior information can be naturally and easily incorporated into the problem, and its effect can be assessed *objectively* in terms of the different codelengths obtained.

2 Low-rank matrix estimation/approximation

Under the low-rank assumption, a matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$ can be written as $\mathbf{Y} = \mathbf{X} + \mathbf{E}$, where $\text{rank}(\mathbf{X}) \ll \min\{m, n\}$ and $\|\mathbf{E}\| \ll \|\mathbf{Y}\|$, where $\|\cdot\|$ is some matrix norm. Classic PCA provides the best rank- k approximation to \mathbf{Y} under the assumption that \mathbf{E} is a random matrix with zero-mean IID Gaussian entries,

$$\hat{\mathbf{X}} = \arg \min_{\mathbf{W}} \|\mathbf{Y} - \mathbf{W}\|_2, \quad \text{s.t.} \quad \text{rank}(\mathbf{W}) \leq k. \quad (\text{E.1})$$

However, PCA is known to be non-robust, meaning that the estimate $\hat{\mathbf{X}}$ can vary significantly if only a few coefficients in \mathbf{E} are modified. This work, providing an example of introducing the MDL framework in this type of problems, focuses on a robust variant of PCA, RPCA, introduced in [1]. RPCA estimates \mathbf{X} via the following convex optimization problem,

$$\hat{\mathbf{X}} = \arg \min_{\mathbf{W}} \|\mathbf{Y} - \mathbf{W}\|_1 + \lambda \|\mathbf{W}\|_*, \quad (\text{E.2})$$

where $\|\mathbf{W}\|_* := \sum_i \sigma(\mathbf{W})_i$ is the nuclear norm of \mathbf{W} . The rationale behind (E.2) is as follows. First, the ℓ_1 fitting term allows for large errors to occur in the approximation. In this sense, it is a robust alternative to the ℓ_2 norm used in PCA. The second term, $\lambda \|\mathbf{W}\|_*$, is

a convex approximation to the PCA constraint $\text{rank}(\mathbf{W}) \leq k$, merged into the cost function via a Lagrange multiplier λ .

This formulation has been recently shown to be notoriously robust, in the sense that, if a true low-rank matrix \mathbf{X} exists, it can be recovered using (E.2) even when a significant amount of coefficients in \mathbf{E} are arbitrarily large [1]. This can be achieved by setting $\lambda = 1/\sqrt{\max\{m, n\}}$, so that the procedure is parameter-free.

2.1 Low-rank approximation as dimensionality reduction

In many applications, the goal of low-rank approximation is not to find a “true” underlying matrix \mathbf{X} , but to perform what is known as “dimensionality reduction,” that is, to obtain a succinct representation of \mathbf{Y} in a lower dimensional subspace. A typical example is feature selection for classification. In such cases, \mathbf{E} is not necessarily a small measurement perturbation, but a *systematic*, possibly large, error derived from the approximation process itself. Thus, RPCA arises as an appealing alternative for low-rank approximation.

However, in the absence of a true underlying signal \mathbf{X} (and deviation \mathbf{E}), it is not clear how to choose a value of λ that produces a good approximation of the given data \mathbf{Y} for a given application. A typical approach would involve some cross-validation step to select λ to maximize the final results of the application (for example, minimize the error rate in a classification problem).

The issue with cross-validation in this situation is that the best model is selected *indirectly* in terms of the final results, which can depend in unexpected ways on later stages in the data processing chain of the application (for example, on some post-processing of the extracted features). Instead, we propose to select the best low-rank approximation by means of a *direct measure* on the intrinsic ability of the resulting model to capture the desired regularity from the data, this also providing a better understanding of the actual structure of the data. To this end, we use the MDL principle, a general information-theoretic

framework for model selection which provides means to define such a direct measure.

3 MDL-based low-rank model selection

Consider a family \mathcal{M} of candidate models which can be used to describe a matrix \mathbf{Y} *exactly* (that is, losslessly) using some encoding procedure. Denote by $L(\mathbf{Y}|M)$ the description length, in bits, of \mathbf{Y} under the description provided by a given model $M \in \mathcal{M}$. MDL will then select the model $\hat{M} \in \mathcal{M}$ for \mathbf{Y} for which $L(\mathbf{Y}|\hat{M})$ is minimal, that is $\hat{M} = \arg \min_{M \in \mathcal{M}} L(\mathbf{Y}|M)$. It is a standard practice in MDL to use the *ideal* Shannon code for translating probabilities into codelengths. Under this scheme, a sample value y with probability $P(y)$ is assigned a code with length $L(y) = -\log P(y)$ (all logarithms are taken on base 2). This is called an ideal code because it only specifies a codelength, not a specific binary code, and because the codelengths produced can be fractional.

By means of the Shannon code assignment, encoding schemes $L(\cdot)$ can be defined naturally in terms of probability models $P(\cdot)$. Therefore, the art of applying MDL lies in defining appropriate probability assignments $P(\cdot)$, that exploit as much prior information as possible about the data at hand, in order to maximize compressibility. In our case, there are two main components to exploit. One is the low-rank nature of the approximation \mathbf{X} , and the other is that most of the entries in \mathbf{E} will be small, or even zero (in which case \mathbf{E} will be *sparse*). Given a low-rank approximation \mathbf{X} of \mathbf{Y} , we describe \mathbf{Y} as the pair (\mathbf{X}, \mathbf{E}) , with $\mathbf{E} = \mathbf{Y} - \mathbf{X}$. Thus, our family of models is given by $\mathcal{M} = \{(\mathbf{X}, \mathbf{E}) : \mathbf{Y} = \mathbf{X} + \mathbf{E}, \text{rank}(\mathbf{X}) \leq \text{rank}(\mathbf{Y})\}$. As $\mathbf{E} = \mathbf{Y} - \mathbf{X}$, we index \mathcal{M} solely by \mathbf{X} . With these definitions, the description codelength of \mathbf{Y} is given by $L(\mathbf{Y}|\mathbf{X}) := L(\mathbf{X}) + L(\mathbf{E})$. Now, to exploit the low rank of \mathbf{X} , we describe it in terms of its reduced SVD decomposition,

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T \quad \mathbf{U} \in \mathbb{R}^{m \times k}, \quad \Sigma \in \mathbb{R}^{k \times k}, \quad \mathbf{V} \in \mathbb{R}^{k \times n}, \quad (\text{E.3})$$

where k is the rank of \mathbf{X} (the zero-eigenvalues and the respective left and right eigenvectors are discarded in this description). We now have $L(\mathbf{X}) = L(\mathbf{U}) + L(\Sigma) + L(\mathbf{V})$. Clearly, such description will be short if $\text{rank}(\mathbf{X})$ is significantly smaller than $\max\{m, n\}$. We may also be able to exploit further structure in \mathbf{U} , Σ and \mathbf{V} .

3.1 Encoding Σ

The diagonal of Σ is a non-increasing sequence of k positive values. However, no safe assumption can be made about the magnitude of such values. For this scenario we propose to use the *universal prior for integers*, a general scheme for encoding arbitrary positive integers in an efficient way [6],

$$L(j) = \log^* j := \log j + \log \log j + \dots + \log 2.865, \quad (\text{E.4})$$

where the sum stops at the first non-positive summand, and $\log 2.865$ is added to satisfy Kraft's inequality (a requirement for the code to be uniquely decodable, see [7, Chapter 5]). In order to apply (E.4), the diagonal of Σ , $\text{diag}(\Sigma)$ is mapped to an integer sequence via $\lceil 10^{16} \text{diag}(\Sigma) \rceil$, where $\lceil \cdot \rceil$ denotes rounding to nearest integer (this is equivalent to quantizing $\text{diag}(\Sigma)$ with precision $\delta_\Sigma = 10^{-16}$).

3.2 Encoding \mathbf{U} and \mathbf{V} , general case

By virtue of the SVD algorithm, the columns of \mathbf{U} and \mathbf{V} have unit norm. Therefore, the most general assumption we can make about \mathbf{U} and \mathbf{V} is that their columns lie on the respective m -dimensional and n -dimensional unit spheres.

An efficient code for this case can be obtained by encoding each column of \mathbf{U} and \mathbf{V} in the following manner. Let \mathbf{u}_i be a column of \mathbf{U} (\mathbf{V} is similarly encoded). Since \mathbf{u}_i is assumed to be distributed uniformly over the m -dimensional unit sphere, the marginal cumulative

density function of the first element u_{1i} , $F(u_{1i}) = P(x \leq u_{1i})$, corresponds to the proportion of vectors \mathbf{u} that lie on the *unit spherical cap* of height $h = 1 + u_{1i}$ (see Figure E.1(a)). This proportion is given by $F(u_{1i}) = A_m(1 + u_{1i}, 1)/S_m(1)$ where $A_m(h, r)$ and $S_m(r)$ are the area of spherical cap of height h and the total surface area of the m -dimensional sphere of radius r respectively. These are given for the case $0 \leq h \leq r$ ($-1 \leq u_{1i} \leq 0$) by (see [8]),

$$\begin{aligned} A_m(h, r) &= \frac{1}{2} S_m(r) I((2hr - h^2)/r^2; \frac{m-1}{2}, \frac{1}{2}) \\ S_m(r) &= 2\pi^{m/2} r^{m-1} \Gamma^{-1}(m/2), \end{aligned}$$

where $I(x; a, b) = \frac{\int_0^x t^{a-1}(1-t)^{b-1} dt}{B(a, b)}$, and $B(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1} dt$ are the regularized incomplete Beta function and the Beta function of parameters a, b respectively, and $\Gamma(\cdot)$ is the Gamma function. When $r < h \leq 2r$ we simply have $A_m(h, r) = 1 - A_m(2r - h, r)$. For encoding u_{1i} we have $r = 1$ so that

$$F(u_{1i}) = (1/2)I((1 - u_{1i}^2); (m-1)/2, 1/2), \quad -1 \leq u_{1i} \leq 0, \quad (\text{E.5})$$

since $2h - h^2 = h(2 - h) = (1 + u_{1i})[2 - (1 + u_{1i})] = 1 - u_{1i}^2$. Finally, we compute the Shannon codelength for u_{1i} as

$$\begin{aligned} p(u_{1i}) &= F'(u_{1i}) \stackrel{(a)}{=} \frac{(1 - u_{1i}^2)^{(m-3)/2} (u_{1i}^2)^{-1/2}}{2 \cdot B\left(\frac{m-1}{2}, \frac{1}{2}\right)} (-2u_{1i}) \\ &= -\text{sgn}(u_{1i}) (1 - u_{1i}^2)^{(m-3)/2} [B((m-1)/2, 1/2)]^{-1} \\ -\log p(u_{1i}) &= -\frac{m-3}{2} \log(1 - u_{1i}^2) + \log B((m-1)/2, 1/2), \end{aligned}$$

where in (a) we applied the Fundamental Theorem of Calculus to the definition of $F(h)$ and the chain rule for derivatives.

With u_{1i} encoded, the vector $(u_{2i}, u_{3i}, \dots, u_{mi})$ is uniformly distributed on the surface of

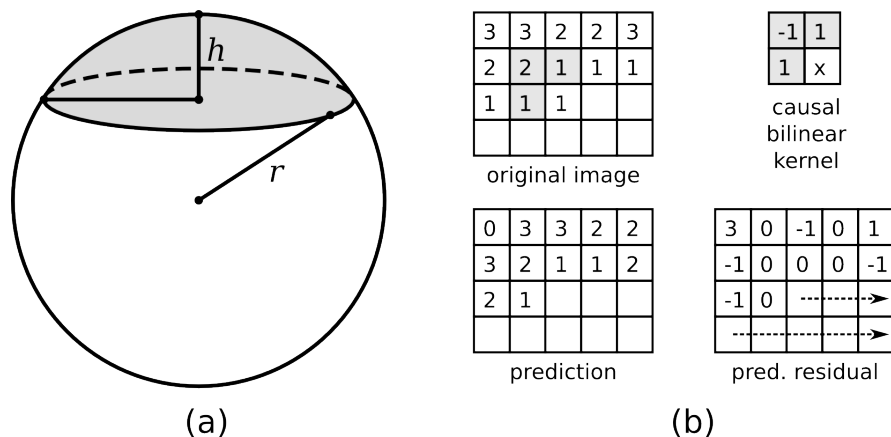


Figure E.1: (a) The spherical cap of radius r and height h (shown in gray). (b) Causal bilinear prediction of smooth 2D images.

the $(m-1)$ -dimensional sphere of radius $r' = 1 - |u_{1i}|$, and we can apply the same formula to compute the probability of u_{2i} , $F(u_{2i}) = A_{m-1}(u_{2i} + r', r')/S_{m-1}(r')$.

Finally, to encode the next column \mathbf{u}_{i+1} , we can exploit its orthogonality with respect to the previous ones and encode it as a vector distributed uniformly over the $m-i$ dimensional sphere corresponding to the intersection of the unit sphere and the subspace perpendicular to $[\mathbf{u}_1, \dots, \mathbf{u}_i]$.

In order to produce finite descriptions $L(\mathbf{U})$ and $L(\mathbf{V})$, both \mathbf{U} and \mathbf{V} also need to be quantized. We choose the quantization steps for \mathbf{U} and \mathbf{V} adaptively, using as a starting point the empirical standard deviation of a normalized vector, that is, $\delta_u = \sqrt{1/m}$ and $\delta_v = \sqrt{1/n}$ respectively, and halving these values until no further decrease in the overall codelength $L(\mathbf{Y}|\mathbf{X})$ is observed.

3.3 Encoding \mathbf{U} predictively

If more prior information about \mathbf{U} and \mathbf{V} is available, it should be used as well. For example, in the case of our example application, the columns of \mathbf{Y} are consecutive frames of a video surveillance camera. In this case, the columns of \mathbf{U} represent “eigen-frames” of the video

sequence, while \mathbf{V} contains information about the evolution in time of those frames (this is clearly observed in figures E.2 and E.3). Therefore, the columns of \mathbf{U} can be assumed to be piecewise smooth, just as normal static images are. To exploit this smoothness, we apply a predictive coding to the columns of \mathbf{U} . Concretely, to encode the i -th column \mathbf{u}_i of \mathbf{U} , we reshape it as an image \mathbf{B} of the same size as the original frames in \mathbf{Y} . We then apply a causal bilinear predictor to produce an estimate of \mathbf{B} , $\hat{\mathbf{B}} = \{\hat{b}_{jl}\}$ where $\hat{b}_{jl} = b_{jl} - b_{j(l-1)} - b_{(j-1)l} + b_{(j-1)(l-1)}$, assuming out-of-range pixels to be 0. The prediction residual $\tilde{\mathbf{B}} = \mathbf{B} - \hat{\mathbf{B}}$ is then encoded in raster scan as sequence of Laplacian random variables with unknown parameter θ_u^i . This encoding procedure, common in predictive coding, is depicted in Figure E.1(b).

Since the parameters $\{\theta_u^i, i = 1, \dots, k\}$ are unknown, we need to encode them as well to produce a complete description of \mathbf{Y} . In MDL, this is done using the so-called *universal encoding schemes*, which can be regarded as a generalization of classical Shannon encoding to the case of distributions with unknown parameters (see [5] for a review on the subject). In this work we adopt the so-called *universal two-part codes*, and apply it to encode each column \mathbf{u}_i separately. Under this scheme, the unknown Laplacian parameter for θ_u^i is estimated via Maximum Likelihood, $\hat{\theta}_u^i(\mathbf{u}_i)$, and quantized with precision $1/\sqrt{m}$, thus requiring $L(\hat{\theta}_u^i) = \frac{1}{2} \log m + c_1$ bits. Given the quantized $\hat{\theta}_u^i$, \mathbf{u}_i is described using the discretized Laplacian distribution $L(\mathbf{u}_i) = -\log P(\mathbf{u}_i | \hat{\theta}_u^i(\mathbf{u}_i)) + c_2$. Here c_1 and c_2 are constants which can be disregarded for optimization purposes. It was shown in [4] that the precision $1/\sqrt{m}$ asymptotically yields the shortest two-parts codelength.

3.4 Encoding \mathbf{V} predictively

We also expect a significant redundancy in the time dimension, so that the columns of \mathbf{V} are also smooth functions of time (in this case, sample index $j = 1, 2, \dots, n$). In this case, we apply a first order causal predictive model to the columns of \mathbf{V} , by encoding them as

sequences of prediction residuals, $\tilde{\mathbf{v}}_i = (\tilde{v}_{i1}, \tilde{v}_{i2}, \dots, \tilde{v}_{in})$, with $\tilde{v}_{ij} = v_{ij} - v_{i(j-1)}$ for $j > 1$ and $\tilde{v}_{i1} = v_{i1}$. Each predicted column \mathbf{v}_i is encoded as a sequence of Laplacian random variables with unknown parameter θ_v^i . As with \mathbf{U} , we use a two-parts code here to describe the data and the unknown Laplacian parameters together. This time, since the length of the columns is n , the codelength associated to each θ_v^i is $L(\theta_v^i) = \frac{1}{2} \log n$.

3.5 Encoding \mathbf{E}

We exploit the (potential) sparsity of \mathbf{E} by first describing the indexes of its non-zero locations using an efficient universal two-parts code for Bernoulli sequences known as Enumerative Code [9], and then the non-zero values at those locations using a Laplacian model. In the specific case of the experiments of Section 4, we encode each row of \mathbf{E} separately. Because each row of \mathbf{E} corresponds to the pixel values at a fixed location across different frames, we expect some of these locations to be better predicted than others (for example, locations which are not occluded by people during the sequences), so that the variance of the error (hence the Laplacian parameter) will vary significantly from row to row. As before, the unknown parameters here are dealt with using a two-parts coding scheme.

3.6 Model selection algorithm

To obtain the family of models \mathcal{M} corresponding to all possible low-rank approximations of \mathbf{Y} , we apply the RPCA decomposition (E.2) for a decreasing sequence of values of λ , $\{\lambda_t : t = 1, 2, \dots\}$ obtaining a corresponding sequence of decompositions $\{(\mathbf{X}_t, \mathbf{E}_t), t = 1, 2, \dots\}$. We obtain such sequence efficiently by solving (E.2) via a simple modification of the Augmented Lagrangian-based (ALM) algorithm proposed in [10] to allow for *warm restarts*, that is, where the initial ALM iterate for computing $(\mathbf{X}_t, \mathbf{E}_t)$ is $(\mathbf{X}_{t-1}, \mathbf{E}_{t-1})$. We then select the pair $(\mathbf{X}_{\hat{t}}, \mathbf{E}_{\hat{t}})$, $\hat{t} = \arg \min_t \{L(\mathbf{X}_t) + L(\mathbf{E}_t)\}$.

4 Results and conclusion

In order to have a reference base, we repeated the experiments performed in [2] using our algorithm. These experiments consist of frames from surveillance cameras which look at a fixed point where people pass by. The idea is that, if frames are stacked as columns of \mathbf{Y} , the background can be well modeled as a low-rank component of \mathbf{Y} (\mathbf{X}), while the people passing by appear as spurious “errors” (\mathbf{E}). Clearly, if the background in all frames is the same, it can be very well modeled as a rank-1 matrix where all the columns are equal. However, lighting changes, shadows, and reflections, “raise” the rank of the background, and the appropriate rank needed to model the background is no longer obvious.

Concretely, the experiments here described correspond to two sequences: “Lobby” and “ShoppingMall,” whose corresponding results are summarized respectively in figures E.2 and E.3.² At the top of both figures, the first two left-eigenvectors \mathbf{u}_1 and \mathbf{u}_2 of \mathbf{X} are shown as 2D images. The middle shows two sample frames of the error approximation. The L -vs- λ curve is shown at the bottom-left (note that the best λ is *not* the one dictated by the theory -0.007 for Lobby and 0.0035 for ShoppingMall– in [1]), and the scaled right-eigenvectors $\sigma_i \mathbf{v}_i$ are shown on the bottom-right. In both cases, the resulting decomposition recovered the low-rank structure correctly, including the background, its changes in illumination, and the effect of shadows. It can be appreciated in the figures E.2-E.3 how such approximations are naturally obtained as combinations of a few significant eigen-vectors, starting with the average background, followed by other details.

4.1 Conclusion

In summary, we have presented an MDL-based framework for low-rank data approximation, which combines state-of-the-art algorithms for robust low-rank decomposition with tools from information theory. This framework is able to capture the underlying low-rank

²The full videos can be viewed at <http://www.tc.umn.edu/~nacho/>.

information on the experiments that we performed, out of the box, and without any hand parameter tuning, thus constituting a promising competitive alternative for automatic data analysis and feature extraction.

References

- [1] E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3), May 2011.
- [2] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. In *Adv. NIPS*, Dec. 2009.
- [3] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2nd edition, Feb. 2009.
- [4] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
- [5] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Trans. IT*, 44(6):2743–2760, 1998.
- [6] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. Singapore: World Scientific, 1992.
- [7] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 2 edition, 2006.
- [8] S. Li. Concise formulas for the area and volume of a hyperspherical cap. *Asian J. Math. Stat.*, 4:66–70, 2011.
- [9] T. M. Cover. Enumerative source encoding. *IEEE Trans. IT*, 19(1):73–77, 1973.

- [10] Z. Lin, M. Chen, and Y. Ma. The Augmented Lagrange Multiplier Method for exact recovery of corrupted low-rank matrices. <http://arxiv.org/abs/1009.5055>.

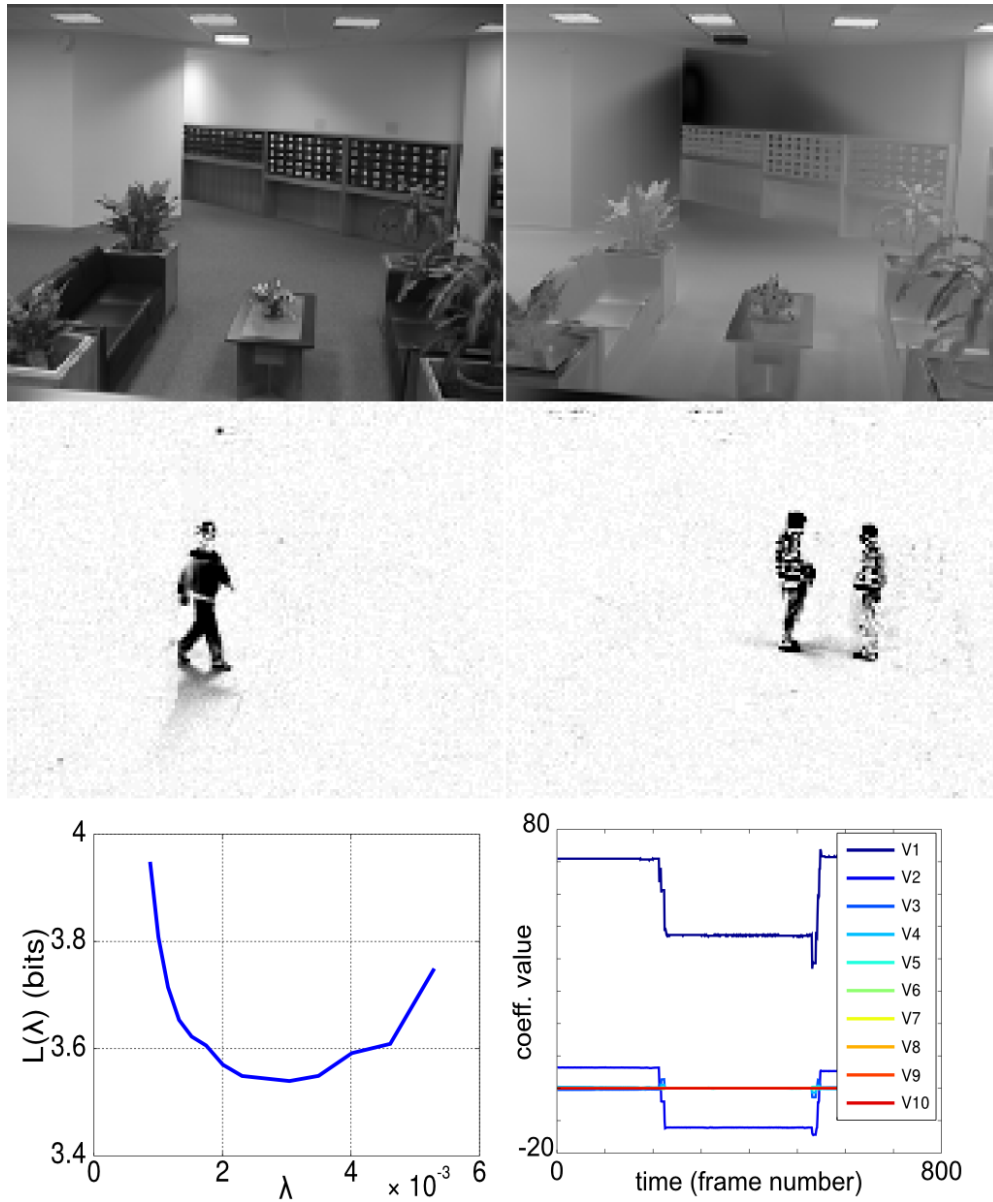


Figure E.2: Results for the “Lobby” sequence (see text for a description of the above pictures and graphs). The rank of the approximation decomposition for this case is $k = 10$. The moment where the lights are turned off is clearly seen here as the “square pulse” in the middle of the first two right-eigenvectors (bottom-right figure). Also note how \mathbf{u}_2 (top-right) compensates for changes in shadows.

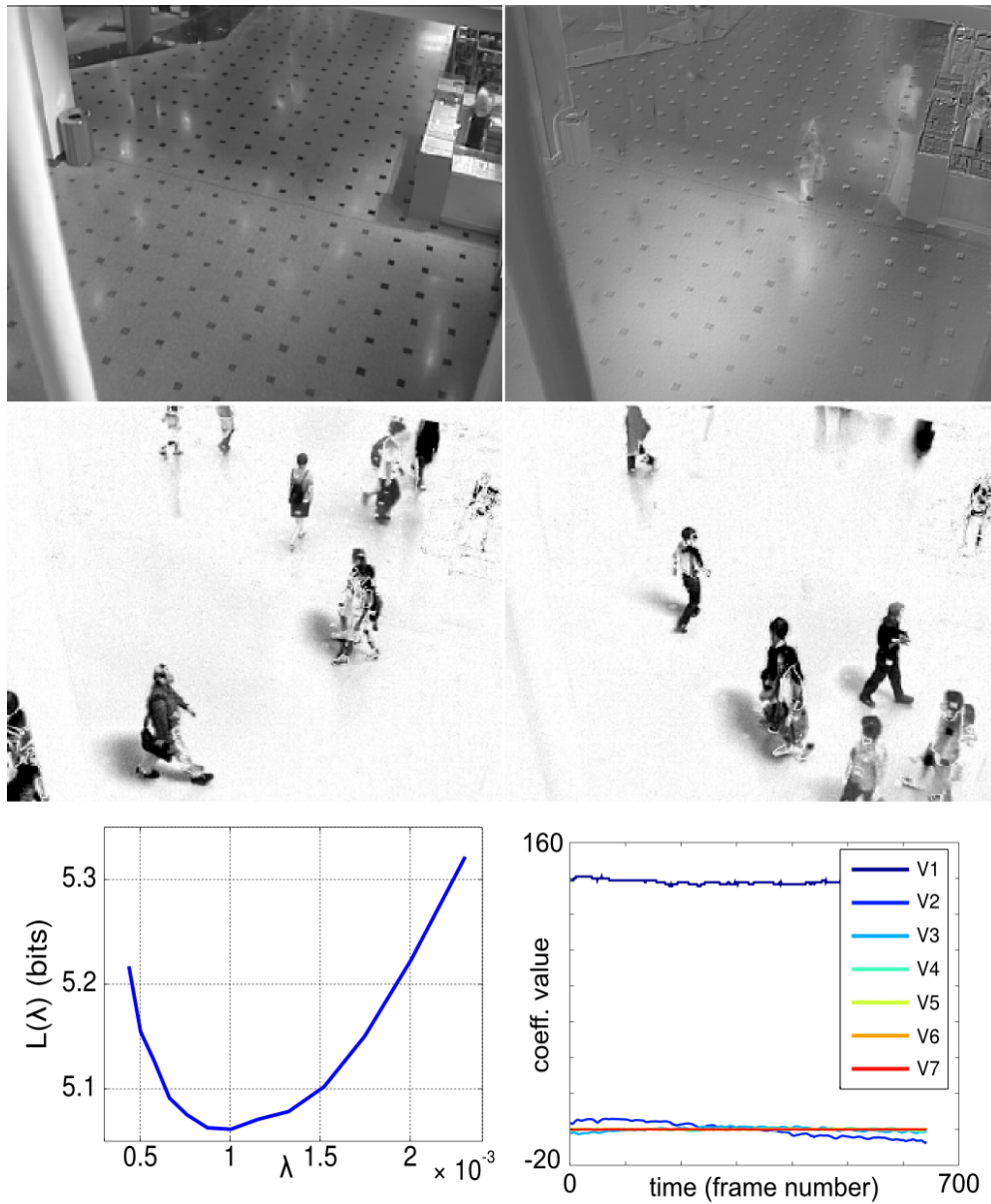


Figure E.3: Results for the “ShoppingMall” sequence (see text for a description of the above pictures and graphs). In this case, the rank of the approximation decomposition is $k = 7$. Here, the first left-eigenvector models the background, whereas the rest tend to capture people that stood still for a while (here we see the “phantom” of two such persons in the second left-eigenvector, top-left picture).