

# THÈSE DE DOCTORAT
# DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

présentée par **JULIEN MAIRAL**

pour obtenir le grade de
**DOCTEUR DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN**

Domaine : **MATHÉMATIQUES APPLIQUÉES**

Sujet de la thèse :

**Représentations parcimonieuses en apprentissage statistique, traitement d'image et vision par ordinateur**

—

**Sparse coding for machine learning, image processing and computer vision**

---

Thèse présentée et soutenue à Cachan le 30 novembre 2010 devant le jury composé de :

| | | |
|---|---|---|
| Francis BACH | Directeur de recherche, INRIA Paris-Rocquencourt | Directeur de thèse |
| Stéphane MALLAT | Professeur, Ecole Polytechnique, New-York University | Rapporteur |
| Eric MOULINES | Professeur, Télécom-ParisTech | Examinateur |
| Bruno OLSHAUSEN | Professeur, University of California, Berkeley | Rapporteur |
| Jean PONCE | Professeur, Ecole Normale Supérieure, Paris | Directeur de thèse |
| Guillermo SAPIRO | Professeur, University of Minnesota | Examinateur |
| Jean-Philippe VERT | Directeur de recherche, Ecoles des Mines-ParisTech | Examinateur |

---

Thèse préparée au sein de l'équipe Willow du laboratore d'informatique de l'École Normale Supérieure, Paris. (INRIA/ENS/CNRS UMR 8548).
23 avenue d'Italie, 75214 Paris.

## RÉSUMÉ

De nos jours, les sciences expérimentales doivent traiter une quantité de données importante et grandissante. Afin de comprendre les phénomènes naturels ainsi que les lois qui les régissent, les scientifiques ont construit des outils améliorant leurs possibilités d'observer le monde, comme des microscopes ou téléscopes. Augmenter la précision de ces outils, ou bien mesurer des quantités "invisibles" par la technologie actuelle sont toujours des préoccupations importantes aujourd'hui. Cette approche empirique soulève toutefois la question de l'analyse et de l'interpétation des données recueillies, de par leur volume et leur complexité. Il s'agit ainsi d'un problème récurrent en neuro-sciences où l'on effectue diverses mesures de l'activité cérébrale, en bio-informatique, où l'on mesure l'expression de gènes, ou bien en radioastronomie avec l'observation du rayonnement fossile de l'univers.

D'autres domaines, en dehors du champ des sciences purement expérimentales, doivent faire face à des problématiques similaires. Ainsi, en robotique, vision artificielle, ou imagerie médicale, les scientifiques souhaitent "comprendre" automatiquement des flux video contenant des millions de pixels ; en sociologie et sciences humaines obtenir des statistiques de population sur de larges bases de données peut être une tâche difficile pour les mêmes raisons. Par ailleurs, le développement d'outils efficaces de traitement de données peut aussi affecter la vie de tous les jours. Nous produisons ainsi pour des raisons de divertissement une grande quantité de signaux, ne serait-ce que par nos appareils photo numériques ou bien nos téléphones portables.

Trouver la meilleure façon de représenter ces signaux numériques est par conséquent une question importante et toujours d'actualité, bien qu'elle ait fait l'objet d'un nombre considérable de publications. Nous étudions dans cette thèse une représentation particulière, intitulée *codage parcimonieux*, fondée sur une méthode d'apprentissage statistique qui s'est révélée empiriquement être très efficace pour certains types de signaux comme les images naturelles. Notre but est de développer de nouveaux outils algorithmiques exploitant cette méthode de codage, ainsi que de nouveaux domaines d'application. Nous adopterons une approche multi-disciplinaire que nous allons détailler par la suite.

Plus concrètement, le codage parcimonieux consiste à représenter des signaux comme combinaisons linéaires de quelques éléments d'un dictionnaire. Ceci peut être vu comme une extension du cadre classique des ondelettes, dont le but est de construire de tels dictionnaires (souvent des bases orthonormales) adaptés aux signaux naturels. De nombreux types d'ondelettes ont ainsi été proposés dans le passé, qui varient essentiellement par leur complexité et leurs propriétés géométriques, mais définir manuellement de tels dictionnaires demeure une tâche difficile. La ligne de recherche que nous poursuivons dans cette thèse diffère du cadre des ondelettes dans le sens où le dictionnaire n'est plus fixe et pré-défini par son utilisateur, mais *appris* à partir de données d'entraînement. Cette approche admet donc des similarités avec l'analyse en composantes principales (ACP), qui "apprend" des "directions principales" orthonormales représentant des données, la principale différence étant l'absence de contrainte d'orthogonalité entre les éléments du dictionnaire. Il en résulte un problème non convexe de factorisation de ma-

trice, qui en pratique nécessite l'utilisation d'outils d'optimisation convexe de fonctions non régulières. Le principal succès des méthodes d'apprentissage de dictionnaire a été la modélisation d'imagettes dans les images naturelles, et la performance des algorithmes de débruitage les utilisant, ce qui a été une motivation importante pour le sujet de nos recherches.

Nous traitons plusieurs questions ouvertes dans cette thèse : Comment apprendre efficacement un dictionnaire ? Comment enrichir le codage parcimonieux en structurant le dictionnaire ? Peut-on améliorer les méthodes de traitement d'image utilisant le codage parcimonieux ? Comment doit-on apprendre le dictionnaire pour une tâche autre que la reconstruction de signaux, quelles en sont les applications en vision par ordinateur ? Nous essayons de répondre à ces questions par une approche multidisciplinaire, en empruntant des outils d'apprentissage statistique, d'optimisation convexe et stochastique, de traitement des signaux et des images, de vision par ordinateur, mais aussi d'optimisation sur des graphes.

L'apprentissage de dictionnaire est souvent considéré comme un processus très coûteux en terme de temps de calcul. La première contribution de cette thèse est un nouvel algorithme d'apprentissage en ligne, fondé sur des méthodes d'approximation stochastique, qui permet d'obtenir un point stationnaire du problème d'optimisation non convexe initial. Notre méthode permet de traiter de grandes bases de données contenant des millions d'exemples d'apprentissage, et s'étend à une large panoplie de problèmes de factorisation de matrices, tels que la factorisation de matrices positives ou l'analyse en composantes principales parcimonieuses. Dans le cadre de ce travail, nous avons aussi développé un logiciel utilisable gratuitement, dont la performance dépasse de façon significative les méthodes concurrentes en termes de vitesse.

Nous nous intéressons ensuite au problème de la structuration du dictionnaire, et à la résolution efficace des problèmes d'optimisation correspondants. A cet effet, nous exploitons des travaux récents qui fournissent un cadre naturel à notre problématique, intitulé codage parcimonieux structuré. Nous étudions en particulier le cas où les dictionnaires sont munis d'une structure hiérarchique, et le cas général où leurs éléments sont structurés en groupes qui se recouvrent. La principale difficulté soulevée par cette nouvelle formulation est le problème d'optimisation correspondant à la décomposition d'un signal étant donné un dictionnaire structuré fixe. La solution que nous proposons combine des outils d'optimisation convexe et d'optimisation sur des graphes et peut en fait être utilisée pour résoudre une grande variété de problèmes d'apprentissage. Plus précisément, nous montrons que l'opérateur proximal associé à la régularisation structurée que nous considérons, est relié à un problème de flot sur un graphe particulier, et peut être calculé efficacement et à grande échelle grâce à un algorithme que nous avons développé. Nous espérons que cette avancée permettra d'ouvrir de nouveaux champs d'application aux méthodes parcimonieuses structurées. Un logiciel implémentant les outils proposés sera disponible gratuitement.

La troisième question traitée dans cette thèse concerne l'amélioration des techniques de traitement d'image utilisant l'apprentissage de dictionnaire. Pour ce faire, nous proposons en sus du codage parcimonieux, d'exploiter explicitement les similarités à l'intérieur

des images, ce qui est le fondement de l'approche de moyennage non-local pour la restauration. A cette fin, nous utilisons le codage parcimonieux simultané, en décomposant de façon jointe des groupes de signaux similaires sur des sous-ensembles d'un dictionnaire appris. Nous montrons que cette approche permet d'obtenir des résultats qui dépassent l'état de l'art pour les tâches de débruitage et dématriçage dans les images, et qu'elle permet de traiter des données brutes d'appareils photos numériques en proposant une qualité meilleure que celle offerte par les logiciels commerciaux.

Nous concluons cette thèse en utilisant l'apprentissage de dictionnaire pour des tâches autres que purement reconstructives. A cet effet, nous présentons une méthode d'apprentissage supervisée, fondée sur un algorithme d'optimisation stochastique, pour des tâches de classification ou de régression, adaptée à des signaux qui admettent des représentations parcimonieuses. Nous illustrons aussi ce concept en modélisant des imagettes de façon discriminative, et montrons que ceci permet de modéliser les contours dans les images. En particulier, nous présentons un détecteur de contour, qui peut aussi être utilisé pour apprendre l'apparence locale des contours d'objets spécifiques.

ABSTRACT

Many fields from experimental sciences now deal with a large and growing amount of data. To understand natural phenomena and eventually their underlying laws, scientists have built physical devices that have enhanced their observation capabilities, such as various types of microscopes or telescopes. Improving upon physical devices, to obtain a better precision or to measure quantities that are invisible with current technologies, is of course still an active scientific topic. On the other hand, scientists have also developed tools to record and process their observations with computers, to analyze and better understand them. This is for instance a common approach to neuroscience with diverse types of measurements of the neural activity, in bioinformatics with gene expressions, or in radio astronomy with measurements of the cosmic microwave background. However, this approach also raises new challenging questions, such as how one should process the resulting large amount of data.

The same need for scalable and efficient data processing tools arises in other fields than pure experimental sciences, such as robotics, computer vision, and biomedical imaging, where one wishes to "understand" continuous video streams containing millions of pixels; but also sociology, where obtaining population statistics from large databases can be difficult. Moreover, developing new data processing tools could also affect the everyday life, where devices such as CCD sensors from digital cameras or cell phones are intensively used for entertainment purposes.

The question of how to represent these digital signals is therefore still acute and of high importance, despite the fact that it has been the topic of a tremendous amount of work in the past. We study in this thesis a particular signal representation called *sparse coding*, based on a machine learning technique, and which has proven to be effective for many modalities such as natural images. Our goal is to provide new algorithmic tools and applications to this coding method, by addressing the problem from various perspectives, which we will detail in the sequel.

Concretely, sparse coding consists of representing signals as linear combinations of a few elements from a dictionary. It can be viewed as an extension of the classical wavelet framework, whose goal is to design such dictionaries (often orthonormal basis) that are adapted to natural signals. Numerous types of wavelets have indeed been proposed in the past, which essentially vary in terms of complexity and geometric properties. Designing by hand such dictionaries remains, however, a difficult task. The line of research we follow in this thesis differs from wavelets in the sense that the dictionary is not fixed and predefined, but *learned* from training data. It shares a similar goal as principal component analysis (PCA), which also "learns" how to represent data by computing orthonormal "principal directions". From an optimization point of view, dictionary learning results in a nonconvex matrix factorization problem, but often deals with nonsmooth convex optimization tools. An important success of dictionary learning has been its ability to model natural image patches and the performance of image denoising algorithms that it

has yielded, which has been an important motivation for our research.

We address in this thesis several open questions: How to efficiently optimize the dictionary? How can sparse coding be enriched by structuring the dictionary? How can one improve sparse coding for image processing tasks? Can we learn the dictionary for a different task than signal reconstruction, and what are the possible applications to computer vision? We try to answer these questions with a multidisciplinarity point of view, using tools from statistical machine learning, convex and stochastic optimization, image and signal processing, computer vision, but also optimization on graphs.

Dictionary learning is often considered as a computationally demanding process. The first contribution of this thesis is a new online learning algorithm, based on stochastic approximations, which is proven to converge to a stationary point of the nonconvex optimization problem. It gracefully scales up to large data sets with millions of training samples, and naturally extends to various matrix factorization formulations, making it suitable for a wide range of learning problems, such as non-negative matrix factorization and sparse principal component analysis. Along with this work, we have developed a freely available software package, which significantly outperforms other approaches in terms of speed.

We then address the questions of how to structure the dictionary, and how to solve the corresponding challenging optimization problems. To that effect, we exploit recent works on structured sparsity, which provide a natural framework to answer our question. We study the case where dictionaries are embedded in a hierarchy and the general case where dictionary elements are structured into overlapping groups. The main difficulty raised by this new formulation is how to decompose a signal given a fixed structured dictionary. The solution we propose combines ideas from convex optimization and network flow optimization. It in fact extends beyond the dictionary learning framework and can be used for solving a new class of regularized machine learning problems. More precisely, we show that the proximal operator associated with the structured regularization we consider is related to a quadratic min-cost flow problem, and can be solved efficiently at large scale with an algorithm we propose. We therefore make a bridge between the literature of sparse methods, and network flow optimization. We hope that this contribution will open up a new range of applications for structured sparse models. A software package implemented these methods has been developed and will be made freely available.

The third question we address also consists of enriching the basic dictionary learning framework, but in a specific way for image processing applications. Explicitly exploiting the self-similarities of natural images has led to the successful "non-local means" approach to image restoration. We propose simultaneous sparse coding as a framework for combining this approach with dictionary learning in a natural manner. This is achieved by jointly decomposing groups of similar signals on subsets of the learned dictionary. We show that this approach achieves state-of-the-art results for image denoising and demosaicking, and competes with commercial software for restoring raw data from digital cameras.

We concludes this thesis by considering dictionary learning as a way to learn features

for a different task. We show that it can be used in a supervised way for different classification or regression tasks, for data that admit sparse representation, and show how to use a stochastic gradient descent algorithm for addressing the new learning problem. We also show that this idea can be used in computer vision for modelling the local appearance of natural image patches in a discriminative way, and that it is especially well adapted for modelling edges in natural images. In particular, we address with this approach the problem of edge detection, category-based edge detection and show that it leads to state-of-the-art results for other tasks such as digit recognition of inverse half-toning.

# ACKNOWLEDGMENTS

Foremost, I would like to thank my advisors Francis Bach and Jean Ponce, who have shared their expertise with me during three years and have been of invaluable help. In particular, I would like to say how lucky I was to meet Jean Ponce at the end of 2006, even before that the Willow project-team existed. He has indeed managed to build, in an amazingly short period of time, one of the leading team in computer vision and machine learning in the world. This would not have been possible without the support of INRIA, which I would like to thank for having funded my thesis and having provided to the Willow group a great research environment.

I would like to thank Stéphane Mallat and Bruno Olshausen for accepting to review my thesis, and sharing interesting comments and discussions with me. I am also grateful to Eric Moulines, Guillermo Sapiro and Jean-Philippe Vert for accepting to be part of the jury.

I would also like to express my gratitude to many people who have directly or indirectly contributed to the work in this thesis: Michael Elad and again Guillermo Sapiro with whom I had the chance to work before starting my PhD; Bin Yu for accepting me as a post-doctoral researcher at UC Berkeley next year; all the members (or former members) of the Willow project-team who have been both great friends and collegues, in alphabetical order, Jean-Yves Audibert, Sylvain Arlot, Louise Benoit, Y-Lan Boureau, Neva Cherniavsky, Timothée Cour, Florent Couzinié-Devy, Vincent Delaitre, Olivier Duchenne, Cécile Espiègle, Loïc Février, Yasutaka Furukawa, Jan Van Gemert, Edouard Grave, Warith Harchaoui, Toby Hocking, Ugo Jardonnet, Rodolphe Jenatton, Armand Joulin, Hui Kong, Akash Kushal, Ivan Laptev, Augustin Lefèvre, Jose Lezama, Guillaume Obozinski, Bryan Russell, Josef Sivic, Mathieu Solnon, Muneeb Ullah, Oliver Whyte, Andrew Zisserman, our "almost members" of Willow, Fredo Durand and Alyosha Efros; with a special thank to Rodolphe Jenatton, who has been my closest collaborator during the last year of my PhD. It was also a pleasure to interact with regular visitors to Willow, among them Martial Hébert, Svetlana Lazebnik, Yann Lecun, Léon Bottou, and Cordélia Schmid.

There are also other students or researchers with whom I had the chance to interact and that I would like to thank here: Alexandre Gramfort, Zaid Harchaoui, Laurent Jacob, Neus Sabater and Mikhail Zaslavskiy; and also all the people I met in Guillermo Sapiro's group: Iman Aganj, Pablo Arias, Xue Bai,Leah Bahr, Gloria Haro, Frederico Lecumberry, Stacey Levine, Hstau Liao, Mona Mahmoudi, Sharareh Noorbaloochi, Ignacio Ramirez, Diego Rother, Pablo Sprechmann.

Finally, I would like to thank Marine for all her support during these three years.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

# 1

# Introduction and Related Work

Finding "good" signal representations has been the topic of a large amount of research since early works in signal and image processing. Estimation problems arising in these fields, such as denoising, reconstruction from incomplete data, or more generally restoration, are indeed often difficult to solve without an arbitrary a priori model of the data source.

Various smoothness assumptions were first used, leading for instance to Laplacian filtering (Kovasznay and Joseph, 1955), anisotropic filtering (Perona and Malik, 1990) or total variation (Rudin and Osher, 1994) in image processing, to cite only a few of them. More recent works have focused on representing data vectors as linear combinations of few elements from a pre-defined *dictionary*, which is often an orthonormal basis set, introducing the concept of *sparsity*. Finding low-dimensional representations of a given signal in a well chosen basis set is intuitively useful for restoration:[1] Suppose that we have at our disposal a dictionary which is good at reconstructing a class of signals (i.e., the signals admit sparse representations over the dictionary), and bad at reconstructing noise. Then, one hopes that a sparse approximation of a noisy signal with the dictionary significantly reduces the amount of noise without losing signal information.[2] Experiments have shown that such a model with sparse decompositions (*sparse coding*) is very effective in many applications (Chen et al., 1998).

However, the question of designing good dictionaries adapted to different modalities (e.g., natural images) remains open, and has in fact been an active topic of research. The discrete cosine transform (Ahmed et al., 1974), wavelets (see Mallat, 1999, and references therein), curvelets (Candes and Donoho, 2002, 2004), contourlets (Do and Vetterli, 2003a,b), wedgelets (Donoho, 1998), bandlets (Mallat and Pennec, 2005a,b; Mallat and Peyré, 2008), and steerable wavelets (Freeman and Adelson, 1991; Simoncelli et al., 1992) are all attempts to fulfill the above sparse coding model for natural signals.

[1] The terminology "basis" is slightly abusive here since the elements of the dictionary are not necessarily linearly independent and the set can be overcomplete—that is, have more elements than the signal dimension.

[2] Formally, let $\mathbf{x}$ be a clean signal in $\mathbb{R}^n$ which lives in a linear subspace $\Gamma$ of dimension $L \ll n$, and let us consider a noisy version $\mathbf{y} = \mathbf{x} + \mathbf{w}$, where $\mathbf{w}$ is a white and Gaussian noise vector of standard deviation $\sigma$. A projection of the noisy vector $\mathbf{y}$ onto the linear subspace $\Gamma$ is equal to $\mathbf{x} + \mathbf{w}'$, with $\mathbb{E}[\|\mathbf{w}'\|_2^2] = L\sigma^2 \ll \mathbb{E}[\|\mathbf{w}\|_2^2] = n\sigma^2$, and the amount of noise is reduced. The main difficulty is in fact to find the right subspace $\Gamma$, which is usually unknown.

Indeed, they have led to effective algorithms for many image processing applications, such as compression (Mallat, 1999; Chang et al., 2000), denoising (Starck et al., 2002; Portilla et al., 2003; Matalon et al., 2005; Eslami and Radha, 2006), inpainting (Elad et al., 2005), and more. Note that the terminology of "models" we have used so far is a bit loose. The ones we have mentioned and will use in this thesis are not "true" models in the generative sense.[3] They in fact define classes of regularized signals which hopefully contain the ones one wants to represent, but also contain (in fact mostly) irrelevant ones.

Originally introduced by Olshausen and Field (1996, 1997) to model the receptive fields of simple cells in the mammalian primary visual cortex, the idea of *learning* the dictionary instead of using a predefined one has recently led to state-of-the-art results in numerous low-level signal processing tasks such as image denoising (Elad and Aharon, 2006; Mairal et al., 2008b,d, 2009c), texture synthesis (Peyré, 2009), and audio processing (Zibulevsky and Pearlmutter, 2001; Grosse et al., 2007; Févotte et al., 2009), as well as higher-level tasks such as image classification (Raina et al., 2007; Mairal et al., 2008a, 2009b; Bradley and Bagnell, 2009; Yang et al., 2009; Boureau et al., 2010), showing that sparse learned models are well adapted to a large class of natural signals. Unlike decompositions based on principal component analysis and its variants, these models do not impose that the basis vectors be orthogonal, allowing more flexibility to adapt the representation to the data, and they have been shown to significantly improve signal reconstruction (Elad and Aharon, 2006). Although some of the learned dictionary elements may sometimes "look like" wavelets (or Gabor filters), they are tuned to the input images or signals, leading to much better results in practice.

It is interesting to see that some of the concepts presented here have also emerged in statistics and machine learning from a slightly different viewpoint. In this literature, What we have called "dictionary" in the previous paragraphs is usually fixed, and is defined as a set of "predictors" or "variables". Statistical estimators and solutions of machine learning problems are often defined as linear combinations of such "predictors" and in fact, due to their simplicity, these linear models are the most widely used ones for prediction tasks (Hastie et al., 2009). In supervised learning, an empirical risk (usually a convex loss) is minimized, so that the linear model fits some training data, and one hopes that the learned model generalizes well on new data points. However, due to possibly small numbers of training samples and/or a large number of predictors, overfitting can occur, meaning that the learned parameters do fit well the training data, but have a bad generalization performance. This issue can be solved by making a priori assumptions on the solution, naturally leading to the concept of *regularization*. When smooth solutions are preferred, one can for instance use the Tikhonov regularization (Tikhonov and Arsenin, 1977), also used in ridge regression (Hoerl and Kennard, 1970). When one knows in advance that the solution is sparse—that is, only a few predictors are relevant, a sparsity-inducing regularization such as the $\ell_1$-norm is well adapted, leading for instance to the Lasso (Tibshirani, 1996), or equivalently to the basis pursuit formulation

[3] In a generative setting, one usually models the underlying probability distribution of input data, from which it is possible to draw new samples. To the best of our knowledge, no such good model exists for natural images.

from the signal processing literature (Chen et al., 1998). Note that the $\ell_1$-norm was also used by Markowitz (1952) for the problem of portfolio selection, and has in fact been revisited several times.

More generally, it is possible to encode additional knowledge in the regularization than just sparsity. A recent topic of research indeed consists of building structured sparsity-inducing norms, which encourage the solutions of sparse regularized problems to have specific patterns of non-zero coefficients. One may want such patterns to be structured in non-overlapping groups (Turlach et al., 2005; Yuan and Lin, 2006; Obozinski et al., 2009), in a tree (Zhao et al., 2009; Bach, 2009), or in overlapping groups (Jenatton et al., 2009; Jacob et al., 2009; Huang et al., 2009; Baraniuk et al., 2010).

The work presented in this thesis follows these lines of research. It provides efficient algorithmic tools for dictionary learning and structured sparse decomposition problems. It also extends the dictionary learning formulation to a supervised setting, and presents applications in image processing and computer vision that achieves state-of-the-art results for different tasks. We present in more details these contributions in Section 1.1, before introducing in Section 1.2 the notation used throughout the thesis. We also present in Section 1.3 sparsity-inducing norms, and in Section 1.4 the optimization tools for sparse methods which we have used. We briefly review the literature of dictionary learning in Section 1.5, as well as its successful applications in image processing in Section 1.6.

## 1.1 Contributions of the Thesis

This thesis brings several contributions to the fields of sparse methods in machine learning, signal and image processing, and computer vision. We now review them, following the organization of the manuscript:

- Chapter 2 presents a fast dictionary learning algorithm based on stochastic approximations, which, to the best of our knowledge, significantly outperforms all approaches in terms of speed. This procedure allows learning dictionaries with millions of training samples, and can be extended to various matrix factorization problems, such as non-negative matrix factorization and sparse principal component analysis. An efficient C++ implementation of this algorithm is available in the software SPAMS, and is presented in more details in Appendix D.[4]

- Chapter 3 introduces new algorithmic tools for solving structured sparse decomposition problems. We show that the proximal operator associated with the norms we consider is related to finding a flow with minimum cost on a particular graph, which makes a bridge between the literature of sparse methods and network flow optimization. We propose an efficient an scalable procedure for solving it, which opens up a new range of applications for structured sparse models. We illustrate our approach for learning hierarchically structured dictionaries of natural image

[4]The software can be freely downloaded at http://www.di.ens.fr/willow/SPAMS/.

patches that show improved performance over classical unstructured ones in noisy settings, and background subtraction in videos.

- We show in Chapter 4 how to exploit both image self-similarities and sparse representations for image restoration using simultaneous sparse coding. The proposed approach achieves state-of-the-art results for image denoising and image demosaicking, as well as competitive results for denoising raw data from CCD sensors of digital cameras.
- In Chapter 5, we introduce discriminative sparse representations, which are well suited for modelling the appearance of image patches, especially edges in images. We use these representations for classifying patches from different textures, from different objects, and learning a class-specific edge detector.
- In Chapter 6, we present a more general formulation than in Chapter 5 for learning dictionaries adapted to classification or regression tasks and an efficient optimization procedure for solving it. This approach leads to (or close to) state-of-the-art results for several problems such as digit recognition and non-linear inverse image mapping tasks such as inverse halftoning.

## 1.2 Notation

We denote vectors by bold lower case letters, and matrices by bold upper case ones. For a vector $\mathbf{x}$ in $\mathbb{R}^m$ and and integer $j$ in $[\![1;m]\!] \triangleq \{1,\dots,m\}$, the $j$-th entry of $\mathbf{x}$ is denoted by $\mathbf{x}_j$. For a matrix $\mathbf{X}$ in $\mathbb{R}^{m\times n}$, and a pair of integers $(i,j) \in [\![1;m]\!] \times [\![1;n]\!]$, the entry at row $i$ and column $j$ of $\mathbf{X}$ is denoted by $\mathbf{X}_{ij}$. When $\Lambda \subseteq [\![1;m]\!]$ is a finite set of indices, the vector $\mathbf{x}_\Lambda$ of size $|\Lambda|$ contains the entries of $\mathbf{x}$ corresponding to the indices in $\Lambda$. Similarly, when $\mathbf{X}$ is a matrix of size $m \times n$ and $\Lambda \subseteq [\![1;n]\!]$, $\mathbf{X}_\Lambda$ is the matrix of size $m \times |\Lambda|$ containing the columns of $\mathbf{X}$ corresponding to the indices in $\Lambda$.

We define for $q \geq 1$ the $\ell_q$-norm of a vector $\mathbf{x}$ in $\mathbb{R}^m$ as:

$$\|\mathbf{x}\|_q \triangleq \Big(\sum_{i=1}^{m} |\mathbf{x}_j|^q\Big)^{1/q}, \quad \text{and} \quad \|\mathbf{x}\|_\infty \triangleq \max_{j=1,\dots,m} |\mathbf{x}_j| = \lim_{q\to\infty} \|\mathbf{x}\|_q.$$

We also define the $\ell_0$ pseudo-norm as the sparsity measure which counts the number of nonzero elements in a vector:[5]

$$\|\mathbf{x}\|_0 \triangleq \#\{j \ \text{ s.t. } \ \mathbf{x}_j \neq 0\} = \lim_{q\to 0^+} \Big(\sum_{j=1}^{m} |\mathbf{x}_j|^q\Big).$$

We denote the Frobenius norm of a matrix $\mathbf{X}$ in $\mathbb{R}^{m\times n}$ by

$$\|\mathbf{X}\|_{\mathrm{F}} \triangleq \Big(\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{X}_{ij}^2\Big)^{1/2}.$$

[5] Note that it would be more proper to write $\|\mathbf{x}\|_0^0$ instead of $\|\mathbf{x}\|_0$ to be consistent with the traditional notation $\|\mathbf{x}\|_q$. However, for the sake of tradition, we will keep this notation unchanged.

We usually denote a sequence of scalars and real-valued functions with lower indices, for instance $u_t$, for $t \geq 0$, and sequences of vectors and matrices using upper indices, for instance $\mathbf{x}^t$ or $\mathbf{X}^t$, $t \geq 0$. For a sequence of vectors (or matrices) $\mathbf{x}^t$ and scalars $u_t$, we write $\mathbf{x}^t = O(u_t)$ when there exists a constant $K > 0$ so that for all $t$, $\|\mathbf{x}^t\|_2 \leq K u_t$. Note that for finite-dimensional vector spaces, the choice of norm is essentially irrelevant (all norms are equivalent).

We denote by $B_\varepsilon(\mathbf{x})$ the open ball of radius $\varepsilon$ centered in $\mathbf{x}$. Given two matrices $\mathbf{X}$ in $\mathbb{R}^{m_1 \times n_1}$ and $\mathbf{Y}$ in $\mathbb{R}^{m_2 \times n_2}$, $\mathbf{X} \otimes \mathbf{Y}$ denotes the Kronecker product between $\mathbf{X}$ and $\mathbf{Y}$, defined as the matrix in $\mathbb{R}^{m_1 m_2 \times n_1 n_2}$, with blocks of sizes $m_2 \times n_2$ equal to $\mathbf{X}_{ij}\mathbf{Y}$. For more details and properties of the Kronecker product, see Golub and Van Loan (1996), and Magnus and Neudecker (1999). When necessary, other specific notations will also be introduced in the remaining chapters.

## 1.3 Sparse Methods and Sparsity-Inducing Norms

Sparse regularized problems in machine learning and signal processing often consist of fitting some model parameters $\boldsymbol{\alpha}$ in $\mathbb{R}^p$ to training data, while making the a priori assumption that $\boldsymbol{\alpha}$ should be sparse. This is usually achieved by minimizing some smooth convex function $f : \mathbb{R}^p \to \mathbb{R}$,[6] which is typically an empirical risk in machine learning or a data fitting term in signal processing, and a sparsity-inducing regularization $\Omega$:

$$\min_{\boldsymbol{\alpha} \in \mathcal{A}} \big[ g(\boldsymbol{\alpha}) \triangleq f(\boldsymbol{\alpha}) + \lambda \Omega(\boldsymbol{\alpha}) \big], \tag{1.1}$$

where $\mathcal{A} \subseteq \mathbb{R}^p$ is a convex set, $\boldsymbol{\alpha}$ is a vector in $\mathcal{A}$, and $\lambda$ is a non-negative parameter controlling the trade-off between data fitting and regularization. To encourage sparsity in $\boldsymbol{\alpha}$, a natural choice would be to take $\Omega$ to be the $\ell_0$ pseudo-norm that counts the number of non-zero coefficients in $\boldsymbol{\alpha}$. However, solving Eq. (1.1) in this setting is often intractable, such that one has either to look for an approximate solution using a greedy algorithm, or one should resort to a convex relaxation instead. A typical example of such a convex formulation is for instance the $\ell_1$-decomposition problem, also known as the Lasso (Tibshirani, 1996) or basis pursuit (Chen et al., 1998):

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \Big[ \frac{1}{2} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \Big], \tag{1.2}$$

where $\mathbf{x}$ in $\mathbb{R}^m$ is a signal and $\mathbf{D} = [\mathbf{d}^1, \ldots, \mathbf{d}^p]$ in $\mathbb{R}^{m \times p}$ is a dictionary whose columns are the dictionary elements. As shown below, when the value of $\lambda$ is large enough, $\boldsymbol{\alpha}$ is known to be sparse, and only a few dictionary elements are involved. The problem of efficiently solving Eq. (1.2) has received a lot of attention lately. Indeed, the corresponding literature is abundant, vast, but also redundant and confusing. We will present later in this manuscript optimization methods which have experimentally proven to be efficient for the applications we are interested in.

Before that, let us develop a bit more the discussion on sparse regularization problems, by answering our first important question

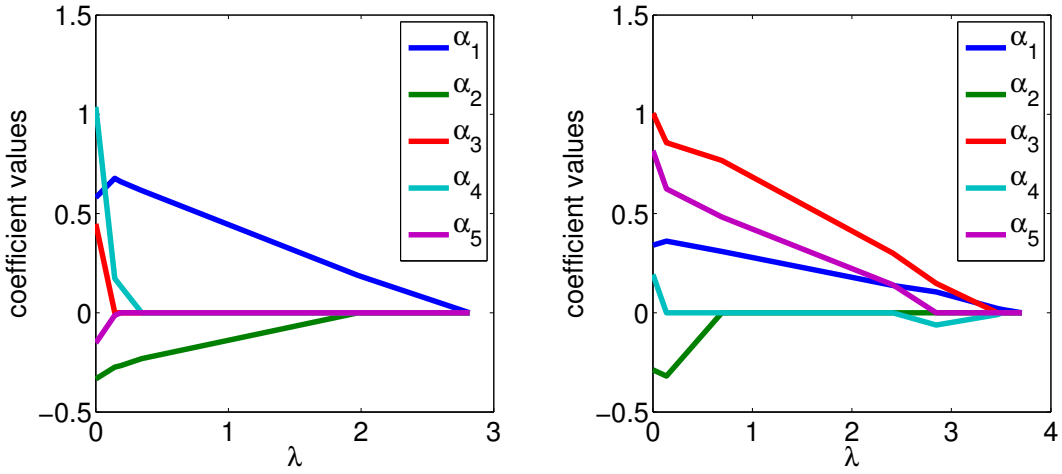
[6] We often assume $f$ to be differentiable with a Lischitz continuous gradient.

### 1.3.1 Does The $\ell_1$-norm Induce Sparsity?

Let us consider the general formulation of Eq. (1.1) and let us take $\Omega$ to be the $\ell_1$-norm. Our first remark will be a bit contradictory with the terminology of "sparsity-inducing norm" often used to characterize the $\ell_1$-norm, but we will clarify this in the sequel. Indeed, depending on the choice of $f$, *the $\ell_1$ regularization in Eq. (1.1) does not always lead to sparse solutions*. Let us consider for instance the problem of minimizing on $\mathbb{R}^+$ the function $g(\alpha) - 2\sqrt{\alpha} + \lambda|\alpha|$, where $\alpha$ is a scalar. The solution is $\alpha^\star(\lambda) = \frac{1}{\lambda^2}$, which is never zero for any value of $\lambda$. To enjoy sparsity-inducing properties of the $\ell_1$-norm, we need to be a bit more careful in the choice of the function $f$ we want to regularize. For instance, when $f$ is differentiable at zero, then the solution is exactly zero when $\lambda$ is large enough (we will characterize in Section 1.4 why this is true). The Lasso formulation presented in Eq. (1.2) enjoys this property, and so will be all the functions that we will consider in this thesis. We will now assume that $f$ satisfies this condition.

Let us now suppose that the solution of Eq. (1.1), which we denote by $\boldsymbol{\alpha}^\star(\lambda)$, is unique. We know that $\boldsymbol{\alpha}^\star(\lambda)$ is equal to 0 when $\lambda$ is large enough, but one can also wonder whether $\lambda$ can help us in controlling the sparsity of $\boldsymbol{\alpha}^\star(\lambda)$. In other words, if given $\lambda$ the solution $\boldsymbol{\alpha}^\star(\lambda)$ has a sparsity $s \triangleq \|\boldsymbol{\alpha}^\star(\lambda)\|_0$, can we increase (or decrease) the value of $s$ by reducing (respectively increasing) the value of $\lambda$? We have observed in this thetis that it is empirically *often true*, even though there are no clear analytical arguments relating the $\ell_1$-norm of a solution to the corresponding sparsity that it yields. On the other hand, it is very easy to generate counter-examples, where this "expected" behavior is not exactly satisfied for every value of $\lambda$, especially when using randomly generated data, for which no good dictionary exists. Let us illustrate this with an example: We consider a random dictionary $\mathbf{D}$ in $\mathbb{R}^{5\times 5}$ whose entries are i.i.d. samples from a normal distribution $\mathcal{N}(0,1)$. We generate a vector $\mathbf{x}$ in $\mathbb{R}^5$ the same way. We present in Figure 1.1 the regularization path of the corresponding Lasso formulation—that is, all the solutions $\boldsymbol{\alpha}^\star(\lambda)$ for every value of $\lambda$ for two different couples $(\mathbf{D}, \mathbf{x})$ obtained in this manner. We observe on the first one a "typical" behavior: When $\lambda$ is large enough, the solution is 0. When $\lambda$ progressively decreases, variables enter the set of active variables, one at a time, until the solution is not sparse anymore. In this case, the sparsity of the solution is a decreasing function of $\lambda$. The second case is a counter-example, where variable number 4 gets active in the path before getting inactive again, making the sparsity of the solution non decreasing with $\lambda$. Note that these regularization paths that we have plotted are piecewise linear. This is in fact a property of the Lasso, which we will formally detail later.

We have presented results on the choice of the $\ell_1$-norm which are both positive and negative, showing that it can induce sparsity under some conditions, and claiming that controlling the $\ell_1$-norm of a solution makes it possible to control its sparsity in many practical situations. Let us now give some intuition about the reasons why.




(a) "Typical Scenario" (b) Counter-example

Figure 1.1: Values of the solutions $\boldsymbol{\alpha}^\star(\lambda)$. Each curve corresponds to one entry in $\boldsymbol{\alpha}^\star$ as a solution of the regularization parameter $\lambda$.

### 1.3.2 Why Does the $\ell_1$-Norm Induce Sparsity?

As already mentioned, there is no general analytical link relating the $\ell_1$-norm of a solution with the sparsity in general. However, there are several intuitive reasons why an $\ell_1$-norm regularizer encourages sparse solutions in general.

#### Analytical Analysis in 1-D — Soft-Thresholding

Let us consider the one-dimensional case, with the following optimization problem

$$\min_{\alpha\in\mathbb{R}} \big[g(\alpha) \triangleq \frac{1}{2}(x-\alpha)^2 + \lambda|\alpha|\big],$$

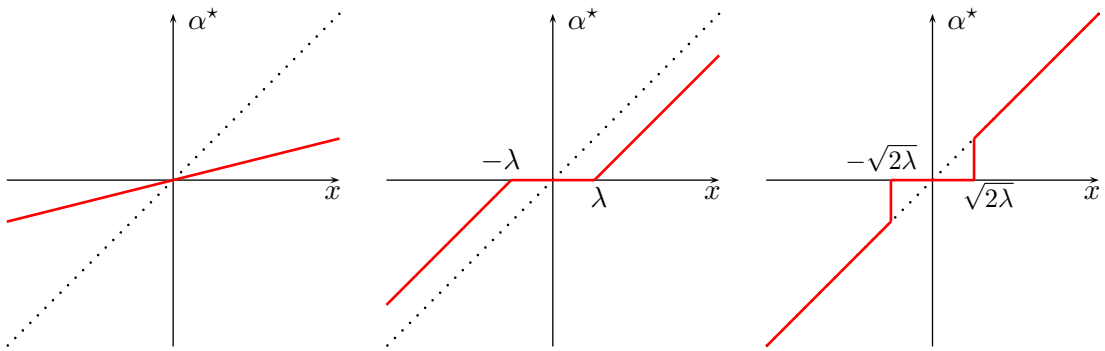
where $x$ is a scalar. The function $g$ is piecewise quadratic with a kink (non-differentiable point) at 0. Optimality conditions of this problem are the following

- If $|\alpha| > 0$, $g$ is differentiable at $\alpha$ and $g'(\alpha) = x - \alpha + \lambda\,\mathrm{sign}(\alpha) = 0$.
- If $\alpha = 0$, the right and left derivatives of $g$ at 0 are both positive, leading to the conditions $-\alpha + \lambda \geq 0$ and $-\alpha - \lambda \geq 0$.

It is easy to see from these conditions that the solution $\alpha^\star$ is necessary obtained with the soft-thresholding operator introduced by Donoho and Johnstone (1995):

$$\alpha^\star(x,\lambda) = \mathrm{sign}(x)(|x| - \lambda)^+,$$

where $(.)^+ \triangleq \max(., 0)$. The $\ell_1$-norm in this problem has first a thresholding effect (the solution is 0 when $|x|$ is smaller than $\lambda$), but also a shrinkage effect (when $|x| > \lambda$, $|\alpha^\star(x, \lambda)| = |x| - \lambda$). In comparison, when using the $\ell_0$-pseudo-norm instead of the $\ell_1$-norm, the solution $\alpha^\star(x, \lambda)$ also admits a closed form which is the hard-thresholding operator $\alpha^\star(x, \lambda) = \mathbf{1}_{|x| \geq \sqrt{2\lambda}} x$, and when using the squared $\ell_2$-norm, the solution is obtained by a scaling $\alpha^\star(x, \lambda) = \frac{x}{1+2\lambda}$. These different effects are shown in Figure 1.2.




(a) scaling operator (b) soft-thresholding operator (c) hard-thresholding operator

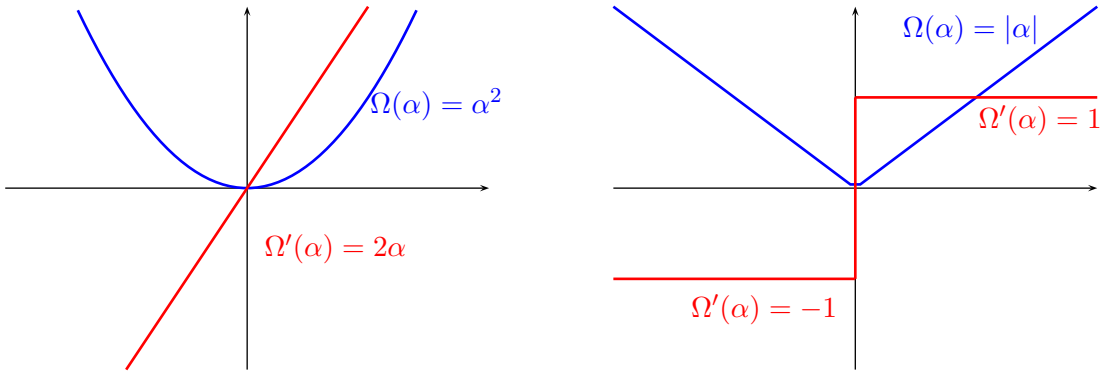
Figure 1.2: From left to right: scaling, soft-thresholding, hard-thresholding operators. The value of $\alpha^\star(x, \lambda)$ is reported as a function of the input $x$ for a fixed $\lambda$. The black dotted curve is the function $\alpha^\star(x, 0)$ (no regularization), whereas the red plain curve corresponds to the value of $\alpha^\star(x, \lambda)$.

We have seen that in 1-D, the $\ell_1$-norm amounts to using a sparsity-inducing operator on the input data—that is for a fixed $x$, $\alpha^\star(x, \lambda) = 0$ for $\lambda$ large enough. We now give a physical illustration of this effect.

### "Physical Explanation" in 1-D

Let us first compare the squared $\ell_2$-regularization with the $\ell_1$-norm. In Figure 1.3, we have plotted the corresponding $\Omega$ functions and their derivatives (where they are defined). A physical interpretation of these functions is to see them as "potential energies", which are minimum when $\alpha$ is equal to zero, and see their derivatives as the intensity of the "force" that tends to make $\alpha$ smaller. In the case of the squared $\ell_2$-norm, the intensity of this force vanishes when $\alpha$ gets closer to 0, preventing the regularization to induce sparsity if the minimum of $f$ is different than 0. In the case of the $\ell_1$-norm, the intensity is constant when $\alpha$ gets closer to 0, and is proportional to the parameter $\lambda$, making it possible to drive $\alpha$ down to 0.

This can further be illustrated with a more concrete example, which we show in Figure 1.4. We start by considering in Figure 1.4a two springs with zero mass and negligible length that are fixed to a wall (the fixation points are represented by red circles). These correspond to initial conditions. The height of the blue points is denoted by $x$. On Figure 1.4b, we attach to the left spring another one, whose other extremity



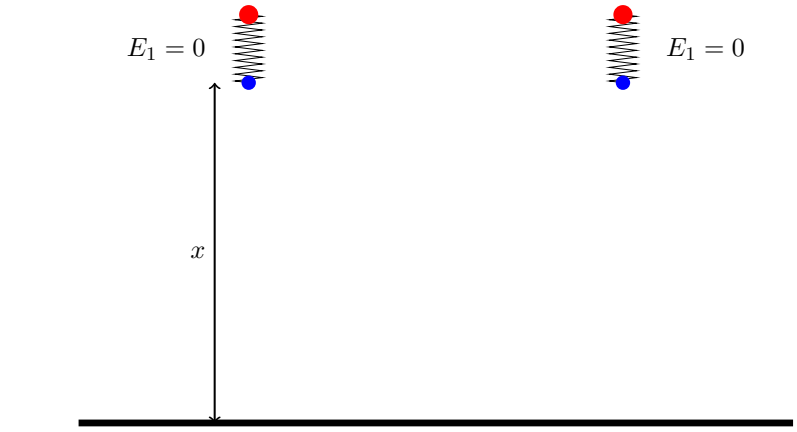

(a) $\ell_2$ squared regularization (b) $\ell_1$ norm

Figure 1.3: Comparison of the Tikhonov ($\ell_2$ squared) and $\ell_1$ regularization in one dimension. Blue curves represents the regularizers as functions of $\alpha$, and red curves the derivatives.

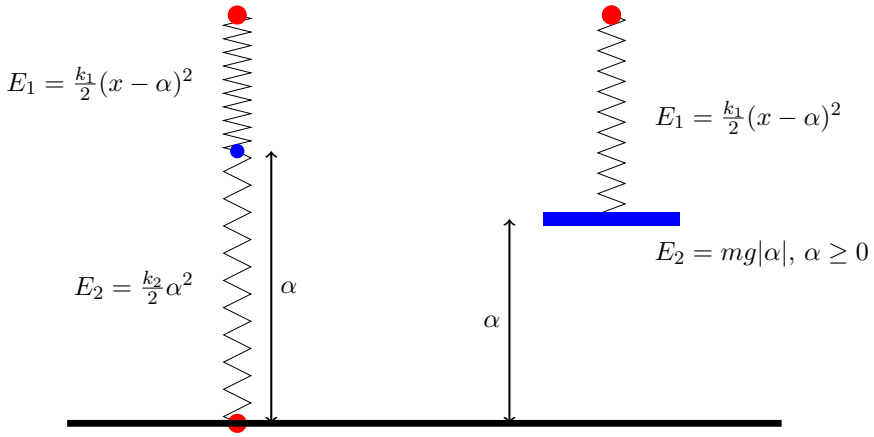
is fixed to the ground. Due to the action of the new spring, the height of the blue point decreases to a new value $\alpha$. The two springs have respective energies $E_1 = \frac{k_1}{2}(x-\alpha)^2$ and $E_2 = \frac{k_2}{2}\alpha^2$, where $k_1$ and $k_2$ are the elasticity coefficients of the springs, and the system stabilizes when the total energy $\frac{k_1}{2}(x-\alpha)^2 + \frac{k_2}{2}\alpha^2$ is minimum. The second spring therefore acts as a Tykhonov regularizer on the energy of the first spring, and it can be controlled by its elasticity coefficient $k_2$. On the right spring, we fix instead an object of mass $m$. Due to its action, the height of the corresponding blue point also decreases to a new position $\alpha$. The potential energy of this object is $E_2 = mg\alpha$, where $g$ is the magnitude of the Earth's gravitational field, and the system stabilizes when the total energy $\frac{k_1}{2}(x-\alpha)^2 + mg|\alpha|$ is minimum, with the constraint $\alpha \geq 0$. The object therefore acts as an $\ell_1$ regularization, which can be controlled by its mass $m$. Figure 1.4b illustrates the situation when the amount of regularization is small—that is, the second spring on the left is weak, and the object on the right is light. Both systems stabilizes with $\alpha > 0$. Figure 1.4c illustrates the situation when one increases the amount of regularization. On the left side, despite the fact that the spring is strong ($k_2$ is large), the blue point does not touch the ground. On the right, when the object is massive enough the object touches the ground and $\alpha = 0$. In fact, as shown in the previous section, the solution $\alpha$ is obtained by soft-thresholding.

### "Geometrical Explanation" in 2-D and 3-D

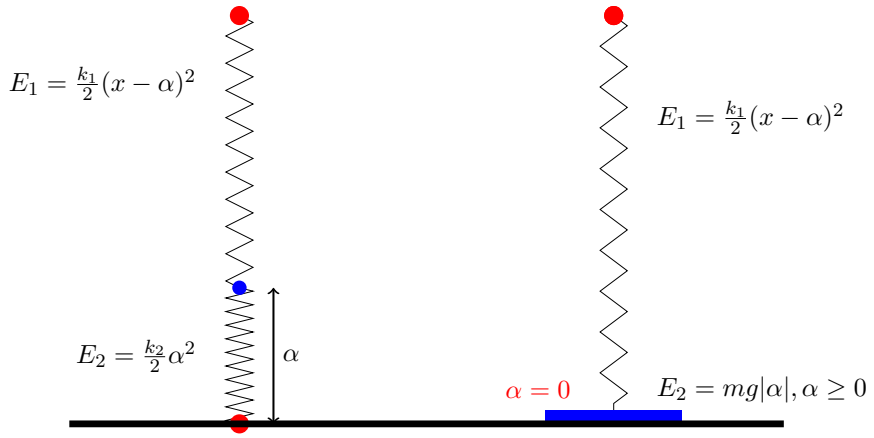
We now present a more classical (but still informal) explanation of the sparsity-inducing property of the $\ell_1$-norm based on the geometry of the $\ell_1$-ball. We consider the Lasso formulation of Eq. (1.2). We know from classical convex optimization arguments (Boyd and Vandenberghe, 2004) that there exists a parameter $T > 0$ such that Equation (1.2)




(a) Initial position.




(b) Small regularization, smallest energy state




(c) High regularization, smallest energy state

Figure 1.4: Simple physical illustration of the sparsifying effect of the $\ell_1$-norm compared to the Tikhonov regularization. The system stabilizes for the value of $\alpha$ that minimizes the energy $E_1 + E_2$. See comments in the text.

has the same solution as the following equivalent constrained optimization problem:[7]

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 \leq T.$$

We present in Figures 1.5a and 1.5c the $\ell_1$-balls of radius $T$ in 2 and 3 dimensions, and the level sets of the function $\boldsymbol{\alpha} \mapsto \frac{1}{2}\|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2^2$. At optimality, the level set corresponding to the optimum value $\boldsymbol{\alpha}^\star$ are necessarily tangent to the $\ell_1$-ball of radius $T$. This tangency point is represented by a small red circle in the figures. In Figure 1.5b, we represent a similar situation when using the squared $\ell_2$-norm instead of the $\ell_1$. Whereas the $\ell_1$-ball is anisotropic and encourages a solution to be on one of the axis $x$ or $y$ (corresponding to sparse solutions), the isotropy of the $\ell_2$-ball does not. In the case of the $\ell_1$-norm, illustrated on Figure 1.5a, it becomes more "likely" that the solution ends up on a corner of the ball, even though it is easy to build counter-examples, where the solution ends up on a face. This sparsifying phenomenon is also true in 3-D, as illustrated in Figure 1.5c, and in fact it is even stronger in higher dimensions.

Now that we have given some intuitive explanations of the sparsity-inducing property of the $\ell_1$-norm, we give a more structured sparse regularization, which we will use intensively in this thesis.

### 1.3.3 Beyond the $\ell_1$-norm: Group Sparsity

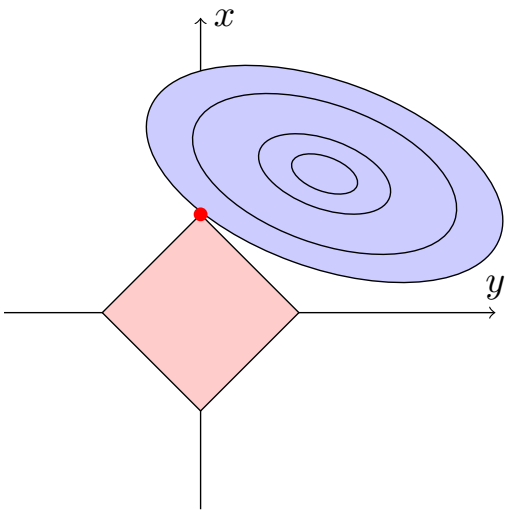
A popular extension of the Lasso is the group Lasso (Yuan and Lin, 2006; Turlach et al., 2005; Obozinski et al., 2009; Bach, 2008). It supposes that variables are structured into predefined groups $g \in \mathcal{G}$, where $\mathcal{G}$ is a partition of $[\![1;p]\!]$. In this context, the sparsity-inducing regularization takes the form:

$$\Omega(\boldsymbol{\alpha}) = \sum_{g\in\mathcal{G}} \|\boldsymbol{\alpha}_g\|,$$

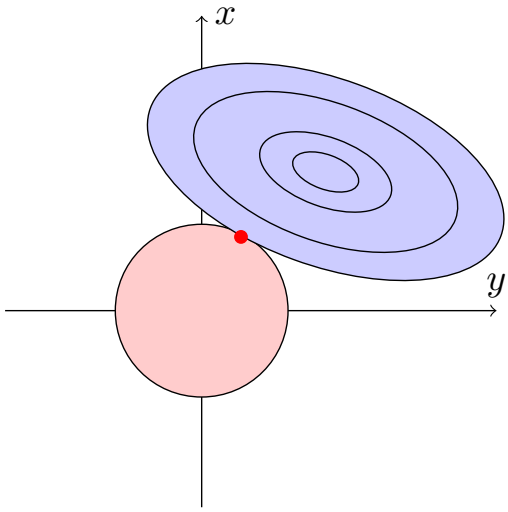
where $\|.\|$ is some norm (in practice, often the $\ell_2$ or $\ell_\infty$-norms). In this case, $\Omega$ is still a norm, and can be interpreted as the $\ell_1$-norm (a sum) of norms of groups, therefore inducing sparsity at the group level.

The goal of using such a regularization is to encode *a priori* knowledge of the sparsity patterns that the coefficients $\boldsymbol{\alpha}$ should have. When such a priori knowledge is given and one knows beforehand that the patterns should be structured in groups, using such a norm can improve the prediction performance and/or interpretability of the learned models (Roth and Fischer, 2008; Yuan and Lin, 2006; Huang et al., 2009; Obozinski et al., 2009). Applications of such norms include for instance multi-task learning, where one is looking for predictors that are shared among different tasks (Obozinski et al., 2009; Quattoni et al., 2009), and multiple kernel learning (Bach, 2008), where groups of variables corresponding to different kernels are selected.

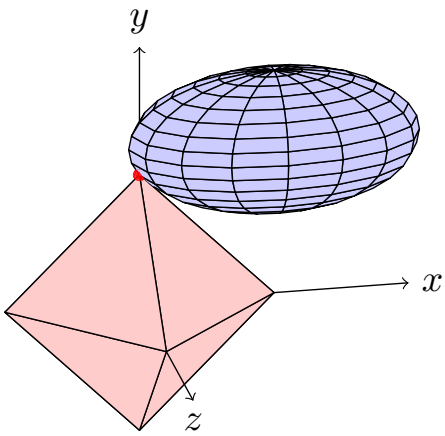
[7] The original formulation of the Lasso proposed by Tibshirani (1996) is actually this constrained formulation.




(a) 2-D case, $\ell_1$-norm




(b) 2-D case, $\ell_2$-norm




(c) 3-D case, $\ell_1$-norm

Figure 1.5: In red, balls for the $\ell_1$-norm in Figures (a) and (c), and $\ell_2$-norm for Figure (b). In blue, some level sets of a quadratic function are plotted. At optimality, the level sets are tangent to the red balls. Corners and edges of the $\ell_1$-ball correspond to sparse solutions.

We present a concrete application of this group Lasso regularization to image processing in Chapter 4. Another generalization of the group Lasso to the case of overlapping groups have been proposed by Zhao et al. (2009); Jenatton et al. (2009); Jacob et al. (2009); Baraniuk et al. (2010). These will be discussed in Chapter 3.

## 1.4 Optimization for Sparse Regularized Problems

Sections 1.4.1, 1.4.2 and the benchmark presented in Section 1.4.5 are based on material from the book chapter:

F. Bach, R. Jenatton, J. Mairal and G. Obozinski. Convex Optimization with Sparsity-Inducing Norms. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, 2011, to appear.

We present here optimization tools and algorithms for solving sparse regularized machine learning and signal processing problems. This section is relatively independent from the rest of the manuscript. It is therefore not mandatory to read it in details before the remaining chapters, but it can be referred to whenever necessary. Section 1.4.1 introduces classical material for non-smooth optimization. Sections 1.4.2 to 1.4.6 give some keys for solving sparse decomposition problems. In particular, we present in Section 1.4.5 a benchmark comparing a large class of methods for solving the Lasso in different scenarii. Section 1.4.8 briefly presents network flow optimization and its connection with sparse methods.

### 1.4.1 Duality and Non-Smooth Convex Optimization

We describe in this section important tools to study non-smooth convex optimization problems related to sparse methods. Most of them can be found in classical books on convex optimization (Boyd and Vandenberghe, 2004; Bertsekas, 1999; Borwein and Lewis, 2006; Nocedal and Wright, 1999), but for self-containedness reasons, we present here a few of them. We consider again the general formulation of Eq. (1.1), which we recall below

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \Big[ g(\boldsymbol{\alpha}) \triangleq f(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha}) \Big],$$

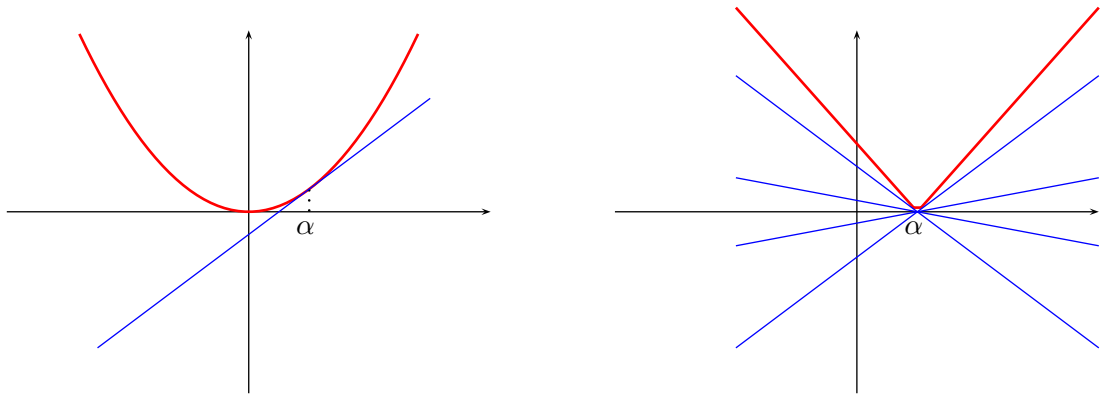
but we restrict $\Omega$ to be a *norm* (and therefore a convex function).

**Subgradients**

Given a convex function $g : \mathbb{R}^p \to \mathbb{R}$ and a vector $\boldsymbol{\alpha}$ in $\mathbb{R}^p$, let us define the *subdifferential* of $g$ at $\boldsymbol{\alpha}$ as

$$\partial g(\boldsymbol{\alpha}) \triangleq \{\boldsymbol{\kappa} \in \mathbb{R}^p \mid g(\boldsymbol{\alpha}) + \boldsymbol{\kappa}^\top(\boldsymbol{\alpha}' - \boldsymbol{\alpha}) \leq g(\boldsymbol{\alpha}') \text{ for all vectors } \boldsymbol{\alpha}' \in \mathbb{R}^p\}.$$

The elements of $\partial g(\boldsymbol{\alpha})$ are called the *subgradients* of $g$ at $\boldsymbol{\alpha}$. This definition admits a clear geometric interpretation: Any subgradient $\boldsymbol{\kappa}$ in $\partial g(\boldsymbol{\alpha})$ defines an affine function $\boldsymbol{\alpha}' \mapsto g(\boldsymbol{\alpha}) + \boldsymbol{\kappa}^\top(\boldsymbol{\alpha}' - \boldsymbol{\alpha})$ which is tangent to the graph of the function $g$ at $\boldsymbol{\alpha}$. Moreover, there is a bijection (one-to-one correspondence) between such "tangent affine functions" and the subgradients. We illustrate this property in Figure 1.6.




(a) Smooth case (b) Non-smooth case

Figure 1.6: Gradients and subgradients for smooth and non-smooth functions. Red curves represent the graph of a function. Blue lines represent subgradients of this function at a point $\alpha$. On the left, the function is smooth and the unique subgradient corresponds to the tangent line. On the right, the function is not differentiable and the subgradients are not unique.

Let us now illustrate how subdifferential can be useful for studying nonsmooth optimization problems with the following classical proposition (see Borwein and Lewis, 2006):

**Proposition 1 (Subgradients at optimality)**
*For any convex function $g : \mathbb{R}^p \to \mathbb{R}$, a point $\boldsymbol{\alpha}$ in $\mathbb{R}^p$ is a global minimum of $g$ if and only if vector $0$ belongs to $\partial g(\boldsymbol{\alpha})$.*

Note that the concept of subdifferential is mainly useful for nonsmooth functions. If $g$ is differentiable in $\boldsymbol{\alpha}$, the set $\partial g(\boldsymbol{\alpha})$ is indeed the singleton $\{\nabla g(\boldsymbol{\alpha})\}$, and the condition $0 \in \partial g(\boldsymbol{\alpha})$ amounts to the classical first-order optimality condition $\nabla g(\boldsymbol{\alpha}) = 0$.

**Dual Norm and Optimality Conditions**
The next tool we introduce is the dual norm, which is important to study sparsity-inducing regularizations (Jenatton et al., 2009; Bach, 2009; Negahban et al., 2009). It notably comes up in the analysis of estimation bounds (Negahban et al., 2009), and in the designs of active-set strategies (Jenatton et al., 2009). The dual norm $\Omega^*$ of $\Omega$ is defined for any vector $\boldsymbol{\kappa}$ in $\mathbb{R}^p$ by

$$\Omega^*(\boldsymbol{\kappa}) \triangleq \max_{\boldsymbol{\alpha} \in \mathbb{R}^p} \boldsymbol{\kappa}^\top \boldsymbol{\alpha} \quad \text{s.t.} \quad \Omega(\boldsymbol{\alpha}) \leq 1.$$

It is easy to show that in the case of an $\ell_q$-norm, $q \in [1;+\infty]$ the dual norm is the $\ell_{q'}$-norm, with $q'$ in $[1;+\infty]$ such that $\frac{1}{q} + \frac{1}{q'} = 1$. In particular, the $\ell_1$- and $\ell_\infty$-norms are dual to each other, and the $\ell_2$-norm is self-dual.

The dual norm plays a direct role in obtaining optimality conditions for sparse regularized problems. By applying Proposition 1 to Eq. (1.1), we obtain, for instance, that a vector $\boldsymbol{\alpha}$ in $\mathbb{R}^p$ is optimal for Eq. (1.1) if and only if

$$-\frac{1}{\lambda}\nabla f(\boldsymbol{\alpha}) \in \partial\Omega(\boldsymbol{\alpha}) = \begin{cases} \{\boldsymbol{\kappa} \in \mathbb{R}^p;\ \Omega^*(\boldsymbol{\kappa}) \leq 1\} \text{ if } \boldsymbol{\alpha} = 0, \\ \{\boldsymbol{\kappa} \in \mathbb{R}^p;\ \Omega^*(\boldsymbol{\kappa}) \leq 1 \text{ and } \boldsymbol{\kappa}^\top \boldsymbol{\alpha} = \Omega(\boldsymbol{\alpha})\} \text{ otherwise.} \end{cases} \tag{1.3}$$

We have presented a useful duality tool for norms. More generally, there exists a related concept for convex functions, which we now introduce.

**Fenchel Conjugate and Duality Gaps**

Let us denote by $f^*$ the Fenchel conjugate of a convex function $f$ (Borwein and Lewis, 2006), defined by

$$f^*(\boldsymbol{\kappa}) \triangleq \sup_{\boldsymbol{\alpha} \in \mathbb{R}^p} [\boldsymbol{\kappa}^\top \boldsymbol{\alpha} - f(\boldsymbol{\alpha})].$$

The Fenchel conjugate is related to the dual norm. Let us define the indicator function $\mathbf{1}_\Omega$ such that $\mathbf{1}_\Omega(\boldsymbol{\alpha})$ is equal to 0 if $\Omega(\boldsymbol{\alpha}) \leq 1$ and $+\infty$ otherwise. Then, $\mathbf{1}_\Omega$ is a convex function and its conjugate is exactly the dual norm $\Omega^*$. For many objective function, the Fenchel conjugate admits closed forms, and can therefore be computed efficiently (Borwein and Lewis, 2006). In this case, it is useful for monitoring the convergence of optimization algorithms with duality gaps, as illustrated by the following proposition:

**Proposition 2 (Duality for Problem (1.1))**
*If $f^*$ and $\Omega^*$ are respectively the Fenchel conjugate of $f$ and the dual-norm of $\Omega$,*

$$\max_{\boldsymbol{\kappa} \in \mathbf{R}^p; \Omega^*(\boldsymbol{\kappa}) \leq \lambda} -f^*(\boldsymbol{\kappa}) \leq \min_{\boldsymbol{\alpha} \in \mathbf{R}^p} f(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha})$$

*Moreover, if the domain of $f$ is non-empty, strong duality holds and the inequality becomes an equality.*

Therefore, if $\boldsymbol{\alpha}^\star$ is a solution of Eq. (1.1), and $\boldsymbol{\alpha}, \boldsymbol{\kappa}$ in $\mathbb{R}^p$ such that $\Omega^*(\boldsymbol{\kappa}) \leq \lambda$, the following inequality holds

$$f(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha}) \geq f(\boldsymbol{\alpha}^\star) + \lambda\Omega(\boldsymbol{\alpha}^\star) \geq -f^*(\boldsymbol{\kappa}). \tag{1.4}$$

The difference between the left and right term of Eq. (1.4) is called a duality gap. It represents the difference between the value of the primal objective function $f(\boldsymbol{\alpha})+\lambda\Omega(\boldsymbol{\alpha})$ and a dual objective function $-f^*(\boldsymbol{\kappa})$, where $\boldsymbol{\kappa}$ is a dual variable. Duality gaps are important in convex optimization. By upperbounding the difference between the current value of an objective function and the optimal value, they define proper stopping criterion for iterative optimization algorithms. Finding a good dual variable $\boldsymbol{\kappa}$ when minimizing

a primal objective function is easy in many cases. Given a primal variable $\boldsymbol{\alpha}$, we often choose the dual variable $\boldsymbol{\kappa} = \frac{\lambda}{\max(\Omega^*(\nabla f(\boldsymbol{\alpha})),\lambda)}\nabla f(\boldsymbol{\alpha})$, which guarantees the duality gap to be zero at optimality. When $\boldsymbol{\alpha} = \boldsymbol{\alpha}^\star$, the conditions presented in Eq. (1.3) are satisfied. It follows that $\boldsymbol{\kappa} = \nabla f(\boldsymbol{\alpha})$ with $\Omega^*(\boldsymbol{\kappa}) \leq \lambda$, and $\boldsymbol{\kappa}^\top \boldsymbol{\alpha} = \lambda\Omega(\boldsymbol{\alpha})$. Since $f$ is differentiable, it is also easy to show that $f^*(\boldsymbol{\kappa}) = \boldsymbol{\kappa}^\top \boldsymbol{\alpha} - f(\boldsymbol{\alpha})$. Therefore, $-f^*(\boldsymbol{\kappa}) = f(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha})$ and the duality gap is zero.

Note that in many of the formulations we are going to introduce, the function $f$ has a particular form $f(\boldsymbol{\alpha}) = \tilde{f}(\mathbf{D}\boldsymbol{\alpha})$, where $\tilde{f}$ is an auxiliary function, and $\mathbf{D}$ a dictionary matrix. In this case, one may be interested in the Fenchel conjugate $\tilde{f}^*$ instead of $f^*$. Fenchel conjugacy naturally extends to this case (see for more details Borwein and Lewis, 2006, Theorem 3.3.5). We present more concrete examples in Appendix D.2 with a toolbox implementing several solvers for sparse methods, where the convergence of the different optimization methods are monitored with such duality gaps.

### 1.4.2 Least Angle Regression - Homotopy

We present in this section a dedicated active-set method for solving the Lasso problem (Tibshirani, 1996), also known as basis pursuit (Chen et al., 1998), which is presented in Eq. (1.2). Under mild assumptions, (which we will detail later) the solution of Eq. (1.2) is unique, and we denote it by $\boldsymbol{\alpha}^\star(\lambda)$. Let us also recall the definition of the *regularization path*, which is the function $\lambda \mapsto \boldsymbol{\alpha}^\star(\lambda)$ that associates with a regularization parameter $\lambda$ the corresponding solution $\boldsymbol{\alpha}^\star(\lambda)$. We will show that this function is piecewise linear, an interesting property that leads both to an efficient algorithm presented in this section, and to a better understanding of the Lasso formulation. This behavior was illustrated in Figure 1.1, where the entries of $\boldsymbol{\alpha}^\star(\lambda)$ for particular instances of the Lasso are represented as functions of $\lambda$.

The regularization path can therefore be characterized by a set of contiguous linear segments. It is now appealing to build an algorithm that finds a solution of Eq. (1.2) for a particular value of $\lambda$, for which finding this solution is trivial, and then follows the piecewise-linear path, computing the directions of the current linear parts, and the points where the direction changes (kinks). This piecewise linearity property was first discovered and exploited by Markowitz (1952) in the context of portfolio selection, revisited by Osborne et al. (2000a) describing an *homotopy* algorithm, and popularized by Efron et al. (2004) with the LARS algorithm. Even though the basic version of LARS is a bit different from the procedure we have just described, it is closely related, and indeed a simple modification makes it possible to obtain the full regularization path of Eq. (1.2).

Let us now construct the solution path. Applying the optimality conditions presented in Eq. (1.3) to the Lasso formulation for a fixed value of $\lambda$ yields

$$\forall j \in [\![1;p]\!], \left\{ \begin{array}{rcll} |\mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)| & \leq & \lambda & \text{if } \boldsymbol{\alpha}^\star_j = 0 \\ \mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star) & = & \lambda \operatorname{sign}(\boldsymbol{\alpha}^\star_j) & \text{if } \boldsymbol{\alpha}^\star_j \neq 0, \end{array} \right. \tag{1.5}$$

where $\mathbf{d}^j$ denotes the $j$-th column of $\mathbf{D}$, and $\boldsymbol{\alpha}^\star_j$ the $j$-th entry of $\boldsymbol{\alpha}^\star$. We define the set of variables $\Lambda \triangleq \{j \in [\![1;p]\!]; |\mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)| = \lambda\}$, and the vector $\boldsymbol{\varepsilon} \triangleq \text{sign}\,(\mathbf{D}^\top(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star))$. We assume the matrix $\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda$ to be invertible (which is a necessary and sufficient condition to guarantee the uniqueness of $\boldsymbol{\alpha}^\star$), and it follows from Eq. (1.5) that

$$\boldsymbol{\alpha}^\star_\Lambda(\lambda) = (\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda)^{-1}(\mathbf{D}_\Lambda^\top\mathbf{x} - \lambda\boldsymbol{\varepsilon}_\Lambda).$$

This is an important point: if one knows in advance the set $\Lambda$ and the signs $\boldsymbol{\varepsilon}_\Lambda$, then the solution $\boldsymbol{\alpha}^\star(\lambda)$ admits a simple closed-form, showing that the difficulty of the Lasso is essentially to find the pair $(\Lambda, \boldsymbol{\varepsilon}_\Lambda)$.

Moreover, when $\Lambda$ and $\boldsymbol{\varepsilon}_\Lambda$ are fixed, the function $\lambda \mapsto (\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda)^{-1}(\mathbf{D}_\Lambda^\top\mathbf{x} - \lambda\boldsymbol{\varepsilon}_\Lambda)$ is affine in $\lambda$. With this observation in hand, we can now present the main steps of the path-following algorithm. It basically starts from a trivial solution of the regularization path, follows the path by exploiting this formula, updating $\Lambda$ and $\boldsymbol{\varepsilon}_\Lambda$ whenever needed so that optimality conditions (1.5) remain satisfied. This procedure requires some assumptions—namely that **(A)** the regularization path is unique (which is equivalent to assuming $\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda$ always invertible), and **(B)** that updating $\Lambda$ along the path consists of adding or removing from this set a single variable at the same time. Concretely, we proceed as follows

1. Set $\lambda$ to $\|\mathbf{D}^\top\mathbf{x}\|_\infty$ for which it is easy to show from Eq. (1.5) that $\boldsymbol{\alpha}^\star(\lambda) = 0$ (trivial solution). This gives us a starting point on the regularization path.
2. Set $\Lambda \triangleq \{j \in [\![1;p]\!]; |\mathbf{d}^{j\top}\mathbf{x}| = \lambda\}$, assuming $|\Lambda| = 1$ (assumption **[B]**).
3. Follow the regularization path by decreasing the value of $\lambda$, with the formula $\boldsymbol{\alpha}^\star_\Lambda(\lambda) = (\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda)^{-1}(\mathbf{D}_\Lambda^\top\mathbf{x} - \lambda\boldsymbol{\varepsilon}_\Lambda)$ keeping $\boldsymbol{\alpha}^\star_{\Lambda^C} = 0$, until one of the following events occurs
   - There exists $j$ in $\Lambda^C$ such that $|\mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)| = \lambda$. Then, add $j$ to the set $\Lambda$.
   - There exists $j$ in $\Lambda$ such that a non-zero coefficient $\boldsymbol{\alpha}^\star_j$ hits zero. Then, remove $j$ from $\Lambda$.

   We suppose that only one of such events can occur at the same time (assumption **[B]**). It is also easy to show that the value of $\lambda$ corresponding to the next event can be obtained in closed form, using the fact that for a fixed pair $(\Lambda, \boldsymbol{\varepsilon})$, the quantities $\boldsymbol{\alpha}^\star_j$ and $\mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)$ for all $j$ in $[\![1;p]\!]$ are also affine in $\lambda$.
4. Go back to 3

Let us now briefly discuss assumptions **(A)** and **(B)**. When the matrix $\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda$ is not invertible, the regularization path is non-unique, and the algorithm fails. This can easily be fixed by addressing instead a slightly modified formulation. It is indeed possible to consider the elastic-net formulation of Zou and Hastie (2005)—that is, with $\Omega(\boldsymbol{\alpha}) = \lambda\|\boldsymbol{\alpha}\|_1 + \frac{\gamma}{2}\|\boldsymbol{\alpha}\|_2^2$, by replacing the matrix $\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda$ by $\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda + \gamma\mathbf{I}_p$, which is positive definite and therefore always invertible. Using a small value for $\gamma$ solves the problem of

non-invertibility of $\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda$ in practice. The second assumption **(B)** can be unsatisfied in practice because of the precision machine. To the best of our knowledge, the algorithm will fail in such cases, but we consider this scenario unlikely.

Now that we are able to follow the regularization path, it is important to notice that we are also able to solve constrained versions of Eq. (1.2), namely

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 \leq T, \tag{1.6}$$

and

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \|\boldsymbol{\alpha}\|_1 \quad \text{s.t.} \quad \|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 \leq \varepsilon, \tag{1.7}$$

These formulations are sometimes said to be "equivalent" to Eq. (1.2) in the sense that for every value of $T$, there exists a value $\lambda$ such that Eq. (1.2) that admits the same solution as Eq. (1.6), and vice versa. This is also true for Eq. (1.7) for every value of $\varepsilon$. They are, however, not equivalent in practice since the relation between $\varepsilon, \lambda$ and $T$ is unknown.

The complexity of the above procedure depends on the number of kinks of the regularization path (which correspond to the number of iterations of the algorithm). It is of course possible to stop the algorithm before its end, if one is not interested in the full path. Even though it is possible to build examples where the number of kinks is large, we often observe in practice that the event where one variable gets out of the active set is rare. The complexity also depends on the implementation. By maintaining the values of $\mathbf{d}^{j\top}(\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}^\star)$ and a Cholesky decomposition of $(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1}$, it is possible to obtain an implementation in $O(psm+ps^2+s^3)$ operations, where $s$ is the number of iterations of the algorithm, with a memory cost in $O(p^2)$. The product $psm$ corresponds to the computation of $\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda$, $ps^2$ to the updates of the correlations $\mathbf{d}^{j\top}(\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}^\star)$ along the path, and $s^3$ to the Cholesky decomposition of $(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1}$.

One can observe from this analysis that the path-following LARS algorithm can be efficient for solving small-scale problems, when the solution one is looking for is sparse ($s$ is small), with a smaller cost than a single $p\times p$ matrix inversion, which is $0(p^3)$. This algorithm is also efficient with highly correlated features, as long as the matrix $\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda$ remains invertible. A fast Cholesky-based implementation of this algorithm is available in the toolbox SPAMS, which we present in Section D.1.

### 1.4.3 Proximal Methods

Proximal methods play an important role in non-smooth optimization. They generalize first-order gradient descent algorithms to handle non-smooth components. This section briefly introduce these methods in a slightly restricted but useful framework (for a more detailed review and general framework, see Combettes and Pesquet 2010).

In the context of this thesis, we apply these methods to convex optimization problems of the same form as Eq. (1.1), with $f$ convex and differentiable with a Lipschitz continuous gradient, and $\Omega$ a non-smooth convex function. Whereas it is often possible to address this kind of optimization problems using subgradient descent algorithms,

proximal methods are preferred because of both theoretically and practically faster convergence rates, which we will detail in the sequel.

The most basic variant of these methods is an iterative procedure which, at step $k$, updates the current estimate $\boldsymbol{\alpha}^k$ by solving a *proximal* problem, defined as follows

$$\boldsymbol{\alpha}^{k+1} \leftarrow \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \Big[ f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)^\top (\boldsymbol{\alpha} - \boldsymbol{\alpha}^k) + \lambda\Omega(\boldsymbol{\alpha}) + \frac{L}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^k\|_2^2 \Big], \tag{1.8}$$

where $f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)^\top (\boldsymbol{\alpha} - \boldsymbol{\alpha}^k)$ is a linear approximation of $f$ around the current estimate $\boldsymbol{\alpha}^k$, the quadratic term $\frac{L}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^k\|_2^2$ keeps the update of $\boldsymbol{\alpha}$ in a neighborhood of $\boldsymbol{\alpha}^k$, where the linear approximation is correct, and $L > 0$ is a parameter. It is possible to show that when $L$ is well chosen (actually larger or equal to the inverse of the Lipschitz constant of $\nabla f$), this optimization scheme converges to the solution of the original problem. This provides a simple scheme for solving Eq. (1.1), supposing that one knows how to efficiently solve Eq. (1.8). Finding automatically a good value for $L$ is also easy, using practical line-search strategies (see Nesterov, 2007; Beck and Teboulle, 2009).

Note that Eq. (1.8) can equivalently be rewritten as

$$\boldsymbol{\alpha}^{k+1} \leftarrow \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \Big[ \frac{1}{2}\Big\|\boldsymbol{\alpha} - (\boldsymbol{\alpha}^k - \frac{1}{L}\nabla f(\boldsymbol{\alpha}^k))\Big\|_2^2 + \frac{\lambda}{L}\Omega(\boldsymbol{\alpha}),$$

meaning that the new estimate $\boldsymbol{\alpha}^{k+1}$ should be close to the quantity $\boldsymbol{\alpha}^k - \frac{1}{L}\nabla f(\boldsymbol{\alpha}^k)$ (equivalent to a classical gradient step), while taking into account the non-smooth component $\lambda\Omega(\boldsymbol{\alpha})$. When $\lambda = 0$, we obtain that $\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k - \frac{1}{L}\nabla f(\boldsymbol{\alpha}^k)$.

More generally, we define the *proximal operator* (sometimes called *proximity operator*) associated with our regularization term $\lambda\Omega$ as the function that maps a vector $\mathbf{u}$ in $\mathbb{R}^p$ to the (unique by strong convexity) solution of

$$\min_{\mathbf{v}\in\mathbb{R}^p} \Big[ \frac{1}{2}\|\mathbf{u} - \mathbf{v}\|_2^2 + \lambda\Omega(\mathbf{v}) \Big]. \tag{1.9}$$

This operator was initially introduced by Moreau (1962) to generalize the projection operator onto a convex set. Since it is called many times within proximal algorithms, it has to be solved efficiently. What makes this appealing for sparse methods is that this operator can often be obtained in closed-form. For instance:

- When $\Omega$ is the $\ell_1$-norm—that is, $\Omega(\mathbf{u}) = \|\mathbf{u}\|_1$, the proximal operator is the well-known elementwise soft-thresholding operator introduced in the previous sections:

$$\forall j \in [\![1; p]\!] \quad \mathbf{v}_j \leftarrow \text{sign}(\mathbf{u}_j)(|\mathbf{u}_j| - \lambda)^+ \quad = \begin{cases} 0 & \text{if } |\mathbf{u}_j| \leq \lambda \\ \text{sign}(\mathbf{u}_j)(|\mathbf{u}_j| - \lambda) & \text{otherwise.} \end{cases}$$

- When $\Omega$ is a Group-Lasso penalty with $\ell_2$-norms—that is, $\Omega(\mathbf{u}) = \sum_{g\in\mathcal{G}} \|\mathbf{u}_g\|_2$, with $\mathcal{G}$ being a partition of $[\![1; p]\!]$, the proximal problem is *separable* in every group,

and the solution is a generalization of the soft-thresholding operator to groups of variables:

$$\forall g \in \mathcal{G} \quad \mathbf{v}_g \leftarrow \begin{cases} 0 & \text{if } \|\mathbf{u}_g\|_2 \leq \lambda \\ \frac{\|\mathbf{u}_g\|_2 - \lambda}{\|\mathbf{u}_g\|_2} \mathbf{u}_g & \text{if } \|\mathbf{u}_g\|_2 > \lambda. \end{cases}$$

- When $\Omega$ is a Group-Lasso penalty with $\ell_\infty$-norms—that is, $\Omega(\mathbf{u}) = \sum_{g \in \mathcal{G}} \|\mathbf{u}_g\|_\infty$, the solution is also a group-thresholding operator:

$$\forall g \in \mathcal{G} \quad \mathbf{v}_g \leftarrow \mathbf{u}_g - \Pi_{\|.\|_1 \leq \lambda}[\mathbf{u}_g],$$

where $\Pi_{\|.\|_1 \leq \lambda}$ denotes the orthogonal projection onto the $\ell_1$-ball of radius $\lambda$. Note that when $\|\mathbf{u}_g\|_1 \leq \lambda$, we have a group-thresholding effect, with $\mathbf{v}_g = 0$.

More generally, when dealing with norms, these closed-forms can be derived from a simple relation between the proximal operator and the projection operator onto the ball of the dual norm:

**Lemma 1 (Relation between proximal and projection operator for norms)** *Let $\mathbf{u}$ be a vector in $\mathbb{R}^p$ and let $\mathbf{v}^\star$ be the solution of the proximal operator*

$$\min_{\mathbf{v} \in \mathbb{R}^p} \Big[ \frac{1}{2} \|\mathbf{u} - \mathbf{v}\|_2^2 + \lambda \Omega(\mathbf{u}) \Big], \tag{1.10}$$

*where $\Omega$ is any norm. Then,*

$$\mathbf{v}^\star = \mathbf{u} - \Pi_{\Omega^*(.) \leq \lambda}[\mathbf{u}], \tag{1.11}$$

*Where $\Pi_{\Omega^*(.) \leq \lambda}$ is the orthogonal projector onto the ball of radius $\lambda$ of the dual norm $\Omega^*$.*

The proof can be obtained by using simple calculus rules for computing proximal operators described by Combettes and Pesquet (2010), or by simply writing the Fenchel dual of the proximal problem, which is described in Proposition 2. This directly gives the solution.

This proximal scheme for solving sparse decomposition problems has been the focus of a lot of attention lately and has been revisited several times. It indeed admits variants (essentially concerning line-search strategies for automatically choosing the parameter $L$). We give here a few names under which it is known, to help the reader find his/her way in the literature. Combettes and Pesquet (2010) present a detailed review of proximal methods and call this a *forward-backward splitting* algorithm, Nesterov (2007) call it *gradient method*, and Beck and Teboulle (2009) *iterative shrinkage-thresholding algorithm* (ISTA). Refinements have been proposed by Wright et al. (2009b) under the name SpaRSA, and by Hale et al. (2007) under the name *fixed-point continuation method* (FPC). Nesterov (2007) and Beck and Teboulle (2009) have shown that the value of the objective function decreases as $O(\frac{1}{k})$, and under strong convexity assumptions on $f$, Nesterov (2007) has further shown that it enjoys a linear convergence rate of $O(\rho^k)$, with $0 \leq \rho < 1$. Interestingly, building on early works by Nesterov (1983), accelerated variants of proximal methods have been proposed by Nesterov (2007) and Beck and Teboulle

(2009) with guaranteed convergence rate of $O(\frac{1}{k^2})$, which can be proven to be optimal among first-order methods. In order to enjoy these fast rates, the proximal operator must be computed both efficiently and *exactly*. This is the topic of Chapter 3 for two particular sparsity-inducing norms.

We have implemented in the software presented in Section D.2 the forward-backward (or ISTA) and the accelerated FISTA algorithms of Beck and Teboulle (2009). We show in Section 1.4.5 how these methods compare to other approaches.

### 1.4.4 Coordinate and Block Coordinate Descent Algorithms

We present here a coordinate descent algorithm for solving the Lasso formulation of Eq. (1.2). It was originally introduced by Fu (1998), rediscovered by Daubechies et al. (2004), and recently popularized by Wu and Lange (2008) and Friedman et al. (2007). We first present the basic algorithm, and then show how it extends to the group Lasso in some specific settings.

Coordinate descent is a procedure that iteratively fixes every entry but one of the current estimate $\boldsymbol{\alpha}$, and optimizes with respect to the selected entry. It cycles among the coordinates, solving each time simple sub-problems that admit closed form solutions. For instance, supposing the columns of $\mathbf{D}$ have unit $\ell_2$-norm, updating the entry $\boldsymbol{\alpha}_j$ can be done as follows

$$\begin{aligned}
\boldsymbol{\alpha}_j &\leftarrow \arg\min_{\boldsymbol{\alpha}_j \in \mathbb{R}} \left[\frac{1}{2}\|\mathbf{x} - \sum_{i \neq j} \boldsymbol{\alpha}_i \mathbf{d}^i - \boldsymbol{\alpha}_j \mathbf{d}^j\|_2^2 + \lambda|\boldsymbol{\alpha}_j|\right] \\
&\leftarrow \arg\min_{\boldsymbol{\alpha}_j \in \mathbb{R}} \left[\frac{1}{2}\big(\mathbf{d}^{j\top}(\mathbf{x} - \sum_{i \neq j} \boldsymbol{\alpha}_i \mathbf{d}^i) - \boldsymbol{\alpha}_j\big)^2 + \lambda|\boldsymbol{\alpha}_j|\right] \\
&\leftarrow \operatorname{sign}(\mathbf{c}_j)(|\mathbf{c}_j| - \lambda)^+ \quad \text{with} \quad \mathbf{c}_j \triangleq \mathbf{d}^{j\top}(\mathbf{x} - \sum_{i \neq j} \boldsymbol{\alpha}_i \mathbf{d}^i).
\end{aligned}$$

This is the simple soft-thresholding operation introduced earlier. This coordinate descent procedure is appealing since it is simple. Supposing that the matrix $\mathbf{D}^\top\mathbf{D}$ is pre-computed, fast implementations maintain the values of the quantity $\mathbf{D}^\top(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})$. Then, updating a coordinate cost $O(1)$ operations if its value does not change, and $O(p)$ otherwise. Such an implementation is available in the software SPAMS presented in Appendix D.

The convergence properties of such an algorithm are relatively weak. Coordinate descent algorithms for minimizing non-differentiable functions are not convergent in general (see Tseng, 2001, for sufficient conditions in such non-differentiable settings). It is however possible to rewrite equivalently the Lasso as a smooth differentiable optimization problem under separable constraints:

$$\min_{\boldsymbol{\alpha}_+, \boldsymbol{\alpha}_- \in \mathbb{R}^p} \left[\frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}_+ + \mathbf{D}\boldsymbol{\alpha}_-\|_2^2 + \lambda\boldsymbol{\alpha}_+^\top\mathbf{1}_p + \lambda\boldsymbol{\alpha}_-^\top\mathbf{1}_p\right] \quad \text{s.t.} \quad \boldsymbol{\alpha}_+ \geq 0, \boldsymbol{\alpha}_- \geq 0.$$

We have here split the vector $\boldsymbol{\alpha}$ into two vectors $\boldsymbol{\alpha}_+$ and $\boldsymbol{\alpha}_-$ in $\mathbb{R}^p$ with nonnegativity constraints. It is easy to show that this problem is equivalent to the Lasso, and that the

coordinate-descent scheme we have introduced is also equivalent to a coordinate-descent algorithm for this new formulation. In such a setting, and under additional conditions that are satisfied here, we know that all the limit points of the sequence of estimates of the solution are stationary points of the Lasso (see Bertsekas, 1999, proposition 2.7.1). To the best of our knowledge, no convergence rate is available, but we show in Section 1.4.5 that the method can be competitive in certain situations.

This algorithm extends in a straightforward way to the group Lasso, when the dictionary elements corresponding to a same group are orthogonal to each other. Whereas this might seem a strong limitation, it turns out to be the case in some practical situation, such as simultaneous sparse coding (Tropp et al., 2006; Tropp, 2006) and multi-task formulations (Obozinski et al., 2009). In this case, the natural extension of the algorithm is a block-coordinate descent scheme, where one iteratively updates the entries of $\boldsymbol{\alpha}$ corresponding to a group, while fixing the other ones. This can be written for a group $g$ in $\mathcal{G}$:

$$\begin{aligned}\boldsymbol{\alpha}_g &\leftarrow \arg\min_{\boldsymbol{\alpha}_g \in \mathbb{R}^{|g|}} \Big[\frac{1}{2}\|\mathbf{x} - \sum_{h \neq g} \mathbf{D}_h \boldsymbol{\alpha}_h - \mathbf{D}_g \boldsymbol{\alpha}_g\|_2^2 + \lambda \Omega(\boldsymbol{\alpha}_g)\Big] \\ &\leftarrow \arg\min_{\boldsymbol{\alpha}_g \in \mathbb{R}^{|g|}} \Big[\frac{1}{2}\|\mathbf{D}_g^\top(\mathbf{x} - \sum_{h \neq g} \mathbf{D}_h \boldsymbol{\alpha}_h) - \boldsymbol{\alpha}_g\|_2^2 + \lambda \Omega(\boldsymbol{\alpha}_g)\Big],\end{aligned}$$

and the solution is given by computing a proximal operator associated with the norm $\Omega$. We gave such closed forms in the previous section in the case of the $\ell_2$- and $\ell_\infty$-norms. Note that a variant of coordinate descent algorithms have been proposed by Tseng (2001) when there is no closed form for updating a variable.

### 1.4.5 The Lasso: Which Algorithm to Choose and When?

We present in this section a large benchmark evaluating the performance of various optimization methods for solving the Lasso. As already mentioned before, the literature on the topic is vast, and there is no clear consensus about which method does perform the best. The purpose of this section is to experimentally clarify this open question. To do so, we have designed a benchmark that takes into account several criteria which significantly influence the convergence speed of all algorithms. More specifically, our benchmark obeys the following rules:

- **Efficiency of implementations**: We use the languages C or C++ and efficient BLAS and LAPACK libraries for basic linear algebra operations, with the hope that the running times of our software correctly reflects the true number of operations required by every algorithm.

- **Exhaustivity**: We have chosen to compare what we believe are the main approaches used in the literature, namely the LARS algorithm, coordinate-descent (CD), reweighted-$\ell_2$ scheme (Re-$\ell_2$), a simple proximal method (ISTA), and its accelerated version (FISTA). We also include in the comparison generic tools such

as a subgradient descent algorithm (SG), and a commercial software (Mosek) for cone programming (CP) and quadratic programming (QP) problems. The reader should refer to (Bach et al., 2011) for all methods that have not been presented here.

- **Influence of scale**: We measure the performance of algorithms for several problems sizes. We design a small-scale experiment with $n = 200, p = 200$, and a medium/large scale one with $n = 2000, p = 10000$. With this collection of settings, we compare the influence of the parameters $n$ and $p$ for all algorithms.

- **Influence of correlation**: When the dictionary is orthogonal, the Lasso admits a closed-form solution and is easy to solve. When the columns are highly correlated, the optimization problem can become ill-conditioned and difficult. To evaluate the robustness of the different methods to this criterion, we generate dictionaries with three *levels of correlation* between the columns.

- **Influence of the regularization**: We measure the performance for three different levels of regularization, corresponding to different sparsities of the solutions.

- **Influence of the required precision**: We report the value of the objective function versus the time of computation. When a low precision is required, a method that quickly provide a rough solution might be preferred.

We therefore compare 8 methods for 18 different conditions (2 scales $\times$ 3 levels of correlation $\times$ 3 levels of regularization).

We generated dictionary matrices as follows. For the scenario with low correlation, all entries of $\mathbf{D}$ are independently drawn from a Gaussian distribution $\mathcal{N}(0, 1/\sqrt{n})$, which is often a setting used for evaluating optimization algorithms in the literature. For the scenario with medium correlation, we draw the rows of the matrix $\mathbf{D}$ from a multivariate Gaussian distribution in a way such that the average absolute value of the correlation between two different columns is four times the one of the scenario with low correlation. We proceed the same way for the scenario with high correlation, increasing again the amount of correlation. Test data vectors $\mathbf{y} = \mathbf{D}\boldsymbol{\alpha} + \mathbf{n}$ where $\boldsymbol{\alpha}$ are randomly generated, with three levels of sparsity to be used with the three different levels of regularization. The variable $\mathbf{n}$ is a noise vector whose entries are i.i.d. samples from a Gaussian distribution $\mathcal{N}(0, 0.1\|\mathbf{D}\boldsymbol{\alpha}\|_2/\sqrt{n})$.

In the low regularization setting, the sparsity of the vectors $\boldsymbol{\alpha}$ is $s = 0.5\min(n, p)$, in the medium regularization one $s = 0.1\min(n, p)$, and in the high regularization one $s = 0.01\min(n, p)$, corresponding to fairly sparse vectors. For the subgradient method (SG), we take the step size to be equal to $a/(k+b)$, where $k$ is the iteration number, and $(a, b)$ are the best[8] parameters selected in a logarithmic grid $(a, b) \in \{10^{-3}, 1O^{-2}, \ldots, 10\} \times \{10^2, 10^3, 10^4\}$ ; we proceeded this way not to disadvantage SG by an arbitrary choice of stepsize.

[8] "The best step size" is understood here as being the step size leading to the smallest objective function after 500 iterations.

We report the value of the objective functions for every combination of criterion, as a function of computation time in Figures 1.7 and 1.8. All reported results are obtained by averaging 5 runs of each experiment on a single-core of a 3.07GHz CPU with 8Go of memory. Interestingly, we observe that the hierarchy between the different methods significantly changes with the scenario. We can now summarize our conclusions for every class of method:

- **LARS**: For the small-scale problem, LARS outperforms every other method for almost every scenario and precision regime. It is therefore *definitely the right choice for small-scale settings.* With a computational complexity of $O(ps^2 + pns)$ and memory complexity of $O(ps)$,[9] its scalability is, however, a bit limited. When the matrix $\mathbf{D}^\top\mathbf{D}$ is pre-computed, its complexity goes down to $O(ps^2 + ps)$, but it is not the case for our benchmark.

  One of its main advantages is that unlike first-order methods, the LARS complexity does not depend on the correlation in the dictionary, but only on the sparsity $s$ of the solution. In our large-scale settings, LARS has proven to be competitive either when the solution is *very sparse* (high regularization), or when there is *high correlation* in the dictionary (in that case, other methods do not perform as well). One important advantage of the LARS is that it gives an exact solution and computes the whole regularization path.

- **Proximal methods (ISTA, FISTA)**: Our first conclusion is that FISTA has always been better than ISTA except for high regularization or low correlation, where both methods have a similar performance. These methods are almost always outperformed by LARS in the small-scale setting, except for *low precision and low correlation.*

  They *suffer from correlated features* since their convergence rate is proportional to the Lipschitz constant of the gradient of $f$, which itself grows with the amount of correlation. They are *well adapted to large-scale settings, with low or medium correlation.*

- **Coordinate descent (CD)**: To the best of our knowledge, no theoretical convergence rate is available for this method. The empirical convergence rate we have observed has been relatively surprising. In every experiment, we observe a "warm-up" phase where updating one coordinate requires computing one column of the matrix $\mathbf{D}^\top\mathbf{D}$ (which we store into memory). During this phase, the convergence is very slow. When all columns of $\mathbf{D}^\top\mathbf{D}$ are computed, the convergence rate becomes often empirically linear.

  Its performance in the *small-scale setting is relatively good* (even though always behind LARS), but *less efficient in the large-scale one.* For a reason we can not explain, *it does not suffer much from correlated features.* Like LARS, this method could also benefit from an off line pre-computation of $\mathbf{D}^\top\mathbf{D}$.

[9] Note that we did not take into account the memory complexity in our benchmark.

- **Reweighted-$\ell_2$**: This method has proven to be relatively disappointing in all our experiments and has never taken the lead against other dedicated methods.
- **Generic Methods (SG, QP, CP)**: As expected, generic methods have proven not to be adapted for solving the Lasso and are always outperformed by dedicated ones such as LARS.

### 1.4.6 Greedy Methods

We have presented so far the problem of sparse decomposition with a convex optimization point of view, considering the formulation of Eq. (1.1) where $\Omega$ is a sparsity-inducing norm, often used as a proxy for the $\ell_0$ pseudo-norm. We present here algorithms that directly address the following $\ell_0$-decomposition problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_0 \leq s, \tag{1.12}$$

where $s$ is the desired sparsity of the solution. Approaches providing an approximate solution to this problem are greedy procedures, and usually do not provide the global optimum since the problem is NP-hard. However, they have some optimality guarantees in a few cases as shown by Tropp (2004). Empirically, they have shown to provide local optima yielding good results in many image processing applications, as shown in Chapter 4. They are known as forward selection techniques in statistics (Weisberg, 1980), and matching pursuit algorithms in signal processing (Mallat and Zhang, 1993).

We present here two variants called matching pursuit and orthogonal matching pursuit. Both approaches start with a null vector $\boldsymbol{\alpha}$, and iteratively update one entry in $\boldsymbol{\alpha}$ until the sparsity of $\boldsymbol{\alpha}$ reaches the threshold $s$.

- **Matching pursuit** (MP) selects at each step the dictionary element $\mathbf{d}^{\hat{\imath}}$ that is the most correlated with the residual according to the formula

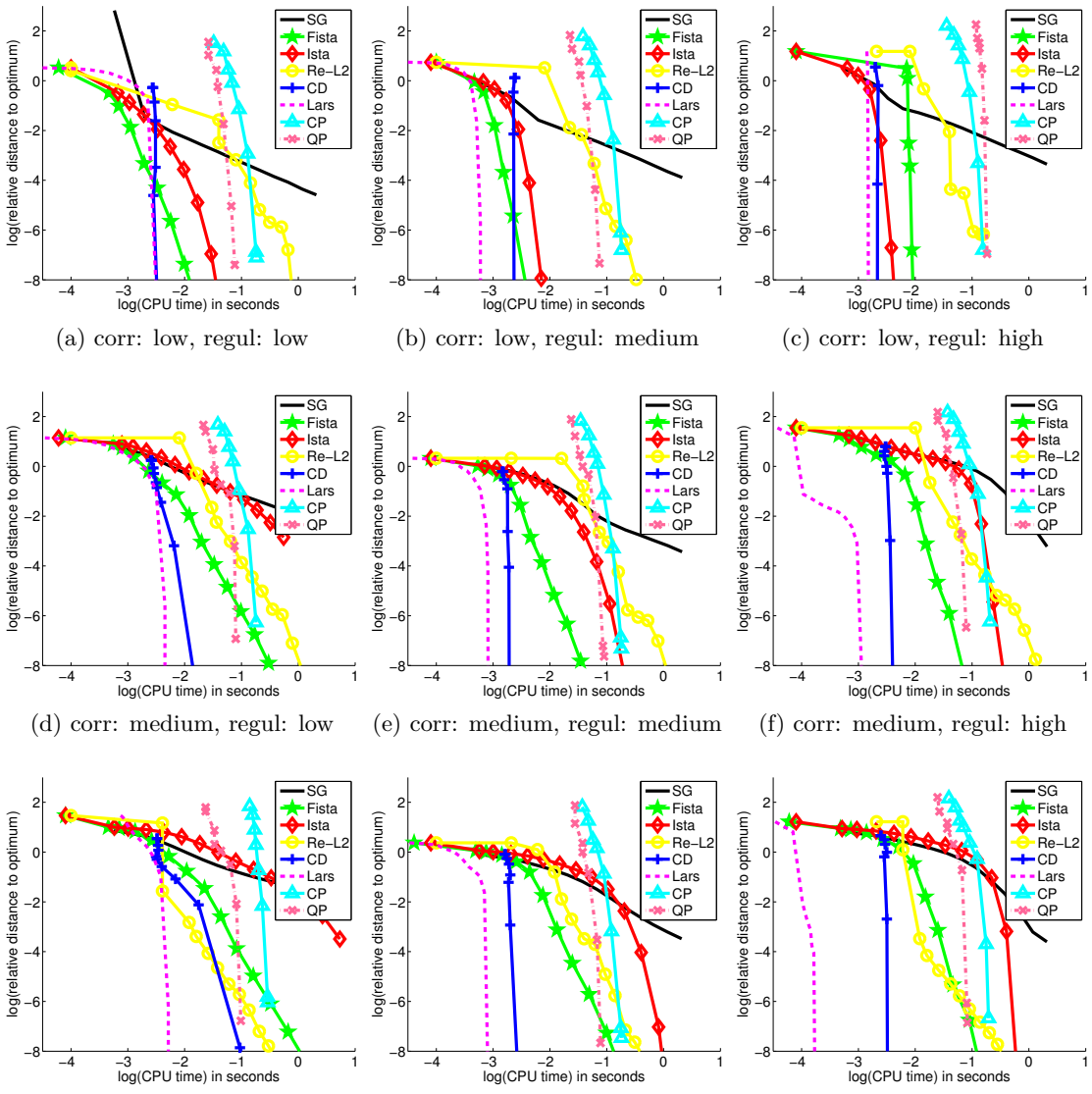
$$\hat{\imath} \leftarrow \arg\min_{i\in[\![1;p]\!]} |\mathbf{d}^{i\top}\mathbf{r}|,$$

  where $\mathbf{r}$ denotes the residual $\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}$. Then, the residual is projected on the line generated by $\mathbf{d}^{\hat{\imath}}$:

$$\begin{aligned}\boldsymbol{\alpha}_{\hat{\imath}} &\leftarrow \boldsymbol{\alpha}_{\hat{\imath}} + \mathbf{d}^{\hat{\imath}\top}\mathbf{r}\\ \mathbf{r} &\leftarrow \mathbf{r} - (\mathbf{d}^{\hat{\imath}\top}\mathbf{r})\mathbf{d}^{\hat{\imath}}.\end{aligned}$$

  Matching Pursuit can in fact be interpreted as a non-cyclic coordinate descent algorithm. It is guaranteed to decrease the objective function at each iteration, but is not guaranteed to converge in a finite number of steps.

- **Orthogonal matching pursuit** (OMP) improves upon Matching Pursuit by ensuring that the residual of the decomposition is always *orthogonal to all previously*

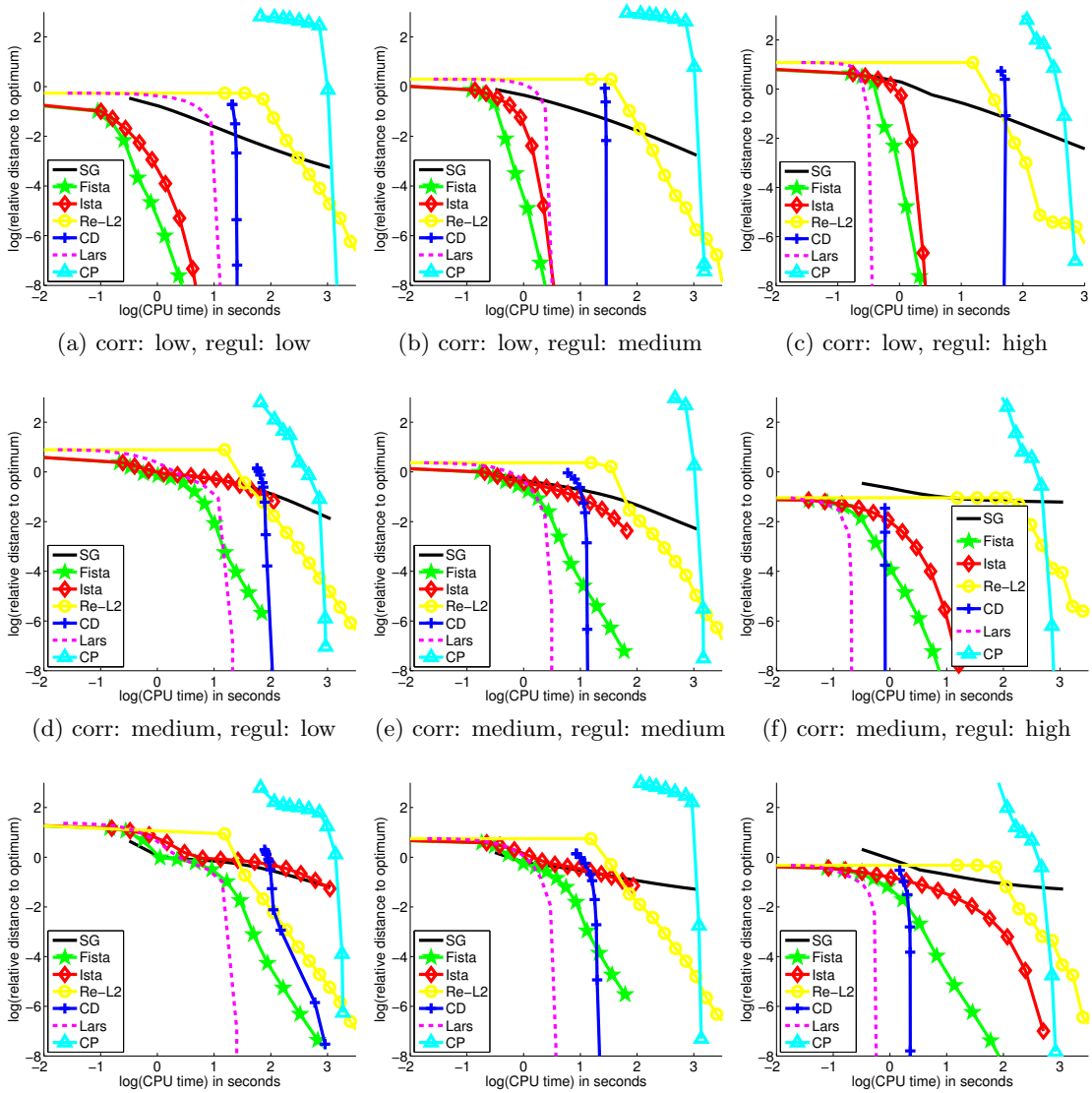



(a) corr: low, regul: low (b) corr: low, regul: medium (c) corr: low, regul: high

(d) corr: medium, regul: low (e) corr: medium, regul: medium (f) corr: medium, regul: high

(g) corr: high, regul: low (h) corr: high, regul: high (i) corr: high, regul: high

Figure 1.7: Benchmark for solving the Lasso for the small-scale experiment ($n = 200, p = 200$), for the three levels of correlation and three levels of regularization, and 8 optimization methods (see main text for details). The curves represent the relative value of the objective function as a function of the computational time in second on a $\log_{10} / \log_{10}$ scale.




(a) corr: low, regul: low (b) corr: low, regul: medium (c) corr: low, regul: high

(d) corr: medium, regul: low (e) corr: medium, regul: medium (f) corr: medium, regul: high

(g) corr: high, regul: low (h) corr: high, regul: high (i) corr: high, regul: high

Figure 1.8: Benchmark for solving the Lasso for the medium-scale experiment $n = 2000, p = 10000$, for the three levels of correlation and three levels of regularization, and 8 optimization methods (see main text for details). The curves represent the relative value of the objective function as a function of the computational time in second on a $\log_{10} / \log_{10}$ scale.

*selected dictionary elements*. It sequentially adds a different dictionary element to a set of active variables, which we denote by $\Lambda$, trying to solve Eq. (1.12) for every sparsity value $s' \leq s$, and stops when the desired sparsity is reached. In some sense, it builds a regularization path, and therefore shares similarities with LARS, even though the two algorithms address different optimization problems. These similarities are even stronger in terms of implementation. Similar tricks as those described in Section 1.4.2 for the LARS algorithm can be used, and in fact both algorithms have the same complexity, and have many steps in common. At each iteration, an active set $\Lambda$ containing the indices of the selected dictionary elements is obtained. A good criterion for choosing the next dictionary element is to select the one that helps most reducing the objective function

$$\hat{\imath} \leftarrow \arg\min_{i \in \Lambda^C} \min_{\boldsymbol{\beta} \in \mathbb{R}^{|\Lambda|+1}} \frac{1}{2}\|\mathbf{x} - \mathbf{D}_{\Lambda \cup \{i\}}\boldsymbol{\beta}\|_2^2.$$

This might seem computationally expensive since it requires solving $|\Lambda^C|$ least-squares problems, but the solution can in fact be obtained efficiently using some tricks, based on Cholesky decomposition and basic linear algebra, which we will not detail here for simplicity reasons. More details can be found in Cotter et al. (1999) or in the software SPAMS presented in Section D.1.

After this step, the active set is updated $\Lambda \leftarrow \Lambda \cup \{\hat{\imath}\}$, and the corresponding residual $\mathbf{r}$ and coefficients $\boldsymbol{\alpha}$ are

$$\begin{aligned}\boldsymbol{\alpha} &\leftarrow (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1}\mathbf{D}_\Lambda^\top \mathbf{x},\\ \mathbf{r} &\leftarrow (\mathbf{I}_p - \mathbf{D}_\Lambda(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1}\mathbf{D}_\Lambda^\top)\mathbf{x},\end{aligned}$$

where $\mathbf{r}$ is the residual of the orthogonal projection of $\mathbf{x}$ onto the linear subspace spanned by the columns of $\mathbf{D}_\Lambda$. It is worth noticing that one does not need to compute these two quantities in practice, but only updating the Cholesky decomposition of $(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1}$ and computing directly $\mathbf{D}^\top \mathbf{r}$, via simple linear algebra relations.

OMP naturally extends to the case of group sparsity, addressing the problem

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \quad \text{s.t.} \quad \#\{\|\boldsymbol{\alpha}_g\| \neq 0; g \in \mathcal{G}\} \leq s,$$

where $\mathcal{G}$ is a set of groups, and the number of active groups should be smaller than $s$. The optimization scheme in this setting is the same as OMP, except that one has to select groups instead of individual variables. The active set $\Lambda$ is now a subset of $\mathcal{G}$, and the criterion for choosing the next group $\hat{g}$ can be

$$\hat{g} \leftarrow \arg\min_{g \in \Lambda^C} \|\mathbf{D}^\top(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})\|_2.$$

This modified version of OMP was first proposed by Tropp et al. (2006) in the context of simultaneous sparse coding with a slightly different criterion, and revisited by Lozano et al. (2009). This version also admits an efficient Cholesky-based implementation, which we have used in Chapter 4 for image processing.

### 1.4.7 Difference of Convex (DC) Programming - Reweighted-$\ell_1$ Schemes

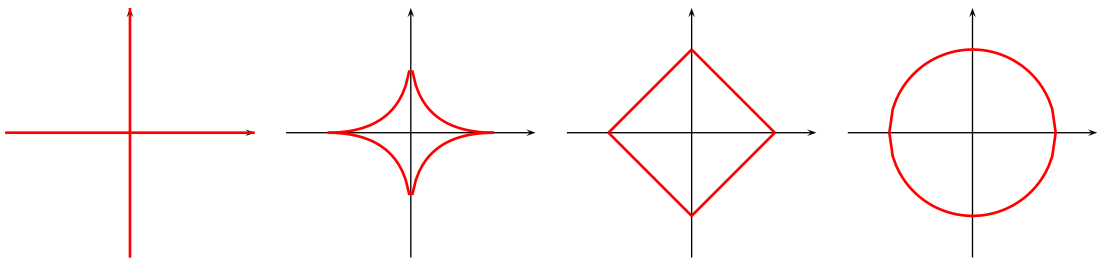
This section addresses the problem of solving

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \Big[ g(\boldsymbol{\alpha}) \triangleq f(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha}) \Big],$$

where $\Omega$ is a *non-convex* regularization function which is *separable* in every component of $\boldsymbol{\alpha}$—that is, there exists a function $\psi : \mathbb{R}^+ \to \mathbb{R}^+$ such that for all $\boldsymbol{\alpha}$ in $\mathbb{R}^p$, $\Omega(\boldsymbol{\alpha}) = \sum_{i=1}^p \psi(|\boldsymbol{\alpha}_i|)$, with $\psi$ *differentiable and concave*.

It is possible for instance to choose $\Omega(\boldsymbol{\alpha}) = \|\boldsymbol{\alpha}\|_q^q \triangleq \sum_{i=1}^p |\boldsymbol{\alpha}_i|^q$, where $\|.\|_q$ is an $\ell_q$-pseudo-norm with $0 < q < 1$. Another classical choice is also $\Omega(\boldsymbol{\alpha}) = \sum_{i=1}^p \log(|\boldsymbol{\alpha}_i| + \varepsilon)$ (see Candès et al., 2008).

The main motivation for using such approaches is to exploit a regularization function that induces more sparsity than the $\ell_1$-norm, and which might be addressed with other tools than greedy methods. The unit balls corresponding to the $\ell_q$ pseudo-norms and norms for several values of $q$ are illustrated in Figure 1.9. When $q$ decreases, the $\ell_q$-ball get "closer" to the $\ell_0$-ball, and better induces sparsity.



(a) $\ell_0$-ball, 2-D (b) $\ell_{0.5}$-ball, 2-D (c) $\ell_1$-ball, 2-D (d) $\ell_2$-ball, 2-D

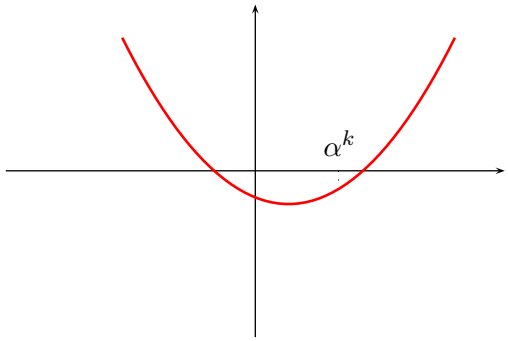
Figure 1.9: Open balls in 2-D corresponding to several $\ell_q$ norms and pseudo-norms.

Even though the corresponding optimization problem is not convex and still not smooth, a local optimum can be obtained using a DC-programming type of approach (see Gasso et al., 2009; Candès et al., 2008, and references therein). The idea behind such a scheme is relatively simple. It consists of iteratively minimizing convex surrogates $\tilde{g}_k$ of the cost function $g$ that are tangent to the graph of $g$ around the current estimate $\boldsymbol{\alpha}^k$. In other words, at iteration $k$, $\tilde{g}_k(\boldsymbol{\alpha}^k) = g(\boldsymbol{\alpha}^k)$ and $\tilde{g}_k(\boldsymbol{\alpha}) \geq g(\boldsymbol{\alpha})$ for all $\boldsymbol{\alpha}$. To obtain such surrogates, the key is to exploit the concavity of the functions $\psi$ on $\mathbb{R}^+$, which are always below their tangents. This is illustrated in Figure 1.10. It is then easy to show that such an iterative scheme can be written

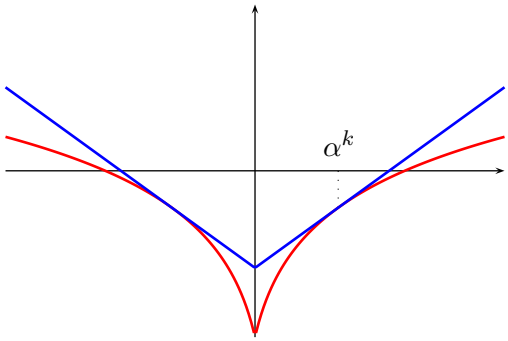
$$\boldsymbol{\alpha}^{k+1} \leftarrow \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \Big[ f(\boldsymbol{\alpha}) + \lambda \sum_{i=1}^p \psi'(|\boldsymbol{\alpha}_i^k|)|\boldsymbol{\alpha}_i| \Big],$$

which is a reweighted-$\ell_1$ decomposition problem. Note that with this scheme, the first step is usually a simple Lasso, with no weights. The effect of the new weights $\psi'(|\boldsymbol{\alpha}_i^k|)$ is

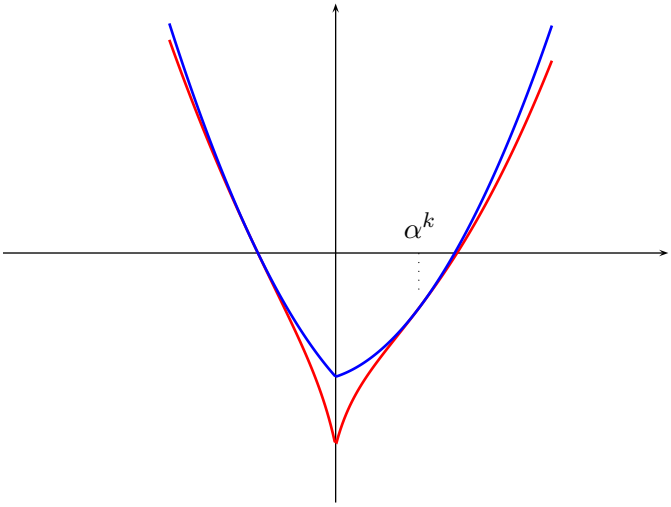
to push to zero the smallest non-zero coefficients returned by the Lasso, and in practice two or three iterations are enough to obtain the desired sparsifying effect.




(a) red: $f(\alpha)$.




(b) red: $\psi(\alpha) = \log(|\alpha| + \varepsilon)$.
blue: convex surrogate $\psi'(\alpha^k)|\alpha| + C$.




(c) red: $g(\alpha) = f(\alpha) + \psi(\alpha)$.
blue: convex surrogate $\tilde{g}_k(\alpha) = f(\alpha) + \psi'(\alpha^k)|\alpha| + C$.

Figure 1.10: Illustration of the DC-programming approach. The non-convex part of the function $g$ is upperbounded by a convex weighted $\ell_1$-norm. The graphs of $g$ and its surrogate $\tilde{g}_k$ are tangent.

### 1.4.8 Network Flow Optimization

We present in this section some elements of network flow optimization (see Bertsekas, 1991; Ahuja et al., 1993, and references therein for more details), and its connections with sparse methods, which we further exploit in Chapter 3.

Let us consider a directed graph $G = (V, E, s, t)$, where $V$ is a set of vertices, $E \subseteq V \times V$ an arc set, $s$ is a vertex called *source*, and $t$ is a vertex called *sink*, such that there is no arc directed to $s$, and no arc outcoming from $t$.

We define a non-negative *capacity* function $c : E \to \mathbb{R}^+$ on the arcs. A *flow* $f : E \to \mathbb{R}^+$ is a non-negative function on arcs that satisfies capacity constraints on all arcs (the value of the flow on an arc is less than or equal to the arc capacity) and conservation constraints on all vertices (the sum of incoming flows at a vertex is equal to the sum of outgoing flows) except for the source and the sink.[10]

To simplify the notations, we can arbitrarily order the vertices and identify $V \setminus \{s, t\}$ with a set $[\![1; p]\!]$, where $p \triangleq |V| - 2$, so that an arc in $E$ can be identified by two indices $(i, j)$. Denoting respectively $c_{ij}$ and $f_{ij}$ the capacity and the flow on an arc $(i, j)$ in $E$, we can write the capacity constraints as

$$\forall (i,j) \in E, \quad f_{ij} \leq c_{ij},$$

and the conservation constraints

$$\forall i \in V, \quad \underbrace{\sum_{j;(i,j)\in E} f_{ij}}_{\text{outgoing flow from } i} = \underbrace{\sum_{j;(j,i)\in E} f_{ji}}_{\text{incoming flow to } i}$$

The value of the flow is the amount of flow outgoing from $s$, $\sum_{i \in V;(s,i)\in E} f_{si}$ which is equal to the flow incoming to $t$, $\sum_{i \in V;(i,t)\in E} f_{it}$. This is illustrated in Figure 1.11

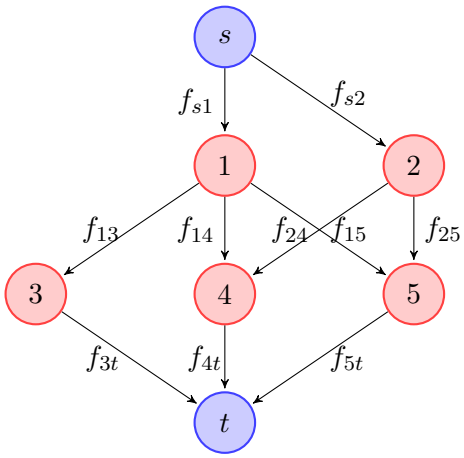



Figure 1.11: Example of a directed graph $G = (V, E, s, t)$, with flows $f_{ij}$, $(i, j) \in E$. The flow should respect the capacity constraint $f_{ij} \leq c_{ij}$ for all $(i, j)$ in $E$.

[10]Note that we only consider here the case of real-valued functions, since this is the one we need in this thesis. Network flow problems with integer-valued functions can also be considered, and in fact many results that are true in the continuous settings are also true in the discrete one.

A classical problem in network flow optimization is the *max-flow* problem (Ford and Fulkerson, 1956), which consists of finding a flow of maximum value in the graph.

We can also define an $(s,t)$-cut in the graph, which is a partition $(S,T)$ of $V$ with $s$ in $S$ and $t$ in $T$. It is possible to define the capacity of the cut $\sum_{(i,j)\in E;i\in S;j\in T} c_{ij}$, and the problem *min-cut* consists of finding a cut in the graph with minimum capacity. With these tools in hand, we can now move to the first interesting result due to Ford and Fulkerson (1956):

**Proposition 3 (Max-flow / min-cut theorem)**
*The maximum value of a flow in a graph is equal to the minimum capacity cut.*

This proposition is not yet related to sparse methods. It just presents a duality relation between the max-flow and min-cut problems. We will intensively use it in Chapter 3. For solving the max-flow problem, a popular algorithm called "push-relabel" is due to Goldberg and Tarjan (1986). We have implemented it in the software presented in Section D.2, using refinements presented by Cherkassky and Goldberg (1997). An example of a cut in a graph is presented in Figure 1.12 as long as a few properties of min $(s,t)$-cuts, namely:

- There is no flow going from $T$ to $S$ (see Bertsekas, 1991).
- The cut goes through all arcs going from $S$ to $t$, and all arcs going from $s$ to $T$, and such arcs are saturated (the value of the flow on the arc equals the capacity).

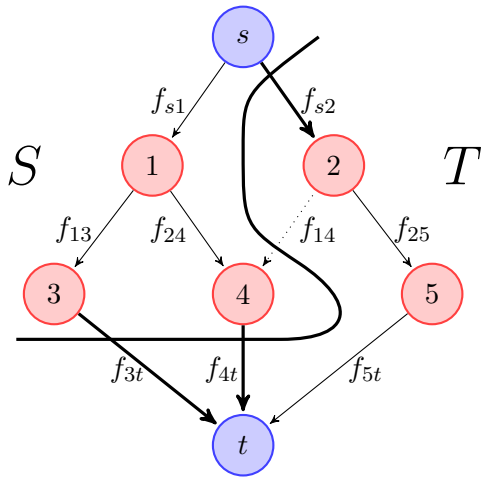



Figure 1.12: Example of a cut in a graph. Arcs in bold are saturated (the value of the flow equals the capacity) and the flow on dotted arcs is zero.

We will also consider in this thesis the class of *min-cost* flow problems, which we now present. In addition to the capacity function, let us define cost functions $C_{ij} : \mathbb{R} \to \mathbb{R}$, one for every arc $(i,j)$ in $E$. The min-cost flow problem consists of finding a flow $f$

that minimizes the total cost on the graph $\sum_{(i,j)\in E} C_{ij}(f_{ij})$. The costs $C_{ij}$ are often linear in the flow $f_{ij}$, and in fact the terminology "min-cost flow problem" often refers to this particular setting in the literature. The more interesting case because of its connection with sparse methods is that of quadratic cost functions. In particular, we show in Chapter 3 that the problem of projecting a vector onto the simplex, which can be written as follows

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{u}-\boldsymbol{\alpha}\|_2^2 \quad \text{s.t.} \quad \sum_{i=1}^{p} \boldsymbol{\alpha}_i = 1, \quad \boldsymbol{\alpha} \geq 0,$$

is a particular cost of a quadratic min-cost flow problem, also known as *continuous quadratic knapsack problem*. This has been addressed with linear-time algorithms by Brucker (1984), revisited later by Maculan and de Paula (1989), and rediscovered recently in the machine learning community by Duchi et al. (2008). We further explore these connections between network flow algorithms and sparse methods in Chapter 3.

## 1.5 Dictionary Learning and Matrix Factorization

We have presented in the previous section several tools to solve sparse decomposition problems when the dictionary is fixed. We now move to the dictionary learning framework.

The problem of learning a basis set, first introduced by Olshausen and Field (1996, 1997), can be formulated as a matrix factorization problem. More specifically, given a training set of $n$ signals $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ in $\mathbb{R}^{m\times n}$, one looks for a dictionary matrix $\mathbf{D}$ in $\mathbb{R}^{m\times p}$ such that each signal $\mathbf{x}^i$ admits a sparse decomposition in $\mathbf{D}$. This can be written in a general form

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{A}\in\mathcal{A}} \frac{1}{n}\sum_{i=1}^{n}\Big[\frac{1}{2}\|\mathbf{x}^i-\mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}^i)\Big],$$

where $\mathcal{D}$ and $\mathcal{A}$ are convex sets, $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n]$ is in $\mathcal{A} \subseteq \mathbb{R}^{p\times n}$ and $\Omega$ is a sparsity-inducing regularization term. The number of samples $n$ is usually large, whereas the signal dimension $m$ is relatively small, for example, $m = 100$ for $10 \times 10$ image patches, and $n \geq 100\,000$ for typical image processing applications. In general, we also have $p \ll n$ (e.g., $p = 200$ for $n = 100\,000$), but each signal only uses a few elements of $\mathbf{D}$ in its representation, say 10 for instance. Note that, in this setting, overcomplete dictionaries with $p > m$ are allowed.

This problem can equivalently be rewritten as a matrix factorization problem:

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{A}\in\mathcal{A}} \Big[\frac{1}{2}\|\mathbf{X}-\mathbf{D}\mathbf{A}\|_{\mathrm{F}}^2 + \lambda\Omega'(\mathbf{A})\Big],$$

where $\Omega'(\mathbf{A}) \triangleq \frac{1}{n}\sum_{i=1}^{n}\Omega(\boldsymbol{\alpha}^i)$. A classical choice consists for instance in choosing $\Omega$ to be the $\ell_1$-norm, $\mathcal{A}$ to be unconstrained ($\mathcal{A} = \mathbb{R}^{p\times n}$), and $\mathcal{D}$ to be the set of matrices whose columns have bounded $\ell_2$-norms:

$$\mathcal{D} \triangleq \{\mathbf{D} \in \mathbb{R}^{m\times p} \quad \text{s.t.} \quad \forall j \in [\![1;p]\!], \quad \|\mathbf{d}^j\|_2^2 \leq 1\}.$$

Since the term $\mathbf{DA}$ is invariant by multiplying $\mathbf{D}$ by a diagonal matrix on the right and $\mathbf{A}$ by its inverse on the left, preventing $\mathbf{D}$ from being arbitrarily large (which would make $\mathbf{A}$ arbitrarily small) has indeed proven to be necessary in practice.

We now present other matrix factorization formulations that are related to dictionary learning.

### 1.5.1 Classical Matrix Factorization Methods

We start by principal component analysis (PCA), then move to vector quantization and non-negative matrix factorization.

#### Principal Component Analysis

Principal component analysis is a widely used tool for data analysis. It looks for a set of orthogonal directions that maximize the explained variance of data vectors. This is equivalent to the matrix factorization problem

$$\min_{\mathbf{D}\in\mathbb{R}^{m\times p},\mathbf{A}\in\mathbb{R}^{p\times n}} \frac{1}{2}\|\mathbf{X}-\mathbf{DA}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \mathbf{D}^\top\mathbf{D}=\mathbf{I}_m \quad \text{and} \quad \mathbf{AA}^\top \text{ is diagonal.}$$

The solution can be obtained by a singular value decomposition (SVD), and the columns on $\mathbf{D}$ are the desired *principal components*.

#### Vector Quantization - Hard Assignment

Vector quantization (or clustering) can also be seen as a matrix factorization problem. Given $n$ data vectors $\mathbf{X}=[\mathbf{x}^1,\dots,\mathbf{x}^n]$, one can look for $p$ clusters, defined by their centroids $[\mathbf{d}^1,\dots,\mathbf{d}^p]$ and a binary assignment for each data vector, which can be represented by binary vectors $\boldsymbol{\alpha}^i$ in $\{0,1\}^p$ such that $\sum_{j=1}^p \boldsymbol{\alpha}_j^i = 1$—that is, one single entry of $\boldsymbol{\alpha}^i$ is equal to 1, and the rest is zero. Since the assignments have binary values, one often uses the terminology "clustering with hard assignment", as opposed to "soft assignment", which is the topic of the next section.

With these notations in hand, we rewrite the clustering problem as

$$\min_{\mathbf{D}\in\mathbb{R}^{m\times p},\mathbf{A}\in\{0,1\}^{p\times n}} \frac{1}{2}\|\mathbf{X}-\mathbf{DA}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \sum_{j=1}^{p}\boldsymbol{\alpha}_j^i = 1, \text{ for all } i\in[\![1;p]\!],$$

which is the same optimization problem addressed by the algorithm K-means (see Hastie et al., 2009, and references therein). It can be seen as a specific sparse matrix factorization, where the columns of $\mathbf{A}$ are forced to have a sparsity of one. And in fact, the algorithm K-SVD introduced by Aharon et al. (2006) for learning dictionaries, is presented by their authors as a generalization of K-means, emphasizing the tight links between clustering and dictionary learning.

### Vector Quantization - Soft Assignment

One possible view of vector quantization with soft assignment is to model data vectors as non-negative linear combinations of centroids that sum to one. More precisely, the corresponding optimization problem is

$$\min_{\mathbf{D}\in\mathbb{R}^{m\times p},\mathbf{A}\in\mathbb{R}^{p\times n}} \frac{1}{2}\|\mathbf{X}-\mathbf{D}\mathbf{A}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \sum_{j=1}^{p} \boldsymbol{\alpha}_j^i = 1, \text{ for all } i \in [\![1;p]\!], \text{ and } \mathbf{A} \geq 0,$$

which is even closer to dictionary learning than vector quantization with hard assignment. Interestingly, these connections have been recently further exploited in computer vision by Yang et al. (2009); Boureau et al. (2010) in the so-called *bags-of-features* models, using dictionary learning instead of classical vector quantization techniques for building visual codebooks that are used for image classification tasks.

### Non-negative Matrix Factorization

We now mention the non-negative matrix factorization technique proposed by Lee and Seung (2001). In its simplest form, it consists of solving

$$\min_{\mathbf{D}\in\mathbb{R}^{m\times p},\mathbf{A}\in\mathbb{R}^{p\times n}} \frac{1}{2}\|\mathbf{X}-\mathbf{D}\mathbf{A}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \mathbf{D} \geq 0 \text{ and } \mathbf{A} \geq 0,$$

With this formulation, the matrices $\mathbf{D}$ and $\mathbf{A}$ are forced to have non negative entries, which can lead to sparse solutions. When applied to images, such as faces, Lee and Seung (2001) have shown that the learned features are more "localized" than the ones learned with a classical singular value decomposition. Whereas the importance of NMF in computer vision remain unclear, it has led to interesting results for audio analysis (Févotte et al., 2009), but with a different loss function than the square loss that is more adapted to audio modalities. Variants of NMF with sparsity constraints (Hoyer, 2002, 2004) have also been proposed, with strong connections with dictionary learning.

## 1.5.2 Dictionary Learning Algorithms

We now move to one of the main topics of this thesis, which is dictionary learning. Like all the matrix factorization formulations presented in the previous section, the corresponding optimization problems are non-convex, and we classify them into two categories. Those relying on $\ell_1$-regularization, and those exploiting directly the $\ell_0$-pseudo-norm.

### Matrix Factorization with $\ell_1$-regularization

We start by considering the $\ell_1$-regularized dictionary learning problem, defined as

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{A}\in\mathbb{R}^{m\times p}} \frac{1}{n}\sum_{i=1}^{n}\Big[\frac{1}{2}\|\mathbf{x}^i-\mathbf{D}\boldsymbol{\alpha}^i\|_2^2+\lambda\|\boldsymbol{\alpha}^i\|_1\Big],$$

which was first considered by Olshausen and Field (1996), along with other non-convex smooth regularizers that induce approximately sparse vectors (vectors that are not sparse but that have many small coefficients).

Even though the optimization problem is not jointly convex in $(\mathbf{D}, \mathbf{A})$, it is convex with respect to each variable $\mathbf{D}$ or $\mathbf{A}$ when the other one is fixed. A natural way of optimizing the cost function is therefore to alternate the minimization between $\mathbf{D}$ and $\mathbf{A}$, fixing one and optimizing with respect to the other. Optimizing with respect to $\mathbf{A}$ can be done with any technique we have presented so far in this thesis, even though we might prefer LARS for the classical setting where the $\mathbf{x}^i$'s are relatively small and the solution very sparse. As shown in our benchmark presented in Section 1.4.5, LARS is indeed particularly efficient in this case. Optimizing with respect to $\mathbf{D}$ can be done with a gradient descent approach as done by Olshausen and Field (1996), or a Newton method in a dual formulation, as proposed by Lee and Seung (2001). We propose in Chapter 2 other approaches to address this problem, which have proven to be more efficient.

### Matrix Factorization with $\ell_0$-regularization

The $\ell_0$ dictionary learning formulation can be written as follows

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{A}\in\mathbb{R}^{m\times p}} \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\alpha}^i\|_0 \leq s, \ \forall i \in [\![1;n]\!]. \tag{1.13}$$

The approach proposed by Engan et al. (1999) and called MOD (method of optimal directions) is also an alternate minimization approach.

- During the *sparse coding step* $\mathbf{D}$ is fixed, and the vectors $\boldsymbol{\alpha}^i$ are obtained using a greedy approach, such as the ones presented in Section 1.4.6.
- During the *dictionary learning* step, $\mathbf{A}$ is fixed and $\mathbf{D}$ is updated with the formula

  $$\mathbf{D} \leftarrow \Pi_{\mathcal{D}}[\mathbf{X}\mathbf{A}(\mathbf{A}\mathbf{A}^\top)^{-1}],$$

  where $\mathbf{X}\mathbf{A}(\mathbf{A}\mathbf{A}^\top)^{-1}$ is the solution of the minimization of Eq. (1.13) with respect to $\mathbf{D}$ when the coefficients $\mathbf{A}$ are fixed and the constraints $\mathcal{D}$ are dropped. $\Pi_{\mathcal{D}}$ is the projection operator on $\mathcal{D}$, that in practice normalizes the columns of a given matrix. Since the cost function of Eq. (1.13) is invariant by replacing $\mathbf{D}$ by $\mathbf{D}\boldsymbol{\Gamma}$ and $\mathbf{A}$ by $\boldsymbol{\Gamma}^{-1}\mathbf{A}$ where $\boldsymbol{\Gamma}$ is a positive definite diagonal matrix, it is possible to show that such an update minimizes Eq. (1.13) with respect to $\mathbf{D}$, when in addition one authorizes the rows of $\mathbf{A}$ to be rescaled.

The K-SVD is another approach proposed by Aharon et al. (2006). It is also an alternate minimization approach between two steps. The sparse coding step is the same as for MOD, but the dictionary update step updates both $\mathbf{D}$ and the values of the non-zero coefficients of $\mathbf{A}$. The dictionary learning step consists of one pass of a block-coordinate approach, where sequentially for all $j$ in $[\![1;p]\!]$, *one column* $\mathbf{d}^j$ is updated

(keeping the other fixed) simultaneously with the non-zero entries of *the $j$-th row* of $\mathbf{A}$. Such an update can equivalently be rewritten as a one-rank approximation of a matrix, which can be obtained with a one-rank singular value decomposition (SVD), giving its name to the algorithm.

## 1.6 Dictionary Learning for Image Processing

Some of the results presented in this section are reported from the following works, which have been undertaken before the beginning of this PhD:

J. Mairal, M. Elad and G. Sapiro. Sparse representation for color image restoration. IEEE Transactions on Image Processing. 17(1):53–69. 2008.

J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. SIAM Multiscale Modelling and Simulation, 7(1):214–241, April 2008.

J. Mairal, G. Sapiro, and M. Elad. Multiscale sparse image representation with learned dictionaries. Proceedings of the IEEE International Conference on Image Processing, 2007.

We show in this section several applications of the dictionary learning problem, whose successes have motivated our research.

### 1.6.1 Dictionary Learning for Natural Image Patches

Before moving to concrete applications, we show the result of learning dictionaries on a database of natural images patches, as originally proposed by Olshausen and Field (1996). To do so, we use the algorithm which will be presented in Chapter 2 on a database of 10 millions patches of size $12 \times 12$ pixels, randomly extracted from natural images. We process both grayscale image patches, and RGB color image patches that are concatenated as a single vector as done by Mairal et al. (2008b). For grayscale patches, the mean value of each patch is removed and the patches are normalized to have unit $\ell_2$-norm. For the color patches, the mean color of the patch is removed and the patches are also normalized. We show in Figure 1.13 visual results obtained when learning $p = 256$ dictionary elements, using the $\ell_1$-regularized version of dictionary learning, with a parameter $\lambda = 0.1$. As already reported in the literature (see Olshausen and Field, 1996; Elad and Aharon, 2006), we observe intriguing results: some of the dictionary elements looking like oriented edges (somewhat similar to Gabor filters), some others look like low-pass filters. As for the color patches, we observe an interesting phenomenon, namely that most of the dictionary elements look gray, and might therefore be "devoted", to reconstructing geometrical structures in images. As for the colored

dictionary elements, they typically have low frequencies and seem, to some extent, to present two opposite colors, with groups of "green-magenta" or "yellow-blue" dictionary elements. A similar observation was also made earlier by Hoyer and Hyvärinen (2000) with a different technique called independent component analysis (ICA), which is used to model data vectors as linear mixtures of independent latent variables. When applied to natural image patches, this technique visually leads to similar results as dictionary learning.

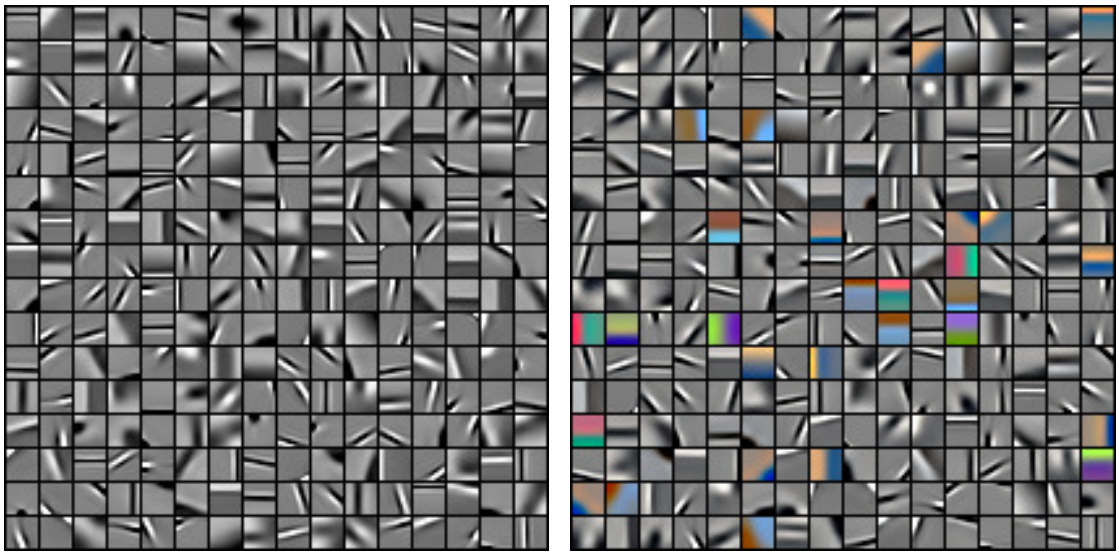


Figure 1.13: Example of a learned dictionaries on $12 \times 12$ patches of natural images with $p = 256$ dictionary elements. Left: dictionary learned on grayscale image patches. Right: dictionary learned on RGB color image patches. The dictionary is learned using the algorithm of Chapter 2 on a database of 10 millions patches. Since patches may have negative values, they are arbitrarily translated and rescaled for display.

We now move to restoration tasks exploiting this image patch representation, which have been quite successful.

### 1.6.2 Image Denoising

We present in this section a successful denoising method first introduced by Elad and Aharon (2006). Let us consider first the classical problem of restoring a noisy image $\mathbf{y}$ in $\mathbb{R}^n$ which has been corrupted by a white Gaussian noise of standard deviation $\sigma$.

Classical techniques often formulate the image denoising problem as an energy minimization one, trying to find an estimate $\hat{\mathbf{x}}$ that minimizes

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 + \psi(\mathbf{x}),$$

where the first quadratic term is called a *data fitting term*, ensuring that the estimate is close enough to the noisy measurement $\mathbf{y}$, and $\psi(\mathbf{x})$ is a regularization function ensuring that the estimate $\mathbf{x}$ respects a particular image model.[11]

Finding a good image model is a notably difficult task. Early works have assumed the image to be smooth using filtering techniques (Kovasznay and Joseph, 1955; Perona and Malik, 1990), to have a small total variation (Rudin and Osher, 1994), or have used Markov Random Fields (MRF) to model regularity between adjacent pixels (Zhu and Mumford, 1997). We now use the assumption that the clean signal can be approximated by a *sparse* linear combination of elements from a dictionary. Like many recent works (Buades et al., 2005; Roth and Black, 2005) the approach we present is patch-based.

Under this assumption, denoising a patch $\mathbf{y}^i$ in $\mathbb{R}^m$ with a dictionary $\mathbf{D}$ in $\mathbb{R}^{m\times p}$ (with $p$ elements), amounts to solving the following sparse decomposition problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \Omega(\boldsymbol{\alpha}) \quad \text{s.t.} \quad \|\mathbf{y}^i - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \leq \varepsilon, \tag{1.14}$$

where $\mathbf{D}\boldsymbol{\alpha}$ is an estimate of the clean signal, and $\Omega$ is a sparsity-inducing regularization function. It can be the $\ell_1$-norm, leading to the well-known Lasso (Tibshirani, 1996) and basis pursuit (Chen et al., 1998) problems, or the $\ell_0$-pseudo-norm. Following Elad and Aharon (2006); Mairal et al. (2008b, 2009c), $\varepsilon$ can be chosen according to the (supposed known) standard deviation $\sigma$ of the noise. One indeed expects the residual $\mathbf{y}^i - \mathbf{D}\boldsymbol{\alpha}$ to behave as a Gaussian vector, and thus $\|\mathbf{y}^i - \mathbf{D}\boldsymbol{\alpha}\|_2^2/\sigma^2$ to follow a chi-squared distribution $\chi^2_m$ concentrated around $m$. The strategy proposed by Mairal et al. (2008b) is to put a threshold the cumulative distribution function $F_m$ of the $\chi^2_m$ distribution and choose $\varepsilon$ as $\varepsilon = \sigma^2 F_m^{-1}(\tau)$, where $F_m^{-1}$ is the inverse of $F_m$. Selecting the value $\tau = 0.9$ leads in practice to acceptable values of $\varepsilon$ (Mairal et al., 2008b, 2009c).

Various types of wavelets (Mallat, 1999) have been used as dictionaries for natural images. Building on ideas proposed by Olshausen and Field (1997) to model neuronal responses in the V1 area of the brain, Elad and Aharon (2006) have proposed instead to *learn* a dictionary $\mathbf{D}$ adapted to the image at hand, and demonstrated that learned dictionaries lead to better empirical performance than off-the-shelf ones. Since images may be very large, efficiency concerns naturally lead to sparsely decomposing image patches rather than the full image. For an image of size $n$, a dictionary in $\mathbb{R}^{m\times p}$ adapted to the $n$ overlapping patches of size $m$ (typically $m = 8\times 8 \ll n$) associated with the image pixels, is learned by addressing the following optimization problem

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{A}\in\mathbb{R}^{p\times n}} \sum_{i=1}^{n} \Omega(\boldsymbol{\alpha}^i) \quad \text{s.t.} \quad \|\mathbf{y}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 \leq \varepsilon, \tag{1.15}$$

where $\mathcal{D}$ is the set of matrices in $\mathbb{R}^{m\times p}$ with unit $\ell_2$-norm columns, $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n]$ is a matrix in $\mathbb{R}^{p\times n}$, $\mathbf{y}^i$ is the $i$-th patch of the *noisy* image $\mathbf{y}$, $\boldsymbol{\alpha}^i$ is the corresponding

[11] In a probabilistic model, the optimization problem would be written $\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2\sigma}\|\mathbf{y}-\mathbf{x}\|_2^2 - \log p(\mathbf{x})$, where $p$ is a prior distribution for $\mathbf{x}$. Therefore $\psi$ can be related to a log-prior.

code, and $\mathbf{D}\boldsymbol{\alpha}^i$ is the estimate of the denoised patch. Note that this procedure implicitly assumes that the patches are *independent* from each other, which is questionable since they overlap. However, this approximation makes the corresponding optimization tractable. Adding some consistency in the reconstruction of adjacent patches, instead of processing them independently is in fact an interesting open topic, which, to the best of our knowledge, has never been addressed effectively.

Once the dictionary $\mathbf{D}$ and codes $\boldsymbol{\alpha}^i$ have been learned, every pixel admits $m$ estimates (one per patch containing it), and its value can be computed by averaging these:

$$\mathbf{x} = \frac{1}{m} \sum_{i=1}^{n} \mathbf{R}^i \mathbf{D}\boldsymbol{\alpha}^i, \tag{1.16}$$

where $\mathbf{R}^i$ in $\mathbb{R}^{n \times m}$ is the binary matrix which places patch number $i$ at its proper position in the image. This approach learns the dictionary on the set of overlapping noisy patches, thereby adapting the dictionary to the image itself, which is a key element in obtaining better results. Such an aggregation procedure averaging estimators obtained by applying a non-translation-invariant operation on different shifted versions of patches, is related to the classical translation-invariant denoising proposed by Coifman and Donoho (1995), which basically proceeds in the same way with wavelet denoising. Even though aggregating estimators by straight averaging might look suboptimal, we are not aware of any other technique, in the context of dictionary learning, leading to better results for reconstructing the final image from the estimated patches.

How to choose between the $\ell_1$- or $\ell_0$-regularizations is not a priori clear. Following Elad and Aharon (2006), we have experimentally observed that, given a fixed dictionary $\mathbf{D}$, the reconstructed image is in general of better quality when using the $\ell_0$-pseudo-norm rather than its convex $\ell_1$ counterpart. However, we have also observed that dictionaries learned with the $\ell_1$-norm are usually better for denoising, even when the final reconstruction is done with the $\ell_0$-pseudo-norm. We investigate this question more thoroughly in Section 1.6.5.

### 1.6.3 Dictionary Learning with Missing Data — Inpainting

It is possible to model the presence of missing data in the dictionary learning formulation (see Mairal et al., 2008b). For a patch $i$ in $[\![1; n]\!]$, we introduce a binary mask $\mathbf{M}^i$ as a diagonal matrix in $\mathbb{R}^{m \times m}$ whose value on the $j$-th entry of the diagonal is 1 if the pixel $\mathbf{y}^i_j$ is known and 0 otherwise, where $\mathbf{y}^i_j$ is the $j$-th pixel of the $i$-th patch of an image $\mathbf{y}$ in $\mathbb{R}^n$. The general dictionary learning formulation with missing data then becomes

$$\min_{\mathbf{D} \in \mathcal{D}, \mathbf{A} \in \mathbb{R}^{p \times n}} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \|\mathbf{M}^i(\mathbf{y}^i - \mathbf{D}\boldsymbol{\alpha}^i)\|_2^2 + \lambda \Omega(\boldsymbol{\alpha}^i).$$

In practice, the presence of the binary mask does not drastically change the optimization procedure, and one still can alternates between the optimization of $\mathbf{D}$ and $\mathbf{A}$. When the image $\mathbf{y}$ is only corrupted by missing pixels and not by other additive noise, one can

also enforce hard reconstruction constraints, and address

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{A}\in\mathcal{A}} \Omega(\boldsymbol{\alpha}^i) \quad \text{s.t.} \quad \mathbf{M}^i(\mathbf{y}^i - \mathbf{D}\boldsymbol{\alpha}^i) = 0.$$

Before showing any inpainting result, we shall comment on *when these formulations are supposed to work*.

- First, the formulation exploits independently for each patch the available pixel values. It can therefore *only handle holes* that are smaller than the patch size. Handling large holes might be possible with a different formulation, for instance with a diffusion process that would allow filling in holes (see Roth and Black, 2005, for such a strategy).

- Second, one assumes that *the noise pattern does not admit a sparse representation*, which the dictionary could learn otherwise. The demosaicking task (see Mairal et al., 2008b) is a typical example of inpainting small holes with such a problematic pattern. In this case, different strategies can be used, such as learning the dictionary offline on a database of clean signals, and then possibly refine it on an estimate of the demosaicked image.

We now show inpainting results in Figure 1.14, one from Mairal et al. (2008b), and one from Mairal et al. (2008d), where a multiscale variant of K-SVD is introduced.

### 1.6.4 Video Processing

The extension of dictionary learning techniques for dealing with videos has been proposed by Protter and Elad (2009). Given a noisy video sequence, a first naive approach consists of processing each frame independently. To exploit temporal consistency and improve the performance of this approach, some key components can be added:

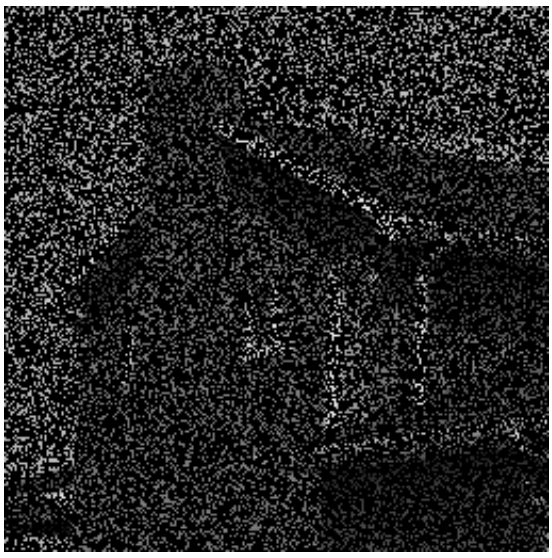
- One should process several frames at the same time, for instance $T$ frames, and consider video patches corresponding to 3-D blocks of size $m = e \times e \times T$ in the video, where $e$ is the edge size of a patch. Typical sizes might be for instance $e = 10$ pixels and $T = 5$ frames.

- After processing $T$ frames, one can move to the next block of $T$ frames (which possibly overlaps with the previous one), and one should use the previously learned dictionary as an initialization of the learning process that adapts the dictionary to the current block.

We show examples in Figures 1.15 and 1.16 two video processing results from (Mairal et al., 2008d), where this video extension has been adapted to the inpainting and color video denoising tasks.


(a) Example A, Damaged

(b) Example A, Restored

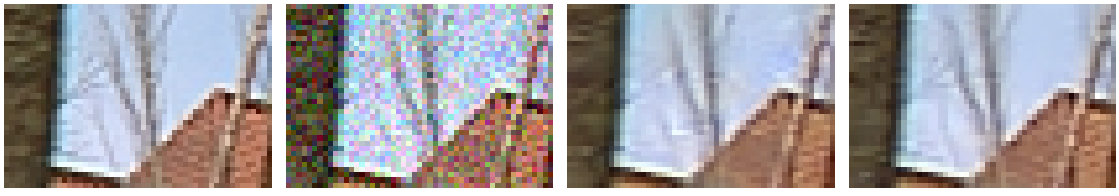


(c) Example B, Damaged

(d) Example B, Restored

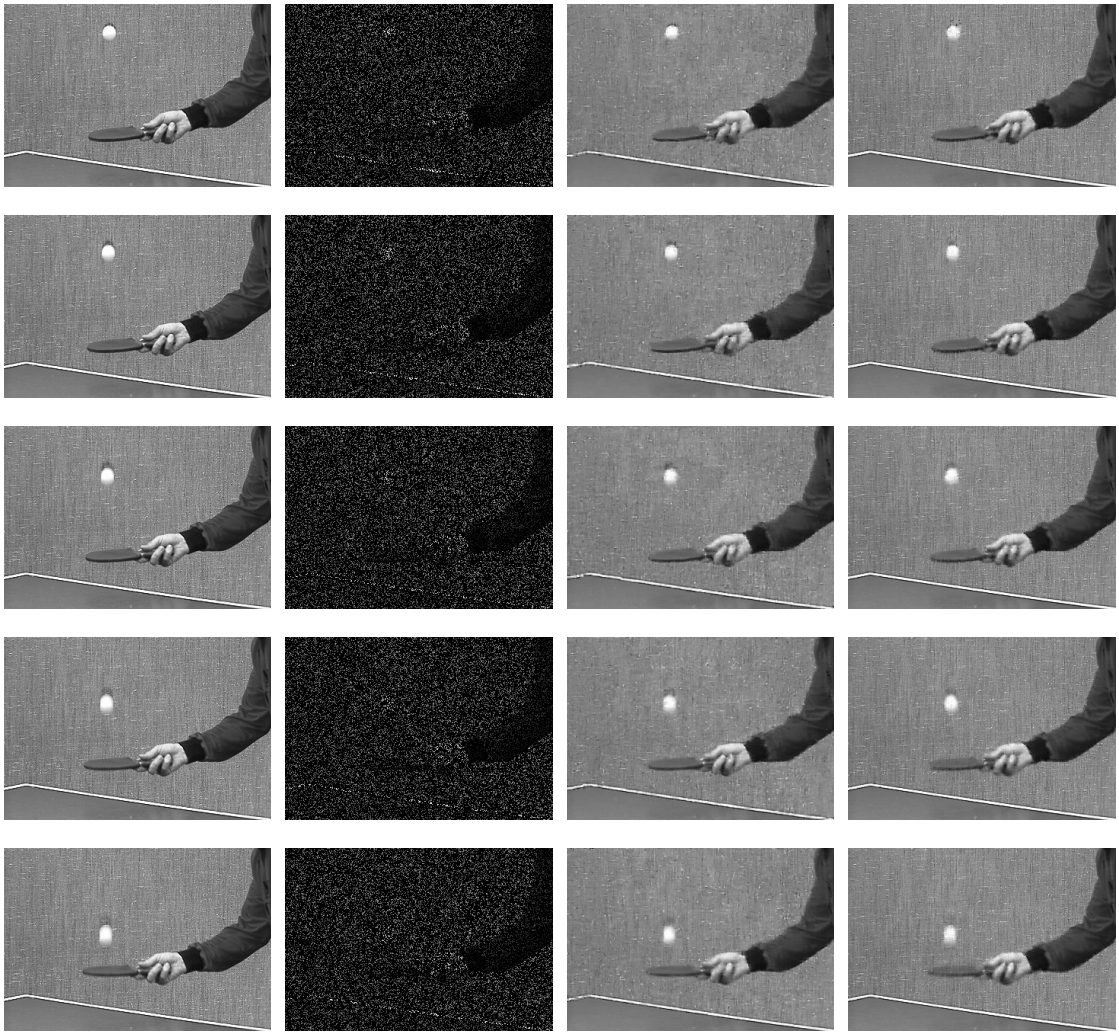
Figure 1.14: Top: Inpainting result from Mairal et al. (2008b), where the text is automatically removed on the restored image. Images are under copyright ©IEEE. Bottom: Inpainting result presented from Mairal et al. (2008d), where 80% of the pixels are randomly removed from the original image. The algorithm is able to reconstruct the brick texture on the right, without seeing the original image. Images under copyright ©SIAM.

(a) Original (b) Damaged (c) Image Denoising (d) Video Denoising

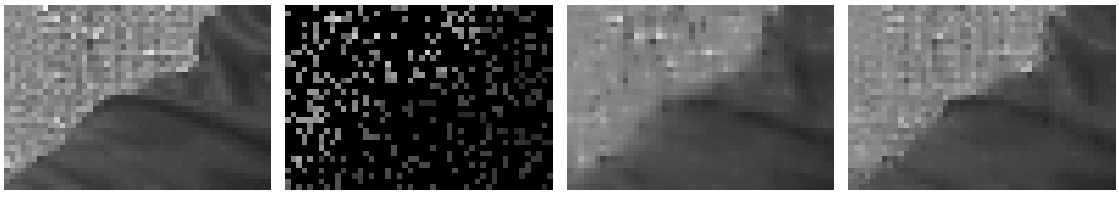


(e) Zoom on (a) (f) Zoom on (b) (g) Zoom on (c) (h) Zoom on (d)

Figure 1.15: Color video denoising result from Mairal et al. (2008d). The third column show the result when each frame is processed independently from the others. Last column show the result of the video processing approach. Images under copyright ©SIAM.



(a) Original (b) Damaged (c) Image Inpainting (d) Video Inpainting



(e) Zoom on (a) (f) Zoom on (b) (g) Zoom on (c) (h) Zoom on (d)

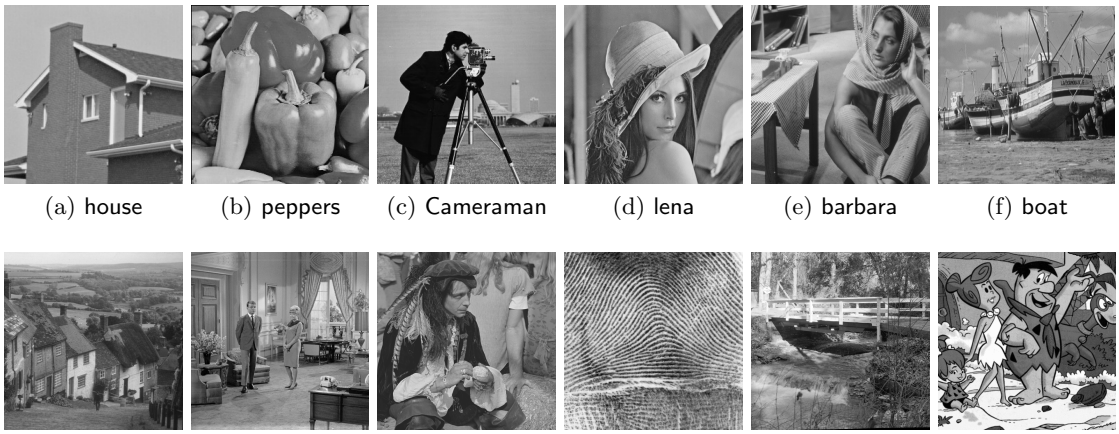
Figure 1.16: Video inpainting result from Mairal et al. (2008d). The third column show the result when each frame is processed independently from the others. Last column show the result of the video processing approach. Images under copyright ©SIAM.

### 1.6.5 $\ell_0$ vs $\ell_1$ for Image Denoising

In this section, we address the question of whether one should use the $\ell_0$ or the $\ell_1$-regularization for restoring natural images. We use the following methodology for processing one image, which follows from Elad and Aharon (2006), but allows using a different regularization scheme for learning the dictionary than for reconstructing the image:

1. **Patch Extraction**: Extract all overlapping patches from the image.
2. **Dictionary Learning Step**: Learn a dictionary on this set of patches using a regularization scheme **(A)**. We use the alternate minimization approach described before, with 50 iterations between updates of the coefficients and updates of the dictionary, after initializing the dictionary with randomly extracted patches from the image.
3. **Final Reconstruction Step**: Reconstruct every patch of the image using a regularization scheme **(B)**.
4. **Averaging Step**: Reconstruct the image using the averaging formula of Eq. (1.16).

For the quantitative evaluation, we have chosen a dataset of 12 standard images, which we also use later in Chapter 4. These images are presented in Figure 1.17.



(a) house (b) peppers (c) Cameraman (d) lena (e) barbara (f) boat

(g) hill (h) couple (i) man (j) fingerprint (k) bridge (l) flintstones

Figure 1.17: Dataset of 12 standard images.

We compare the denoising performance of $\ell_0$ and $\ell_1$-regularization, during the training of the dictionary, and the final reconstruction of the image patches. We arbitrarily choose image patches of size $8 \times 8$, following Elad and Aharon (2006) and a dictionary size of $p = 200$ elements. We add synthetic noise to the 12 images, with standard deviations $\sigma$ in $\{5, 10, 15, 20, 25, 50, 75, 100\}$. For each image, we follow the restoration

procedure described above. For learning the dictionary, we alternate 50 times between the minimization of $\mathbf{A}$ and $\mathbf{D}$ using the regularization scheme **(A)**. Then, we decompose again every patch of the image using the regularization scheme **(B)**, before reconstructing the final image with the averaging procedure. The regularization schemes **(A)** and **(B)** for decomposing a patch $\mathbf{y}$ in $\mathbb{R}^m$ can be

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{y}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1, \tag{$\ell_1$-P}$$

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \|\boldsymbol{\alpha}\|_1 \quad \text{s.t.} \quad \|\mathbf{y}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 \leq \varepsilon, \tag{$\ell_1$-R}$$

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{y}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_0, \tag{$\ell_0$-P}$$

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \|\boldsymbol{\alpha}\|_0 \quad \text{s.t.} \quad \|\mathbf{y}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 \leq \varepsilon. \tag{$\ell_0$-R}$$

For $\varepsilon$, we try the values $\boldsymbol{\varepsilon} = Cm\sigma^2$ with $C$ taken on a grid $\{0.9, 0.94, 0.98, \ldots, 1.2\}$, which we refine on an additive scale with step 0.01. For $\lambda$, we try the values $\lambda = 10^i\sigma$, with $i$ taken on a grid $\{-6, -5, \ldots, 1\}$, which we further refine by trying values of the form $\lambda = 10^{\frac{i}{4}}\sigma$, and then $\lambda = 10^{\frac{i}{16}}\sigma$ For each value of noise, we keep the parameters providing the best results on average on the first 3 images, house, peppers and cameraman, after 20 dictionary updates.

The average PSNR of the reconstructed images are presented on Table 1.1, for each value of noise, and each combination where we train with one of of the regularization schemes, and finally reconstruct the image with another one. The corresponding performance of the reconstruction for each individual patches *before the averaging step* is presented on Table 1.2.

Our conclusions from this experiment are the following:

- The averaging step is a key component of the denoising algorithm. The quality of the results is much higher after the averaging step than before.
- For reconstructing *individual patches* (before the averaging step), the $\ell_1$-regularization is significantly better than $\ell_0$ one.
- For reconstructing *full images* (after the averaging step), it is always better to use an $\ell_0$-regularization during the *final reconstruction* than $\ell_1$, but at the same time, it is also better to use an $\ell_1$-regularization during the *dictionary learning step*.
- For large amount of noise, penalized formulations ($\ell_0$-P or $\ell_1$-P) should be prefered to constrained formulations ($\ell_0$-R or $\ell_1$-R). For small standard deviations, one should prefer the constrained formulations.

These conclusions are intriguing, and to the best of our knowledge have only been mentioned before by us in (Mairal et al., 2009c). We do not have clear theoretical explanations for them, but propose the following intuitive arguments.

| | $\sigma = 5$ | | | | $\sigma = 10$ | | | |
|---|---|---|---|---|---|---|---|---|
| **(A)** \ **(B)** | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R |
| $\ell_0$-P | 37.07 | 37.45 | 36.84 | 37.05 | 33.75 | 33.81 | 32.52 | 33.38 |
| $\ell_0$-R | 37.39 | 37.45 | 36.89 | 37.16 | 33.75 | 33.83 | 33.14 | 33.43 |
| $\ell_1$-P | 37.08 | **37.62** | 36.63 | 37.31 | 33.88 | 33.91 | 33.34 | 33.47 |
| $\ell_1$-R | 37.59 | **37.62** | 37.23 | 37.31 | 33.89 | **33.94** | 33.41 | 33.50 |
| | $\sigma = 15$ | | | | $\sigma = 20$ | | | |
| **(A)** \ **(B)** | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R |
| $\ell_0$-P | 31.79 | 31.81 | 31.13 | 31.30 | 30.47 | 30.41 | 29.78 | 29.84 |
| $\ell_0$-R | 31.77 | 31.85 | 31.31 | 31.34 | 30.50 | 30.47 | 29.81 | 29.88 |
| $\ell_1$-P | 31.87 | 31.86 | 31.37 | 31.33 | 30.49 | 30.43 | 29.88 | 29.85 |
| $\ell_1$-R | 31.90 | **31.92** | 31.34 | 31.39 | **30.55** | 30.50 | 29.90 | 29.91 |
| | $\sigma = 25$ | | | | $\sigma = 50$ | | | |
| **(A)** \ **(B)** | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R |
| $\ell_0$-P | 29.44 | 29.34 | 28.76 | 28.71 | 26.23 | 25.95 | 25.57 | 25.31 |
| $\ell_0$-R | 29.47 | 29.40 | 28.81 | 28.76 | **26.24** | 25.95 | 25.53 | 25.28 |
| $\ell_1$-P | 29.43 | 29.34 | 28.82 | 28.71 | 26.22 | 25.93 | 25.65 | 25.29 |
| $\ell_1$-R | **29.52** | 29.41 | 28.87 | 28.78 | **26.24** | 25.94 | 25.59 | 25.26 |
| | $\sigma = 75$ | | | | $\sigma = 100$ | | | |
| **(A)** \ **(B)** | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R |
| $\ell_0$-P | 24.20 | 23.77 | 23.70 | 23.34 | 22.61 | 22.32 | 22.42 | 22.10 |
| $\ell_0$-R | 24.15 | 23.74 | 23.63 | 23.32 | 22.65 | 22.26 | 22.37 | 22.07 |
| $\ell_1$-P | **24.19** | 23.73 | 23.77 | 23.32 | **22.74** | 22.30 | 22.46 | 22.09 |
| $\ell_1$-R | 24.14 | 23.72 | 23.66 | 23.29 | 22.62 | 22.22 | 22.36 | 22.05 |

Table 1.1: Comparison between $\ell_0$ and $\ell_1$-regularizations for image denoising. Results are presented in PSNR. For every value of the standard deviation $\sigma$, we present the results for every combination of regularization schemes, where the ones **(A)** for learning the dictionary are represented on rows, and the ones for reconstructing the image **(B)** on columns. Best results are in bold.

| | $\sigma = 5$ | | | | $\sigma = 10$ | | | |
|---|---|---|---|---|---|---|---|---|
| **(A)** \ **(B)** | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R |
| $\ell_0$-P | 34.28 | 34.61 | 36.00 | 36.01 | 30.66 | 30.74 | 31.49 | 32.14 |
| $\ell_0$-R | 35.01 | 34.82 | 35.95 | 36.08 | 30.90 | 30.96 | 32.03 | 32.17 |
| $\ell_1$-P | 34.28 | 34.50 | 35.55 | 36.26 | 30.52 | 30.63 | 32.16 | 32.22 |
| $\ell_1$-R | 34.66 | 34.63 | **36.31** | 36.26 | 30.78 | 30.77 | **32.26** | **32.26** |
| | $\sigma = 15$ | | | | $\sigma = 20$ | | | |
| **(A)** \ **(B)** | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R |
| $\ell_0$-P | 28.62 | 28.73 | 29.93 | 29.94 | 27.18 | 27.27 | 28.58 | 28.38 |
| $\ell_0$-R | 28.93 | 28.84 | 30.00 | 29.98 | 27.48 | 27.35 | 28.63 | 28.44 |
| $\ell_1$-P | 28.41 | 28.39 | 30.06 | 29.97 | 26.97 | 27.07 | 28.63 | 28.41 |
| $\ell_1$-R | 28.78 | 28.65 | **30.14** | 30.03 | 27.29 | 27.21 | **28.72** | 28.46 |
| | $\sigma = 25$ | | | | $\sigma = 50$ | | | |
| **(A)** \ **(B)** | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R |
| $\ell_0$-P | 26.28 | 26.27 | 27.52 | 27.18 | 23.07 | 22.71 | 24.06 | 23.42 |
| $\ell_0$-R | 26.39 | 26.25 | 27.54 | 27.22 | 22.94 | 22.58 | 23.97 | 23.44 |
| $\ell_1$-P | 26.08 | 26.08 | **27.60** | 27.18 | 23.03 | 22.65 | **24.21** | 23.39 |
| $\ell_1$-R | 26.33 | 26.17 | 27.57 | 27.24 | 22.79 | 22.48 | 24.07 | 23.43 |
| | $\sigma = 75$ | | | | $\sigma = 100$ | | | |
| **(A)** \ **(B)** | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R | $\ell_0$-P | $\ell_0$-R | $\ell_1$-P | $\ell_1$-R |
| $\ell_0$-P | 20.83 | 20.23 | 21.92 | 21.33 | 17.94 | 18.79 | 20.48 | 20.01 |
| $\ell_0$-R | 20.72 | 20.21 | 21.80 | 21.31 | 19.06 | 18.69 | 20.38 | 19.95 |
| $\ell_1$-P | 20.88 | 20.37 | **22.09** | 21.31 | 19.24 | 18.78 | **20.56** | 20.00 |
| $\ell_1$-R | 20.65 | 20.14 | 21.84 | 21.28 | 19.02 | 18.63 | 20.35 | 19.93 |

Table 1.2: Comparison between $\ell_0$ and $\ell_1$-regularizations for denoising individual patches. Results are presented in PSNR. For every value of the standard deviation $\sigma$, we present the results for every combination of regularization schemes, where the ones **(A)** for learning the dictionary are represented on rows, and the ones for reconstructing the image **(B)** on columns. Best results are in bold.

We believe that the reason of the good performance of $\ell_1$-regularization for learning the dictionary might be: (i) a better stability of the sparsity patterns than the ones obtained with $\ell_0$, and/or (ii) a better behavior in terms of optimization, where the $\ell_1$-schemes guarantee to obtain a stationary point of the formulation, whereas $\ell_0$ does not.

The stability argument that favors $\ell_1$ might explain why this regularization is better than $\ell_0$ for individual patches. However, the reason why the hierarchy is reversed after the averaging step remain alusive. It may be that the errors made with $\ell_0$ are greater than with $\ell_1$ for individual patches, but are quite independent from a patch to another one, even when the latter overlap. This would explain why these errors are greatly reduced by the averaging step. As for the $\ell_1$-regularization, the errors are individually smaller, but are highly correlated from one patch to another, and do not average well. One could argue that the Lasso estimator is biased, and indeed it is classical to use the Lasso for selecting the dictionary elements, and then perform an orthogonal projection onto the span of these selected dictionary elements to obtain an unbiased estimator (see Hastie et al., 2009, and references therein). This argument is true in part, and we have indeed observed that the quality of images obtained with the $\ell_1$ reconstruction improve with this modification, but it is not sufficient. Even with un unbiased estimator based on $\ell_1$, significantly better results are obtained using greedy approaches. Note also that these conclusions stand for the dictionary learning approach based on alternate minimization which we have described before, but we have not observed significant differences when trying other approaches such as the online learning procedure we present in Chapter 2, or the K-SVD introduced by Aharon et al. (2006).

# 2

# Online Learning for Matrix Factorization and Sparse Coding

**Chapter abstract:** Sparse coding—that is, modelling data vectors as sparse linear combinations of basis elements—is widely used in machine learning, neuroscience, signal processing, and statistics. This work focuses on the large-scale matrix factorization problem that consists of *learning* the basis set in order to adapt it to specific data. Variations of this problem include dictionary learning in signal processing, non-negative matrix factorization and sparse principal component analysis. In this work, we propose to address these tasks with a new online optimization algorithm, based on stochastic approximations, which scales up gracefully to large data sets with millions of training samples, and extends naturally to various matrix factorization formulations, making it suitable for a wide range of learning problems. A proof of convergence is presented, along with experiments with natural images and genomic data demonstrating that it leads to state-of-the-art performance in terms of speed and optimization for both small and large data sets.

The reader is advised to read the Section 1.5 on dictionary learning before reading this chapter. The material of this part is based on the two following publications:

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

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.

## 2.1 Introduction

In machine learning, statistics and signal processing, slightly different matrix factorization problems are formulated in order to obtain a few *interpretable* basis elements from a set of data vectors. This includes dictionary learning, non-negative matrix factorization and its variants (Lee and Seung, 2001; Hoyer, 2002, 2004; Lin, 2007), and sparse principal component analysis (Zou et al., 2006; d'Aspremont et al., 2007, 2008; Witten et al., 2009; Zass and Shashua, 2007). As shown in this chapter, these problems have

strong similarities; even though we first focus on the problem of dictionary learning, the algorithm we propose is able to address all of them. While learning the dictionary has proven to be critical to achieve (or improve upon) state-of-the-art results in signal and image processing, effectively solving the corresponding optimization problem is a significant computational challenge, particularly in the context of large-scale data sets that may include millions of training samples. Addressing this challenge and designing a generic algorithm which is capable of efficiently handling various matrix factorization problems, is the topic of this chapter.

Most recent algorithms for dictionary learning (Engan et al., 1999; Lewicki and Sejnowski, 2000; Aharon et al., 2006; Lee et al., 2007) are iterative *batch* procedures, accessing the whole training set at each iteration in order to minimize a cost function under some constraints, and cannot efficiently deal with very large training sets (Bottou and Bousquet, 2008), or dynamic training data changing over time, such as video sequences. To address these issues, we propose an *online* approach that processes the signals, one at a time, or in mini-batches. This is particularly important in the context of image and video processing (Protter and Elad, 2009; Mairal et al., 2008d), where it is common to learn dictionaries adapted to small patches, with training data that may include several millions of these patches (roughly one per pixel and per frame). In this setting, online techniques based on stochastic approximations are an attractive alternative to batch methods (see, e.g., Bottou, 1998; Kushner and Yin, 2003; Shalev-Shwartz et al., 2009). For example, first-order stochastic gradient descent with projections on the constraint set (Kushner and Yin, 2003) is sometimes used for dictionary learning (see Olshausen and Field, 1997, 1996; Aharon and Elad, 2008; Kavukcuoglu et al., 2008 for instance). We show in this chapter that it is possible to go further and exploit the specific structure of sparse coding in the design of an optimization procedure tuned to this problem, with low memory consumption and lower computational cost than classical batch algorithms. As demonstrated by our experiments, it scales up gracefully to large data sets with millions of training samples, is easy to use, and is faster than competitive methods.

The chapter is structured as follows: Section 2.2 briefly recalls the dictionary learning problem. The proposed method is introduced in Section 2.3, with a proof of convergence in Section 2.4. Section 2.5 extends our algorithm to various matrix factorization problems that generalize dictionary learning, and Section 2.6 is devoted to experimental results, demonstrating that our algorithm is suited to a wide class of learning problems.

### 2.1.1 Contributions

This chapter makes four main contributions:

- We cast in Section 2.2 the dictionary learning problem as the optimization of a smooth nonconvex objective function over a convex set, minimizing the (desired) *expected* cost when the training set size goes to infinity, and propose in Section 2.3 an iterative online algorithm that solves this problem by efficiently minimizing at each step a quadratic surrogate function of the empirical cost over the set of

constraints. This method is shown in Section 2.4 to converge almost surely to a stationary point of the objective function.

- As shown experimentally in Section 2.6, our algorithm is significantly faster than previous approaches to dictionary learning on both small and large data sets of natural images. To demonstrate that it is adapted to difficult, large-scale image-processing tasks, we learn a dictionary on a 12-Megapixel photograph and use it for inpainting—that is, filling some holes in the image.

- We show in Sections 2.5 and 2.6 that our approach is suitable to large-scale matrix factorization problems such as non-negative matrix factorization and sparse principal component analysis, while being still effective on small data sets.

- To extend our algorithm to several matrix factorization problems, we propose in Appendix C efficient procedures for projecting onto two convex sets, which can be useful for other applications that are beyond the scope of this chapter.

## 2.2 Problem Statement

Classical dictionary learning techniques for sparse representation (Engan et al., 1999; Lewicki and Sejnowski, 2000; Aharon et al., 2006; Lee et al., 2007) consider a finite training set of signals $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ in $\mathbb{R}^{m \times n}$ and optimize the empirical cost function

$$f_n(\mathbf{D}) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{x}^i, \mathbf{D}), \tag{2.1}$$

where $\mathbf{D} = [\mathbf{d}^1, \ldots, \mathbf{d}^p]$ in $\mathbb{R}^{m \times p}$ is the dictionary, each column representing a basis vector, and $\ell$ is a loss function such that $\ell(\mathbf{x}, \mathbf{D})$ should be small if $\mathbf{D}$ is "good" at representing the signal $\mathbf{x}$ in a sparse fashion. The number of samples $n$ is usually large, whereas the signal dimension $m$ is relatively small, for example, $m = 100$ for $10 \times 10$ image patches, and $n \geq 100,000$ for typical image processing applications. In general, we also have $p \ll n$ (e.g., $p = 200$ for $n = 100,000$), but each signal only uses a few elements of $\mathbf{D}$ in its representation, say 10 for instance. Note that, in this setting, overcomplete dictionaries with $p > m$ are allowed. As others (see for example Olshausen and Field, 1997, 1996; Lee et al., 2007), we define $\ell(\mathbf{x}, \mathbf{D})$ as the optimal value of the $\ell_1$ *sparse coding* problem:

$$\ell(\mathbf{x}, \mathbf{D}) \triangleq \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1, \tag{2.2}$$

where $\lambda$ is a regularization parameter. To prevent $\mathbf{D}$ from having arbitrarily large values (which would lead to arbitrarily small values of $\boldsymbol{\alpha}$), it is common to constrain its columns $\mathbf{d}^1, \ldots, \mathbf{d}^p$ to have an $\ell_2$-norm less than or equal to one. We will call $\mathcal{D}$ the convex set of matrices verifying this constraint:

$$\mathcal{D} \triangleq \{\mathbf{D} \in \mathbb{R}^{m \times p} \quad \text{s.t.} \quad \forall j \in [\![1; p]\!], \quad \|\mathbf{d}^j\|_2^2 \leq 1\}.$$

Note that the problem of minimizing the empirical cost $f_n(\mathbf{D})$ is not convex with respect to $\mathbf{D}$. It can be rewritten as a joint optimization problem with respect to the dictionary $\mathbf{D}$ and the coefficients $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n]$ in $\mathbb{R}^{p \times n}$ of the sparse decompositions, which is not jointly convex, but convex with respect to each of the two variables $\mathbf{D}$ and $\boldsymbol{\alpha}$ when the other one is fixed:

$$\min_{\mathbf{D} \in \mathcal{D}, \mathbf{A} \in \mathbb{R}^{p \times n}} \sum_{i=1}^{n} \Big[ \frac{1}{2} \|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda \|\boldsymbol{\alpha}^i\|_1 \Big]. \tag{2.3}$$

This can be rewritten as a *matrix factorization* problem with a sparsity penalty:

$$\min_{\mathbf{D} \in \mathcal{D}, \mathbf{A} \in \mathbb{R}^{p \times n}} \frac{1}{2} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_{\mathrm{F}}^2 + \lambda \|\mathbf{A}\|_{1,1},$$

where, as before, $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ is the matrix of data vectors, and $\|\mathbf{A}\|_{1,1}$ denotes the $\ell_1$ norm of the matrix $\mathbf{A}$—that is, the sum of the magnitude of its coefficients. A natural approach for solving this problem is to alternate between the two variables, minimizing over one while keeping the other one fixed, as proposed by Lee et al. (2007) (see also Engan et al. 1999 and Aharon et al. 2006, who use $\ell_0$ rather than $\ell_1$ penalties, or Zou et al. 2006 for the problem of sparse principal component analysis).[1] Since the computation of the coefficients vectors $\boldsymbol{\alpha}^i$ dominates the cost of each iteration in this block-coordinate descent approach, a second-order optimization technique can be used to accurately estimate $\mathbf{D}$ at each step when $\boldsymbol{\alpha}$ is fixed.

As pointed out by Bottou and Bousquet (2008), however, one is usually not interested in the minimization of the *empirical cost* $f_n(\mathbf{D})$ with high precision, but instead in the minimization of the *expected cost*

$$f(\mathbf{D}) \triangleq \mathbb{E}_{\mathbf{x}}[\ell(\mathbf{x}, \mathbf{D})] = \lim_{n \to \infty} f_n(\mathbf{D}) \quad \text{a.s.},$$

where the expectation (which is supposed finite) is taken relative to the (unknown) probability distribution $p(\mathbf{x})$ of the data.[2] In particular, given a finite training set, one should not spend too much effort on accurately minimizing the empirical cost, since it is only an approximation of the expected cost. An "inaccurate" solution may indeed have the same or better expected cost than a "well-optimized" one. Bottou and Bousquet (2008) further show that stochastic gradient algorithms, whose rate of convergence is very poor in conventional optimization terms, may in fact in certain settings be shown both theoretically and empirically to be faster in reaching a solution with low expected cost than second-order batch methods. With large training sets, the risk of overfitting is lower, but classical optimization techniques may become impractical in terms of speed or memory requirements.

In the case of dictionary learning, the classical projected first-order projected stochastic gradient descent algorithm (as used by Olshausen and Field 1997, 1996; Aharon and

[1] In our setting, as in Lee et al. (2007), we have preferred to use the convex $\ell_1$ norm, that has empirically proven to be better behaved in general than the $\ell_0$ pseudo-norm for dictionary learning.

[2] We use "a.s." to denote almost sure convergence.

Elad 2008; Kavukcuoglu et al. 2008 for instance) consists of a sequence of updates of $\mathbf{D}$:

$$\mathbf{D}^t = \Pi_{\mathcal{D}}\Big[\mathbf{D}^{t-1} - \delta_t \nabla_{\mathbf{D}} \ell(\mathbf{x}^t, \mathbf{D}^{t-1})\Big],$$

where $\mathbf{D}^t$ is the estimate of the optimal dictionary at iteration $t$, $\delta_t$ is the gradient step, $\Pi_{\mathcal{D}}$ is the orthogonal projector onto $\mathcal{D}$, and the vectors $\mathbf{x}^t$ are i.i.d. samples of the (unknown) distribution $p(\mathbf{x})$. Even though it is often difficult to obtain such i.i.d. samples, the vectors $\mathbf{x}^t$ are in practice obtained by cycling on a randomly permuted training set. As shown in Section 2.6, we have observed that this method can be competitive in terms of speed compared to batch methods when the training set is large and when $\delta_t$ is carefully chosen. In particular, good results are obtained using a learning rate of the form $\delta_t \triangleq a/(t+b)$, where $a$ and $b$ have to be well chosen in a data set-dependent way. Note that first-order stochastic gradient descent has also been used for other matrix factorization problems (see Koren et al., 2009 and references therein).

The optimization method we present in the next section falls into the class of online algorithms based on stochastic approximations, processing one sample at a time (or a mini-batch), but further exploits the specific structure of the problem to efficiently solve it by sequentially minimizing a quadratic local surrogate of the expected cost. As shown in Section 2.3.5, it uses second-order information of the cost function, allowing the optimization without any explicit learning rate tuning.

## 2.3 Proposed Approach

We present in this section the basic components of our online algorithm for dictionary learning (Sections 2.3.1–2.3.3), as well as a few minor variants which speed up our implementation in practice (Section 2.3.4) and show some links with a Kalman algorithm (Section 2.3.5).

### 2.3.1 Algorithm Outline

Our procedure is summarized in Algorithm 1. Assuming that the training set is composed of i.i.d. samples of a distribution $p(\mathbf{x})$, its inner loop draws one element $\mathbf{x}^t$ at a time, as in stochastic gradient descent, and alternates classical sparse coding steps for computing the decomposition $\boldsymbol{\alpha}^t$ of $\mathbf{x}^t$ over the dictionary $\mathbf{D}^{t-1}$ obtained at the previous iteration, with dictionary update steps where the new dictionary $\mathbf{D}^t$ is computed by minimizing over $\mathcal{D}$ the function

$$\hat{f}_t(\mathbf{D}) \triangleq \frac{1}{t}\sum_{i=1}^{t}\Big[\frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda\|\boldsymbol{\alpha}^i\|_1\Big], \tag{2.4}$$

and the vectors $\boldsymbol{\alpha}^i$, for $i < t$, have been computed during the previous steps of the algorithm. The motivation behind this approach is twofold:

- The function $\hat{f}_t$, which is quadratic in $\mathbf{D}$, aggregates the past information with a few sufficient statistics obtained during the previous steps of the algorithm, namely

the vectors $\boldsymbol{\alpha}^i$, and it is easy to show that it upperbounds the empirical cost $f_t(\mathbf{D}^t)$ from Eq. (2.1). One key aspect of our convergence analysis will be to show that $\hat{f}_t(\mathbf{D}^t)$ and $f_t(\mathbf{D}^t)$ converge almost surely to the same limit, and thus that $\hat{f}_t$ acts as a *surrogate* for $f_t$.

- Since $\hat{f}_t$ is close to $\hat{f}_{t-1}$ for large values of $t$, so are $\mathbf{D}^t$ and $\mathbf{D}^{t-1}$, under suitable assumptions, which makes it efficient to use $\mathbf{D}^{t-1}$ as warm restart for computing $\mathbf{D}^t$.

**Algorithm 1** Online dictionary learning.

**Require:** $\mathbf{x} \in \mathbb{R}^m \sim p(\mathbf{x})$ (random variable and an algorithm to draw i.i.d samples of $p$), $\lambda \in \mathbb{R}$ (regularization parameter), $\mathbf{D}^0 \in \mathbb{R}^{m \times p}$ (initial dictionary), $T$ (number of iterations).

1: $\mathbf{B}^0 \in \mathbb{R}^{p \times p} \leftarrow 0$, $\mathbf{C}^0 \in \mathbb{R}^{m \times p} \leftarrow 0$ (reset the "past" information).
2: **for** $t = 1$ to $T$ **do**
3: Draw $\mathbf{x}_t$ from $p(\mathbf{x})$.
4: Sparse coding: compute using LARS

$$\boldsymbol{\alpha}^t \triangleq \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{x}^t - \mathbf{D}^{t-1}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1.$$

5: $\mathbf{B}^t \leftarrow \mathbf{B}^{t-1} + \boldsymbol{\alpha}^t \boldsymbol{\alpha}^{t\top}$.
6: $\mathbf{C}^t \leftarrow \mathbf{C}^{t-1} + \mathbf{x}^t \boldsymbol{\alpha}^{t\top}$.
7: Compute $\mathbf{D}^t$ using Algorithm 2, with $\mathbf{D}^{t-1}$ as warm restart, so that

$$\begin{aligned} \mathbf{D}^t &\triangleq \arg\min_{\mathbf{D} \in \mathcal{D}} \frac{1}{t}\sum_{i=1}^{t}\Big(\frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda\|\boldsymbol{\alpha}^i\|_1\Big), \\ &= \arg\min_{\mathbf{D} \in \mathcal{D}} \frac{1}{t}\Big(\frac{1}{2}\operatorname{Tr}(\mathbf{D}^\top \mathbf{D}\mathbf{B}^t) - \operatorname{Tr}(\mathbf{D}^\top \mathbf{C}^t)\Big). \end{aligned} \tag{2.5}$$

8: **end for**
9: **return** $\mathbf{D}^T$ (learned dictionary).

### 2.3.2 Sparse Coding

The sparse coding problem of Eq. (2.2) with fixed dictionary is an $\ell_1$-regularized linear least-squares problem. A number of recent methods for solving this type of problems are based on coordinate descent with soft thresholding (Fu, 1998; Friedman et al., 2007; Wu and Lange, 2008). When the columns of the dictionary have low correlation, we have observed that these simple methods are very efficient. However, the columns of learned dictionaries are in general highly correlated, and we have empirically observed that these algorithms become much slower in this setting. This has led us to use instead the LARS-Lasso algorithm, a homotopy method (Osborne et al., 2000b; Efron et al., 2004) that

**Algorithm 2** Dictionary update.

**Require:** $\mathbf{D} = [\mathbf{d}^1, \ldots, \mathbf{d}^p] \in \mathbb{R}^{m \times p}$ (input dictionary),
$\mathbf{B} = [\mathbf{b}^1, \ldots, \mathbf{b}^p] \in \mathbb{R}^{p \times p}$,
$\mathbf{C} = [\mathbf{c}^1, \ldots, \mathbf{c}^p] \in \mathbb{R}^{m \times p}$.

1: **repeat**
2: **for** $j = 1$ to $p$ **do**
3: Update the $j$-th column to optimize for (2.5):

$$\begin{aligned} \mathbf{u}^j &\leftarrow \frac{1}{\mathbf{B}_{jj}}(\mathbf{c}^j - \mathbf{D}\mathbf{b}^j) + \mathbf{d}^j, \\ \mathbf{d}^j &\leftarrow \frac{1}{\max(\|\mathbf{u}^j\|_2, 1)}\mathbf{u}^j. \end{aligned} \tag{2.6}$$

4: **end for**
5: **until convergence**
6: **return** $\mathbf{D}$ (updated dictionary).

provides the whole regularization path—that is, the solutions for all possible values of $\lambda$. With an efficient Cholesky-based implementation (see Efron et al., 2004; Zou and Hastie, 2005) for brief descriptions of such implementations), it has proven experimentally at least as fast as approaches based on soft thresholding, while providing the solution with a higher accuracy and being more robust as well since it does not require an arbitrary stopping criterion. We provide more details on these methods in Sections 1.4.3–1.4.2.

### 2.3.3 Dictionary Update

Our algorithm for updating the dictionary uses block-coordinate descent with warm restarts (see Bertsekas, 1999). One of its main advantages is that it is parameter free and does not require any learning rate tuning. Moreover, the procedure does not require to store all the vectors $\mathbf{x}^i$ and $\boldsymbol{\alpha}^i$, but only the matrices $\mathbf{B}^t = \sum_{i=1}^t \boldsymbol{\alpha}^i \boldsymbol{\alpha}^{i\top}$ in $\mathbb{R}^{p \times p}$ and $\mathbf{C}^t = \sum_{i=1}^t \mathbf{x}^i \boldsymbol{\alpha}^{i\top}$ in $\mathbb{R}^{m \times p}$. Concretely, Algorithm 2 sequentially updates each column of $\mathbf{D}$. A simple calculation shows that solving (2.5) with respect to the $j$-th column $\mathbf{d}^j$, while keeping the other ones fixed under the constraint $\|\mathbf{d}^j\|_2 \leq 1$, amounts to an orthogonal projection of the vector $\mathbf{u}^j$ defined in Eq. (2.6), onto the constraint set, namely the $\ell_2$-ball here, which is solved by Eq. (2.6). Since the convex optimization problem (2.5) admits separable constraints in the updated blocks (columns), convergence to a global optimum is guaranteed (Bertsekas, 1999). In practice, the vectors $\boldsymbol{\alpha}^i$ are sparse and the coefficients of the matrix $\mathbf{B}^t$ are often concentrated on the diagonal, which makes the block-coordinate descent more efficient.[3] After a few iterations of our algorithm, using the value of $\mathbf{D}^{t-1}$ as a warm restart for computing $\mathbf{D}^t$ becomes effective,

[3] We have observed that this is true when the columns of $\mathbf{D}$ are not too correlated. When a group of columns in $\mathbf{D}$ are highly correlated, the coefficients of the matrix $\mathbf{B}^t$ concentrate instead on the corresponding principal submatrices of $\mathbf{B}^t$.

and a single iteration of Algorithm 2 has empirically found to be sufficient to achieve convergence of the dictionary update step. Other approaches have been proposed to update $\mathbf{D}$: For instance, Lee et al. (2007) suggest using a Newton method on the dual of Eq. (2.5), but this requires inverting a $p \times p$ matrix at each Newton iteration, which is impractical for an online algorithm.

### 2.3.4 Optimizing the Algorithm

We have presented so far the basic building blocks of our algorithm. This section discusses a few simple improvements that significantly enhance its performance.

#### Handling Fixed-Size Data Sets

In practice, although it may be very large, the size of the training set often has a predefined finite size (of course this may not be the case when the data must be treated on the fly like a video stream for example). In this situation, the same data points may be examined several times, and it is very common in online algorithms to simulate an i.i.d. sampling of $p(\mathbf{x})$ by cycling over a randomly permuted training set (see Bottou and Bousquet, 2008 and references therein). This method works experimentally well in our setting but, when the training set is small enough, it is possible to further speed up convergence: In Algorithm 1, the matrices $\mathbf{B}^t$ and $\mathbf{C}^t$ carry all the information from the past coefficients $\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^t$. Suppose that at time $t_0$, a signal $\mathbf{x}$ is drawn and the vector $\boldsymbol{\alpha}^{t_0}$ is computed. If the same signal $\mathbf{x}$ is drawn again at time $t > t_0$, then it is natural to replace the "old" information $\boldsymbol{\alpha}^{t_0}$ by the new vector $\boldsymbol{\alpha}^t$ in the matrices $\mathbf{B}^t$ and $\mathbf{C}^t$—that is, $\mathbf{B}^t \leftarrow \mathbf{B}^{t-1} + \boldsymbol{\alpha}^t \boldsymbol{\alpha}^{t\top} - \boldsymbol{\alpha}^{t_0} \boldsymbol{\alpha}^{t_0\top}$ and $\mathbf{C}^t \leftarrow \mathbf{C}^{t-1} + \mathbf{x}^t \boldsymbol{\alpha}^{t\top} - \mathbf{x}^t \boldsymbol{\alpha}^{t_0\top}$. In this setting, which requires storing all the past coefficients $\boldsymbol{\alpha}^{t_0}$, this method amounts to a block-coordinate descent for the problem of minimizing Eq. (2.3). When dealing with large but finite sized training sets, storing all coefficients $\boldsymbol{\alpha}^i$ is impractical, but it is still possible to partially exploit the same idea, by removing the information from $\mathbf{B}^t$ and $\mathbf{C}^t$ that is older than two *epochs* (cycles through the data), through the use of two auxiliary matrices $\tilde{\mathbf{B}}^t$ and $\tilde{\mathbf{C}}^t$ of size $p \times p$ and $m \times p$ respectively. These two matrices should be built with the same rules as $\mathbf{B}^t$ and $\mathbf{C}^t$, except that at the end of an epoch, $\mathbf{B}^t$ and $\mathbf{C}^t$ are respectively replaced by $\tilde{\mathbf{B}}^t$ and $\tilde{\mathbf{C}}^t$, while $\tilde{\mathbf{B}}^t$ and $\tilde{\mathbf{C}}^t$ are set to 0. Thanks to this strategy, $\mathbf{B}^t$ and $\mathbf{C}^t$ do not carry any coefficients $\boldsymbol{\alpha}^i$ older than two epochs.

#### Scaling the "Past" Data

At each iteration, the "new" information $\boldsymbol{\alpha}^t$ that is added to the matrices $\mathbf{B}^t$ and $\mathbf{C}^t$ has the same weight as the "old" one. A simple and natural modification to the algorithm is to rescale the "old" information so that newer coefficients $\boldsymbol{\alpha}^t$ have more weight, which is classical in online learning. For instance, Neal and Hinton (1998) present an online algorithm for EM, where sufficient statistics are aggregated over time, and an exponential decay is used to forget out-of-date statistics. In this work, we propose to replace lines 5

and 6 of Algorithm 1 by

$$\mathbf{B}^t \leftarrow \beta_t \mathbf{B}^{t-1} + \boldsymbol{\alpha}^t \boldsymbol{\alpha}^{t\top},$$
$$\mathbf{C}^t \leftarrow \beta_t \mathbf{C}^{t-1} + \mathbf{x}^t \boldsymbol{\alpha}^{t\top},$$

where $\beta_t \triangleq (1 - \frac{1}{t})^\rho$, and $\rho$ is a new parameter. In practice, one can apply this strategy after a few iterations, once $\mathbf{B}^t$ is well-conditioned. Tuning $\rho$ improves the convergence rate, when the training sets are large, even though, as shown in Section 2.6, it is not critical. To understand better the effect of this modification, note that Eq. (2.5) becomes

$$\begin{aligned}\mathbf{D}_t &\triangleq \arg\min_{\mathbf{D}\in\mathcal{D}} \frac{1}{\sum_{j=1}^t (j/t)^\rho} \sum_{i=1}^t \Big(\frac{i}{t}\Big)^\rho \Big(\frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda\|\boldsymbol{\alpha}^i\|_1\Big),\\ &= \arg\min_{\mathbf{D}\in\mathcal{D}} \frac{1}{\sum_{j=1}^t (j/t)^\rho} \Big(\frac{1}{2}\operatorname{Tr}(\mathbf{D}^\top\mathbf{D}\mathbf{B}^t) - \operatorname{Tr}(\mathbf{D}^\top\mathbf{C}^t)\Big).\end{aligned}$$

When $\rho = 0$, we obtain the original version of the algorithm. Of course, different strategies and heuristics could also be investigated. In practice, this parameter $\rho$ is useful for large data sets only ($n \geq 100\,000$). For smaller data sets, we have not observed a better performance when using this extension.

### Mini-Batch Extension

In practice, we can also improve the convergence speed of our algorithm by drawing $\eta > 1$ signals at each iteration instead of a single one, which is a classical heuristic in stochastic gradient descent algorithms. In our case, this is further motivated by the fact that the complexity of computing $\eta$ vectors $\boldsymbol{\alpha}^i$ is not linear in $\eta$. A Cholesky-based implementation of LARS-Lasso for decomposing a single signal has a complexity of $O(pms + ps^2)$, where $s$ is the number of nonzero coefficients. When decomposing $\eta$ signals, it is possible to pre-compute the Gram matrix $\mathbf{D}^{t\top}\mathbf{D}^t$ and the total complexity becomes $O(p^2m + \eta(pm + ps^2))$, which is much cheaper than $\eta$ times the previous complexity when $\eta$ is large enough and $s$ is small. Let us denote by $\mathbf{x}^{t,1}, \ldots, \mathbf{x}^{t,\eta}$ the signals drawn at iteration $t$. We can now replace lines 5 and 6 of Algorithm 1 by

$$\mathbf{B}^t \leftarrow \mathbf{B}^{t-1} + \frac{1}{\eta}\sum_{i=1}^\eta \boldsymbol{\alpha}^{t,i}\boldsymbol{\alpha}^{t,i\top},$$
$$\mathbf{C}^t \leftarrow \mathbf{C}^{t-1} + \frac{1}{\eta}\sum_{i=1}^\eta \mathbf{x}^{t,i}\boldsymbol{\alpha}^{t,i\top}.$$

### Slowing Down the First Iterations

As in the case of stochastic gradient descent, the first iterations of our algorithm may update the parameters with large steps, immediately leading to large deviations from the initial dictionary. To prevent this phenomenon, classical implementations of stochastic gradient descent use gradient steps of the form $a/(t+b)$, where $b$ "reduces" the step size.

An initialization of the form $\mathbf{B}^0 = t_0\mathbf{I}$ and $\mathbf{C}^0 = t_0\mathbf{D}^0$ with $t_0 \geq 0$ also slows down the first steps of our algorithm by forcing the solution of the dictionary update to stay close to $\mathbf{D}^0$. As shown in Section 2.6, we have observed that our method does not require this extension to achieve good results in general.

**Purging the Dictionary from Unused Atoms**

Every dictionary learning technique sometimes encounters situations where some of the dictionary atoms are never (or very seldom) used, which typically happens with a very bad initialization. A common practice is to replace these during the optimization by randomly chosen elements of the training set, which solves in practice the problem in most cases. For more difficult and highly regularized cases, it is also possible to choose a continuation strategy consisting of starting from an easier, less regularized problem, and gradually increasing $\lambda$. This continuation method has not been used in this work.

### 2.3.5 Link with Second-order Stochastic Gradient Descent

For unconstrained learning problems with twice differentiable expected cost, the second-order stochastic gradient descent algorithm (see Bottou and Bousquet, 2008 and references therein) improves upon its first-order version, by replacing the learning rate by the inverse of the Hessian. When this matrix can be computed or approximated efficiently, this method usually yields a faster convergence speed and removes the problem of tuning the learning rate. However, it cannot be applied easily to constrained optimization problems and requires at every iteration an inverse of the Hessian. For these two reasons, it cannot be used for the dictionary learning problem, but nevertheless it shares some similarities with our algorithm, which we illustrate with the example of a different problem.

Suppose that two major modifications are brought to our original formulation: (i) the vectors $\boldsymbol{\alpha}^t$ are independent of the dictionary $\mathbf{D}$—that is, they are drawn at the same time as $\mathbf{x}^t$; (ii) the optimization is unconstrained—that is, $\mathcal{D} = \mathbb{R}^{m\times p}$. This setting leads to the least-square estimation problem

$$\min_{\mathbf{D}\in\mathbb{R}^{m\times p}} \mathbb{E}_{(\mathbf{x},\boldsymbol{\alpha})}\left[\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2\right], \tag{2.7}$$

which is of course different from the original dictionary learning formulation. Nonetheless, it is possible to address Eq. (2.7) with our method and show that it amounts to using the recursive formula

$$\mathbf{D}^t \leftarrow \mathbf{D}^{t-1} + (\mathbf{x}^t - \mathbf{D}^{t-1}\boldsymbol{\alpha}^t)\boldsymbol{\alpha}^{t\top}\Big(\sum_{i=1}^{t}\boldsymbol{\alpha}^i\boldsymbol{\alpha}^{i\top}\Big)^{-1},$$

which is equivalent to a second-order stochastic gradient descent algorithm: The gradient obtained at $(\mathbf{x}^t, \boldsymbol{\alpha}^t)$ is the term $-(\mathbf{x}^t - \mathbf{D}^{t-1}\boldsymbol{\alpha}^t)\boldsymbol{\alpha}^{t\top}$, and the sequence $(1/t)\sum_{i=1}^{t}\boldsymbol{\alpha}^i\boldsymbol{\alpha}^{i\top}$ converges to the Hessian of the objective function. Such sequence of updates admit a fast implementation called Kalman algorithm (see Kushner and Yin, 2003; Bottou, 1998 and references therein).

## 2.4 Convergence Analysis

The main tools used in our proofs are the convergence of empirical processes (Van der Vaart, 1998) and, following Bottou (1998), the convergence of quasi-martingales (Fisk, 1965). Our analysis is limited to the basic version of the algorithm, although it can in principle be carried over to the optimized versions discussed in Section 2.3.4. Before proving our main result, let us first discuss the (reasonable) assumptions under which our analysis holds.

### 2.4.1 Assumptions

**(A) The data admits a distribution with compact support** $K$. Assuming a compact support for the data is natural in audio, image, and video processing applications, where it is imposed by the data acquisition process.

**(B) The quadratic surrogate functions $\hat{f}_t$ are strictly convex with lower-bounded Hessians.** We assume that the smallest eigenvalue of the positive semi-definite matrix $\frac{1}{t}\mathbf{B}^t$ defined in Algorithm 1 is greater than or equal to some constant $\kappa_1$. As a consequence, $\mathbf{B}^t$ is invertible and $\hat{f}_t$ is strictly convex with Hessian $\mathbf{I} \otimes \frac{2}{t}\mathbf{B}^t$. This hypothesis is in practice verified experimentally after a few iterations of the algorithm when the initial dictionary is reasonable, consisting for example of a few elements from the training set, or any common dictionary, such as DCT (bases of cosines products) or wavelets (Mallat, 1999). Note that it is easy to enforce this assumption by adding a term $\frac{\kappa_1}{2}\|\mathbf{D}\|_{\mathrm{F}}^2$ to the objective function, which is equivalent to replacing the positive semi-definite matrix $\frac{1}{t}\mathbf{B}^t$ by $\frac{1}{t}\mathbf{B}^t + \kappa_1\mathbf{I}$. We have omitted for simplicity this penalization in our analysis.

**(C) A particular sufficient condition for the uniqueness of the sparse coding solution is satisfied.** Before presenting this assumption, let us briefly recall classical optimality conditions for the $\ell_1$ decomposition problem in Eq. (2.2) (Fuchs, 2005). For $\mathbf{x}$ in $K$ and $\mathbf{D}$ in $\mathcal{D}$, $\boldsymbol{\alpha}$ in $\mathbb{R}^p$ is a solution of Eq. (2.2) if and only if

$$\begin{aligned} &\mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}) = \lambda \operatorname{sign}(\boldsymbol{\alpha}_j) \quad \text{if} \quad \boldsymbol{\alpha}_j \neq 0, \\ &|\mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})| \leq \lambda \quad \text{otherwise.} \end{aligned} \tag{2.8}$$

Let $\boldsymbol{\alpha}^\star$ be such a solution. Denoting by $\Lambda$ the set of indices $j$ such that $|\mathbf{d}^{j\top}(\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}^\star)| = \lambda$, and $\mathbf{D}_\Lambda$ the matrix composed of the columns from $\mathbf{D}$ restricted to the set $\Lambda$, it is easy to see from Eq. (2.8) that the solution $\boldsymbol{\alpha}^\star$ is necessary unique if $(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)$ is invertible and that

$$\boldsymbol{\alpha}^\star_\Lambda = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1}(\mathbf{D}_\Lambda^\top \mathbf{x} - \lambda \boldsymbol{\varepsilon}_\Lambda), \tag{2.9}$$

where $\boldsymbol{\alpha}^\star_\Lambda$ is the vector containing the values of $\boldsymbol{\alpha}^\star$ corresponding to the set $\Lambda$ and $\boldsymbol{\varepsilon}_\Lambda$ carries the signs of $\boldsymbol{\alpha}^\star_\Lambda$ (elementwise). With this preliminary uniqueness condition in hand, we can now formulate our assumption: *We assume that there exists* $\kappa_2 > 0$ *such that, for all* $\mathbf{x}$ *in* $K$ *and all dictionaries* $\mathbf{D}$ *in the subset of* $\mathcal{D}$ *considered by our algorithm,*

*the smallest eigenvalue of* $\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda$ *is greater than or equal to* $\kappa_2$. This guarantees the invertibility of $(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)$ and therefore the uniqueness of the solution of Eq. (2.2). It is of course easy to build a dictionary $\mathbf{D}$ for which this assumption fails. However, having $\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda$ invertible is a common assumption in linear regression and in methods such as the LARS algorithm aimed at solving Eq. (2.2) (Efron et al., 2004). It is also possible to enforce this condition using an elastic net penalization (Zou and Hastie, 2005), replacing $\|\boldsymbol{\alpha}\|_1$ by $\|\boldsymbol{\alpha}\|_1 + \frac{\kappa_2}{2}\|\boldsymbol{\alpha}\|_2^2$ and thus improving the numerical stability of homotopy algorithms, which is the choice made by Zou et al. (2006). Again, we have omitted this penalization in our analysis.

### 2.4.2 Main Results

Given assumptions **(A)**–**(C)**, let us now show that our algorithm converges to a stationary point of the objective function. Since this work is dealing with non-convex optimization, neither our algorithm nor any one in the literature is guaranteed to find the global optimum of the optimization problem. However, such stationary points have often been found to be empirically good enough for practical applications, for example, for image restoration (Elad and Aharon, 2006; Mairal et al., 2008b).

Our first result (Proposition 4 below) states that given **(A)**–**(C)**, $f(\mathbf{D}^t)$ converges almost surely and $f(\mathbf{D}^t) - \hat{f}_t(\mathbf{D}^t)$ converges almost surely to 0, meaning that $\hat{f}_t$ acts as a converging surrogate of $f$. First, we prove a lemma to show that $\mathbf{D}^t - \mathbf{D}^{t-1} = O(1/t)$. It does not ensure the convergence of $\mathbf{D}^t$, but guarantees the convergence of the positive sum $\sum_{t=1}^{\infty} \|\mathbf{D}^t - \mathbf{D}^{t-1}\|_{\mathrm{F}}^2$, a classical condition in gradient descent convergence proofs (Bertsekas, 1999).

**Lemma 2** (**Asymptotic variations of $\mathbf{D}_t$.**)
*Assume* ***(A)***–***(C)***. *Then,*

$$\mathbf{D}^{t+1} - \mathbf{D}^t = O\Big(\frac{1}{t}\Big) \quad a.s.$$

The proof of this lemma as well the ones of the subsequent propositions are all given in Appendix B for readability purposes. We can now state and prove our first proposition, which shows that we are indeed minimizing a smooth function.

**Proposition 4** (**Regularity of $f$.**)
*Assume* ***(A)*** *to* ***(C)***. *For* $\mathbf{x}$ *in the support* $K$ *of the probability distribution* $p$, *and* $\mathbf{D}$ *in the feasible set* $\mathcal{D}$, *let us define*

$$\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}) = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1. \tag{2.10}$$

*Then,*

1. *the function* $\ell$ *defined in Eq. (2.2) is continuously differentiable and*

$$\nabla_{\mathbf{D}} \ell(\mathbf{x}, \mathbf{D}) = -(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}))\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})^\top.$$

2. $f$ *is continuously differentiable and* $\nabla f(\mathbf{D}) = \mathbb{E}_{\mathbf{x}}\big[\nabla_{\mathbf{D}} \ell(\mathbf{x}, \mathbf{D})\big]$;

*3.* $\nabla f(\mathbf{D})$ *is Lipschitz on* $\mathcal{D}$.

Now that we have shown that $f$ is a smooth function, we can state our first result showing that the sequence of functions $\hat{f}_t$ acts asymptotically as a surrogate of $f$ and that $f(\mathbf{D}^t)$ converges almost surely in the following proposition.

**Proposition 5** (**Convergence of $f(\mathbf{D}^t)$ and of the surrogate function.**)
*Let* $\hat{f}_t$ *denote the surrogate function defined in Eq. (2.4). Assume* ***(A)*** *to* ***(C)****. Then,*

1. $\hat{f}_t(\mathbf{D}^t)$ *converges almost surely;*
2. $f(\mathbf{D}^t) - \hat{f}_t(\mathbf{D}^t)$ *converges almost surely to* 0*;*
3. $f(\mathbf{D}^t)$ *converges almost surely.*

With Proposition 5 in hand, we can now prove our final and strongest result, namely that first-order necessary optimality conditions are verified asymptotically with probability one.

**Proposition 6** (**Convergence to a stationary point.**)
*Under assumptions* ***(A)*** *to* ***(C)****, the distance between* $\mathbf{D}^t$ *and the set of stationary points of the dictionary learning problem converges almost surely to* 0 *when* $t$ *tends to infinity.*

## 2.5 Extensions to Matrix Factorization

In this section, we present variations of the basic online algorithm to address different optimization problems. We first present different possible regularization terms for the coefficients $\boldsymbol{\alpha}$ and $\mathbf{D}$, which can be used with our algorithm, and then detail some specific cases such as non-negative matrix factorization, sparse principal component analysis, constrained sparse coding, and simultaneous sparse coding.

### 2.5.1 Using Different Regularizers for $\boldsymbol{\alpha}$

In various applications, different priors for the coefficients $\boldsymbol{\alpha}$ may lead to different regularizers $\Omega(\boldsymbol{\alpha})$. As long as the assumptions of Section 2.4.1 are verified, our algorithm can be used with:

- Positivity constraints on $\boldsymbol{\alpha}$ that are added to the $\ell_1$-regularization. The homotopy method presented by Efron et al. (2004) is able to handle such constraints.
- The Tikhonov regularization, $\Omega(\boldsymbol{\alpha}) = \frac{\lambda_1}{2}\|\boldsymbol{\alpha}\|_2^2$, which does not lead to sparse solutions.
- The elastic net (Zou and Hastie, 2005), $\Omega(\boldsymbol{\alpha}) = \lambda_1\|\boldsymbol{\alpha}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}\|_2^2$, leading to a formulation relatively close to Zou et al. (2006).
- The group Lasso (Yuan and Lin, 2006; Turlach et al., 2005; Bach, 2008), $\Omega(\boldsymbol{\alpha}) = \sum_{i=1}^{s}\|\boldsymbol{\alpha}^i\|_2$, where $\boldsymbol{\alpha}^i$ is a vector corresponding to a group of variables.

Non-convex regularizers such as the $\ell_0$ pseudo-norm, $\ell_q$ pseudo-norms with $q < 1$ can be used as well. However, as with any classical dictionary learning techniques exploiting non-convex regularizers (e.g., Olshausen and Field, 1997; Engan et al., 1999; Aharon et al., 2006), there is no theoretical convergence results in these cases. Note also that convex smooth approximation of sparse regularizers (Bradley and Bagnell, 2009), or structured sparsity-inducing regularizers (Jenatton et al., 2009; Jacob et al., 2009) could be used as well even though we have not tested them.

### 2.5.2 Using Different Constraint Sets for D

In the previous subsection, we have claimed that our algorithm could be used with different regularization terms on $\boldsymbol{\alpha}$. For the dictionary learning problem, we have considered an $\ell_2$-regularization on $\mathbf{D}$ by forcing its columns to have less than unit $\ell_2$-norm. We have shown that with this constraint set, the dictionary update step can be solved efficiently using a block-coordinate descent approach. Updating the $j$-th column of $\mathbf{D}$, when keeping the other ones fixed is solved by orthogonally projecting the vector $\mathbf{u}^j = \mathbf{d}^j + (1/\mathbf{B}_{jj})(\mathbf{c}^j - \mathbf{D}\mathbf{b}^j)$ on the constraint set $\mathcal{D}$, which in the classical dictionary learning case amounts to a projection of $\mathbf{u}^j$ on the $\ell_2$-ball.

It is easy to show that this procedure can be extended to different convex constraint sets $\mathcal{D}'$ as long as the constraints are a union of independent constraints on each column of $\mathbf{D}$ and the orthogonal projections of the vectors $\mathbf{u}^j$ onto the set $\mathcal{D}'$ can be done efficiently. Examples of different sets $\mathcal{D}'$ that we propose as an alternative to $\mathcal{D}$ are

- The "non-negative" constraints:

$$\mathcal{D}' \triangleq \{\mathbf{D} \in \mathbb{R}^{m \times p} \text{ s.t. } \forall j \in [\![1; p]\!], \quad \|\mathbf{d}^j\|_2 \leq 1 \text{ and } \mathbf{d}^j \geq 0\}.$$

- The "elastic-net" constraints:

$$\mathcal{D}' \triangleq \{\mathbf{D} \in \mathbb{R}^{m \times p} \text{ s.t. } \forall j \in [\![1; p]\!], \quad \|\mathbf{d}^j\|_2^2 + \gamma \|\mathbf{d}^j\|_1 \leq 1\}.$$

  These constraints induce sparsity in the dictionary $\mathbf{D}$ (in addition to the sparsity-inducing regularizer on the vectors $\boldsymbol{\alpha}^i$). By analogy with the regularization proposed by Zou and Hastie (2005), we call these constraints "elastic-net constraints." Here, $\gamma$ is a new parameter, controlling the sparsity of the dictionary $\mathbf{D}$. Adding a non-negativity constraint is also possible in this case. Note that the presence of the $\ell_2$ regularization is important here. It has been shown by Bach et al. (2008) that using the $\ell_1$-norm only in such problems lead to trivial solutions when $p$ is large enough. The combination of $\ell_1$ and $\ell_2$ constraints has also been proposed recently for the problem of matrix factorization by Witten et al. (2009), but in a slightly different setting.

- The "fused lasso" (Tibshirani et al., 2005) constraints. When one is looking for a dictionary whose columns are sparse and piecewise-constant, a fused lasso regularization can be used. For a vector $\mathbf{u}$ in $\mathbb{R}^m$, we consider the $\ell_1$-norm of the

consecutive differences of $\mathbf{u}$ denoted by

$$\mathrm{FL}(\mathbf{u}) \triangleq \sum_{i=2}^{m} |\mathbf{u}_i - \mathbf{u}_{i-1}|,$$

and define the "fused lasso" constraint set

$$\mathcal{D}' \triangleq \{\mathbf{D} \in \mathbb{R}^{m \times p} \quad \text{s.t.} \quad \forall j \in [\![1; p]\!], \quad \|\mathbf{d}^j\|_2^2 + \gamma_1 \|\mathbf{d}^j\|_1 + \gamma_2 \,\mathrm{FL}(\mathbf{d}^j) \leq 1\}.$$

This kind of regularization has proven to be useful for exploiting genomic data such as CGH arrays (Tibshirani and Wang, 2008).

In all these settings, replacing the projections of the vectors $\mathbf{u}^j$ onto the $\ell_2$-ball by the projections onto the new constraints, our algorithm is still guaranteed to converge and find a stationary point of the optimization problem. The orthogonal projection onto the "non negative" ball is simple (additional thresholding) but the projection onto the two other sets is slightly more involved. In Appendix C, we propose two algorithms for efficiently solving these problems. The first one is presented in Section C.1 and computes the projection of a vector onto the elastic-net constraint in linear time, by extending the efficient projection onto the $\ell_1$-ball from Maculan and de Paula (1989) and Duchi et al. (2008). The second one is a homotopy method, which solves the projection on the fused lasso constraint set in $O(ps)$, where $s$ is the number of piecewise-constant parts in the solution. This method also solves efficiently the fused lasso signal approximation problem presented in Friedman et al. (2007):

$$\min_{\mathbf{u} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{b} - \mathbf{u}\|_2^2 + \gamma_1 \|\mathbf{u}\|_1 + \gamma_2 \,\mathrm{FL}(\mathbf{u}) + \gamma_3 \|\mathbf{u}\|_2^2.$$

Being able to solve this problem efficiently has also numerous applications, which are beyond the scope of this work. For instance, it allows us to use the fast algorithm of Nesterov (2007) for solving the more general fused lasso problem (Tibshirani et al., 2005). Note that the proposed method could be used as well with more complex constraints for the columns of $\mathbf{D}$, which we have not tested in this work, addressing for instance the problem of structured sparse PCA (Jenatton et al., 2010c).

Now that we have presented a few possible regularizers for $\boldsymbol{\alpha}$ and $\mathbf{D}$, that can be used within our algorithm, we focus on a few classical problems which can be formulated as dictionary learning problems with specific combinations of such regularizers.

### 2.5.3 Non Negative Matrix Factorization

Given a matrix $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ in $\mathbb{R}^{m \times n}$, Lee and Seung (2001) have proposed the non negative matrix factorization problem (NMF), which consists of minimizing the following cost

$$\min_{\mathbf{D} \in \mathcal{D}, \mathbf{A} \in \mathbb{R}^{p \times n}} \sum_{i=1}^{n} \left[\frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2\right] \quad \text{s.t.} \quad \mathbf{D} \geq 0, \ \forall\, i, \quad \boldsymbol{\alpha}^i \geq 0,$$

where we recall that $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n]$. With this formulation, the matrix $\mathbf{D}$ and the vectors $\boldsymbol{\alpha}^i$ are forced to have non negative components, which leads to sparse solutions. When applied to images, such as faces, Lee and Seung (2001) have shown that the learned features are more localized than the ones learned with a classical singular value decomposition. As for dictionary learning, classical approaches for addressing this problem are batch algorithms, such as the multiplicative update rules of Lee and Seung (2001), or the projected gradient descent algorithm of Lin (2007).

Following this line of research, Hoyer (2002, 2004) has proposed non negative sparse coding (NNSC), which extends non-negative matrix factorization by adding a sparsity-inducing penalty to the objective function to further control the sparsity of the vectors $\boldsymbol{\alpha}^i$:

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{A}\in\mathbb{R}^{p\times n}} \sum_{i=1}^{n} \left[\frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda \sum_{j=1}^{p} \boldsymbol{\alpha}_j^i\right] \quad \text{s.t.} \quad \mathbf{D} \geq 0, \ \forall\, i \in [\![1; n]\!], \quad \boldsymbol{\alpha}^i \geq 0.$$

When $\lambda = 0$, this formulation is equivalent to NMF. The only difference with the dictionary learning problem is that non-negativity constraints are imposed on $\mathbf{D}$ and the vectors $\boldsymbol{\alpha}^i$. A simple modification of our algorithm, presented above, allows us to handle these constraints, while guaranteeing to find a stationary point of the optimization problem. Moreover, our approach can work in the setting when $n$ is large.

### 2.5.4 Sparse Principal Component Analysis

Principal component analysis (PCA) is a classical tool for data analysis, which can be interpreted as a method for finding orthogonal directions maximizing the variance of the data, or as a low-rank matrix approximation method. Jolliffe et al. (2003), Zou et al. (2006), d'Aspremont et al. (2007), d'Aspremont et al. (2008), Witten et al. (2009) and Zass and Shashua (2007) have proposed different formulations for sparse principal component analysis (SPCA), which extends PCA by estimating sparse vectors maximizing the variance of the data, some of these formulations enforcing orthogonality between the sparse components, whereas some do not. In this work, we formulate SPCA as a sparse matrix factorization which is equivalent to the dictionary learning problem with possibly sparsity constraints on the dictionary—that is, we use the $\ell_1$-regularization term for $\boldsymbol{\alpha}$ and the "elastic-net" constraint for $\mathbf{D}$ (as used in a penalty term by Zou et al. 2006):

$$\min_{\mathbf{D}\in\mathbb{R}^{m\times p},\mathbf{A}\in\mathbb{R}^{p\times n}} \sum_{i=1}^{n} \left[\frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda\|\boldsymbol{\alpha}^i\|_1\right] \quad \text{s.t.} \quad \forall j \in [\![1; p]\!], \quad \|\mathbf{d}^j\|_2^2 + \gamma\|\mathbf{d}^j\|_1 \leq 1.$$

As detailed above, our dictionary update procedure amounts to successive orthogonal projection of the vectors $\mathbf{u}^j$ on the constraint set. More precisely, the update of $\mathbf{d}^j$ becomes

$$\begin{aligned}
\mathbf{u}^j &\leftarrow \frac{1}{\mathbf{B}_{jj}}(\mathbf{c}^j - \mathbf{D}\mathbf{b}^j) + \mathbf{d}^j, \\
\mathbf{d}^j &\leftarrow \arg\min_{\mathbf{d}\in\mathbb{R}^m} \|\mathbf{u}^j - \mathbf{d}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{d}\|_2^2 + \gamma\|\mathbf{d}\|_1 \leq 1,
\end{aligned}$$

which can be solved in linear time using Algorithm 9 presented in Appendix C. In addition to that, our SPCA method can be used with fused Lasso constraints as well.

### 2.5.5 Constrained Sparse Coding

Constrained sparse coding problems are often encountered in the literature, and lead to different loss functions such as

$$\ell'(\mathbf{x}, \mathbf{D}) = \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 \leq T, \tag{2.11}$$

or

$$\ell''(\mathbf{x}, \mathbf{D}) = \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \|\boldsymbol{\alpha}\|_1 \quad \text{s.t.} \quad \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \leq \varepsilon, \tag{2.12}$$

where $T$ and $\varepsilon$ are pre-defined thresholds. Even though these loss functions lead to equivalent optimization problems in the sense that for given $\mathbf{x}, \mathbf{D}$ and $\lambda$, there exist $\varepsilon$ and $T$ such that $\ell(\mathbf{x}, \mathbf{D})$, $\ell'(\mathbf{x}, \mathbf{D})$ and $\ell''(\mathbf{x}, \mathbf{D})$ admit the same solution $\boldsymbol{\alpha}^\star$, the problems of learning $\mathbf{D}$ using $\ell$, $\ell'$ of $\ell''$ are not equivalent. For instance, using $\ell''$ has proven experimentally to be particularly well adapted to image denoising (Elad and Aharon, 2006; Mairal et al., 2008b).

For all $T$, the same analysis as for $\ell$ can be carried for $\ell'$, and the simple modification which consists of computing $\boldsymbol{\alpha}^t$ using Eq. (2.11) in the sparse coding step leads to the minimization of the expected cost $\min_{\mathbf{D} \in \mathbf{C}} \mathbb{E}_{\mathbf{x}}[\ell'(\mathbf{x}, \mathbf{D})]$.

Handling the case $\ell''$ is a bit different. We propose to use the same strategy as for $\ell'$—that is, using our algorithm but computing $\boldsymbol{\alpha}^t$ solving Eq. (2.12). Even though our analysis does not apply since we do not have a quadratic surrogate of the expected cost, experimental evidence shows that this approach is efficient in practice.

### 2.5.6 Simultaneous Sparse Coding

In some situations, the signals $\mathbf{x}^i$ are not i.i.d samples of an unknown probability distribution, but are structured in groups (which are however independent from each other), and one may want to address the problem of simultaneous sparse coding, which appears also in the literature under various names such as group sparsity or grouped variable selection (Cotter et al., 2005; Turlach et al., 2005; Yuan and Lin, 2006; Obozinski et al., 2009, 2008; Zhang et al., 2008; Tropp et al., 2006; Tropp, 2006). Let $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^q] \in \mathbb{R}^{m \times q}$ be a set of signals. Suppose one wants to obtain sparse decompositions of the signals on the dictionary $\mathbf{D}$ that share the same active set (non-zero coefficients). Let $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^q]$ in $\mathbb{R}^{p \times q}$ be the matrix composed of the coefficients. One way of imposing this *joint sparsity* is to penalize the number of non-zero rows of $\boldsymbol{\alpha}$. A classical convex relaxation of this joint sparsity measure is to consider the $\ell_{1,2}$-norm on the matrix $\boldsymbol{\alpha}$

$$\|\mathbf{A}\|_{1,2} \triangleq \sum_{j=1}^{p} \|\mathbf{A}_j\|_2,$$

where $\mathbf{A}_j$ is the $j$-th row of $\mathbf{A}$. In that setting, the $\ell_{1,2}$-norm of $\mathbf{A}$ is the $\ell_1$-norm of the $\ell_2$-norm of the rows of $\mathbf{A}$.

The problem of jointly decomposing the signals $\mathbf{x}^i$ can be written as a $\ell_{1,2}$-sparse decomposition problem, which is a subcase of the group Lasso (Turlach et al., 2005;

Yuan and Lin, 2006; Bach, 2008), by defining the cost function

$$\ell'''(\mathbf{X}, \mathbf{D}) = \min_{\mathbf{A} \in \mathbb{R}^{p \times q}} \frac{1}{2} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_{\mathrm{F}}^2 + \lambda \|\mathbf{A}\|_{1,2},$$

which can be computed using a block-coordinate descent approach (Friedman et al., 2007) or an active set method (Roth and Fischer, 2008).

Suppose now that we are able to draw groups of signals $\mathbf{X}^i$, $i = 1, \ldots, n$ which have bounded size and are independent from each other and identically distributed, one can learn an adapted dictionary by solving the optimization problem

$$\min_{\mathbf{D} \in \mathcal{D}} \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \ell'''(\mathbf{X}^i, \mathbf{D}).$$

Being able to solve this optimization problem is important for many applications. For instance, in Mairal et al. (2009c), state-of-the-art results in image denoising and demosaicking are achieved with this formulation. The extension of our algorithm to this case is relatively easy, computing at each sparse coding step a matrix of coefficients $\mathbf{A}$, and keeping the updates of $\mathbf{B}^t$ and $\mathbf{C}^t$ unchanged.

All of the variants of this section have been implemented. Next section evaluates some of them experimentally. An efficient C++ implementation with a Matlab interface of these variants is available on the Willow project-team web page.[4]

## 2.6 Experimental Validation

In this section, we present experiments on natural images and genomic data to demonstrate the efficiency of our method for dictionary learning, non-negative matrix factorization, and sparse principal component analysis.

### 2.6.1 Performance Evaluation for Dictionary Learning

For our experiments, we have randomly selected $1.25 \times 10^6$ patches from images in the Pascal VOC'06 image database (Everingham et al., 2007), which is composed of varied natural images; $10^6$ of these are kept for training, and the rest for testing. We used these patches to create three data sets $A$, $B$, and $C$ with increasing patch and dictionary sizes representing various settings which are typical in image processing applications: We have

| Data set | Signal size $m$ | Nb $p$ of atoms | Type |
|---|---|---|---|
| $A$ | $8 \times 8 = 64$ | 256 | b&w |
| $B$ | $12 \times 12 \times 3 = 432$ | 512 | color |
| $C$ | $16 \times 16 = 256$ | 1024 | b&w |

centered and normalized the patches to have unit $\ell_2$-norm and used the regularization

[4] http://www.di.ens.fr/willow/SPAMS/

parameter $\lambda = 1.2/\sqrt{m}$ in all of our experiments. The $1/\sqrt{m}$ term is a classical normalization factor (Bickel et al., 2009), and the constant 1.2 has shown to yield about 10 nonzero coefficients for data set A and 40 for data sets B and C in these experiments. We have implemented the proposed algorithm in C++ with a Matlab interface. All the results presented in this section use the refinements from Section 2.3.4 since this has lead empirically to speed improvements. Although our implementation is multithreaded, our experiments have been run for simplicity on a single-CPU, single-core 2.66Ghz machine.

The first parameter to tune is $\eta$, the number of signals drawn at each iteration. Trying different powers of 2 for this variable has shown that $\eta = 512$ was a good choice (lowest objective function values on the training set—empirically, this setting also yields the lowest values on the test set). Even though this parameter is fairly easy to tune since values of 64, 128, 256 and 1024 have given very similar performances, the difference with the choice $\eta = 1$ is significant.

Our implementation can be used in both the online setting it is intended for, and in a regular batch mode where it uses the entire data set at each iteration. We have also implemented a first-order stochastic gradient descent algorithm that shares most of its code with our algorithm, except for the dictionary update step. This setting allows us to draw meaningful comparisons between our algorithm and its batch and stochastic gradient alternatives, which would have been difficult otherwise. For example, comparing our algorithm to the Matlab implementation of the batch approach from Lee et al. (2007) developed by its authors would have been unfair since our C++ program has a built-in speed advantage.[5] To measure and compare the performances of the three tested methods, we have plotted the value of the objective function on *the test set*, acting as a surrogate of the expected cost, as a function of the corresponding training time.

### Online vs. Batch

The left column of Figure 2.1 compares the online and batch settings of our implementation. The full training set consists of $10^6$ samples. The online version of our algorithm draws samples from the entire set, and we have run its batch version on the full data set as well as subsets of size $10^4$ and $10^5$ (see Figure 2.1). The online setting systematically outperforms its batch counterpart for every training set size and desired precision. We use a logarithmic scale for the computation time, which shows that in many situations, the difference in performance can be dramatic. Similar experiments have given similar results on smaller data sets. Our algorithm uses all the speed-ups from Section 2.3.4. The parameter $\rho$ was chosen by trying the values $0, 5, 10, 15, 20, 25$, and $t_0$ by trying different powers of 10. We have selected $(t_0 = 0.001, \rho = 15)$, which has given the best performance in terms of objective function evaluated on the training set for the three data sets. We have plotted three curves for our method: OL1 corresponds to the optimal

[5]Both LARS and the feature-sign algorithm (Lee et al., 2007) require a large number of low-level operations which are not well optimized in Matlab. We have indeed observed that our C++ implementation of LARS is up to 50 times faster than the Matlab implementation of the feature-sign algorithm of Lee et al. (2007) for our experiments.

setting ($t_0 = 0.001, \rho = 15$). Even though tuning two parameters might seem cumbersome, we have plotted two other curves showing that, on the contrary, our method is very easy to use. The curve OL2, corresponding to the setting ($t_0 = 0.001, \rho = 10$), is very difficult to distinguish from the first curve and we have observed a similar behavior with the setting ($t_0 = 0.001, \rho = 20$). showing that our method is *robust to the choice of the parameter* $\rho$. We have also observed that the parameter $\rho$ is useful for large data sets only. When using smaller ones ($n \leq 100,000$), it did not bring any benefit.

Moreover, the curve OL3 is obtained without using a tuned parameter $t_0$—that is, $\rho = 15$ and $t_0 = 0$, and shows that its influence is very limited since very good results are obtained without using it. On the other hand, we have observed that using a parameter $t_0$ too big, could slightly slow down our algorithm during the first epoch (cycle on the training set).

### Comparison with Stochastic Gradient Descent

Our experiments have shown that obtaining good performance with stochastic gradient descent requires using both the mini-batch heuristic *and* carefully choosing a learning rate of the form $a/(\eta t + b)$. To give the fairest comparison possible, we have thus optimized these parameters. As for our algorithm, sampling $\eta$ values among powers of 2 (as before) has shown that $\eta = 512$ was a good value and gives a significant better performance than $\eta = 1$.

In an earlier version of this work (Mairal et al., 2009a), we have proposed a strategy for our method which does not require any parameter tuning except the mini-batch $\eta$ and compared it with the stochastic gradient descent algorithm (SGD) with a learning rate of the form $a/(\eta t)$. While our method has improved in performance using the new parameter $\rho$, SGD has also proven to provide much better results when using a learning rate of the form $a/(\eta t + b)$ instead of $a/(\eta t)$, at the cost of an extra parameter $b$ to tune. Using the learning rate $a/(\eta t)$ with a high value for $a$ results indeed in too large initial steps of the algorithm increasing dramatically the value of the objective function, and a small value of $a$ leads to bad asymptotic results, while a learning rate of the form $a/(\eta t + b)$ is a good compromise.

We have tried different powers of 10 for $a$ and $b$. First selected the couple ($a = 100,000, b = 100,000$) and then refined it, trying the values $100,000 \times 2^i$ for $i = -3, \ldots, 3$. Finally, we have selected ($a = 200,000, b = 400,000$). As shown on the right column of Figure 2.1, this setting represented by the curve SG1 leads to similar results as our method. The curve SG2 corresponds to the parameters ($a = 400,000, b = 400,000$) and shows that increasing slightly the parameter $a$ makes the curves worse than the others during the first iterations (see for instance the curve between 1 and $10^2$ seconds for data set A), but still lead to good asymptotic results. The curve SG3 corresponds to a situation where $a$ and $b$ are slightly too small ($a = 50,000, b = 100,000$). It is as good as SG1 for data sets A and B, but asymptotically slightly below the others for data set C. All the curves are obtained as the average of three experiments with different initializations. Interestingly, even though the problem is not convex, the different initializations have

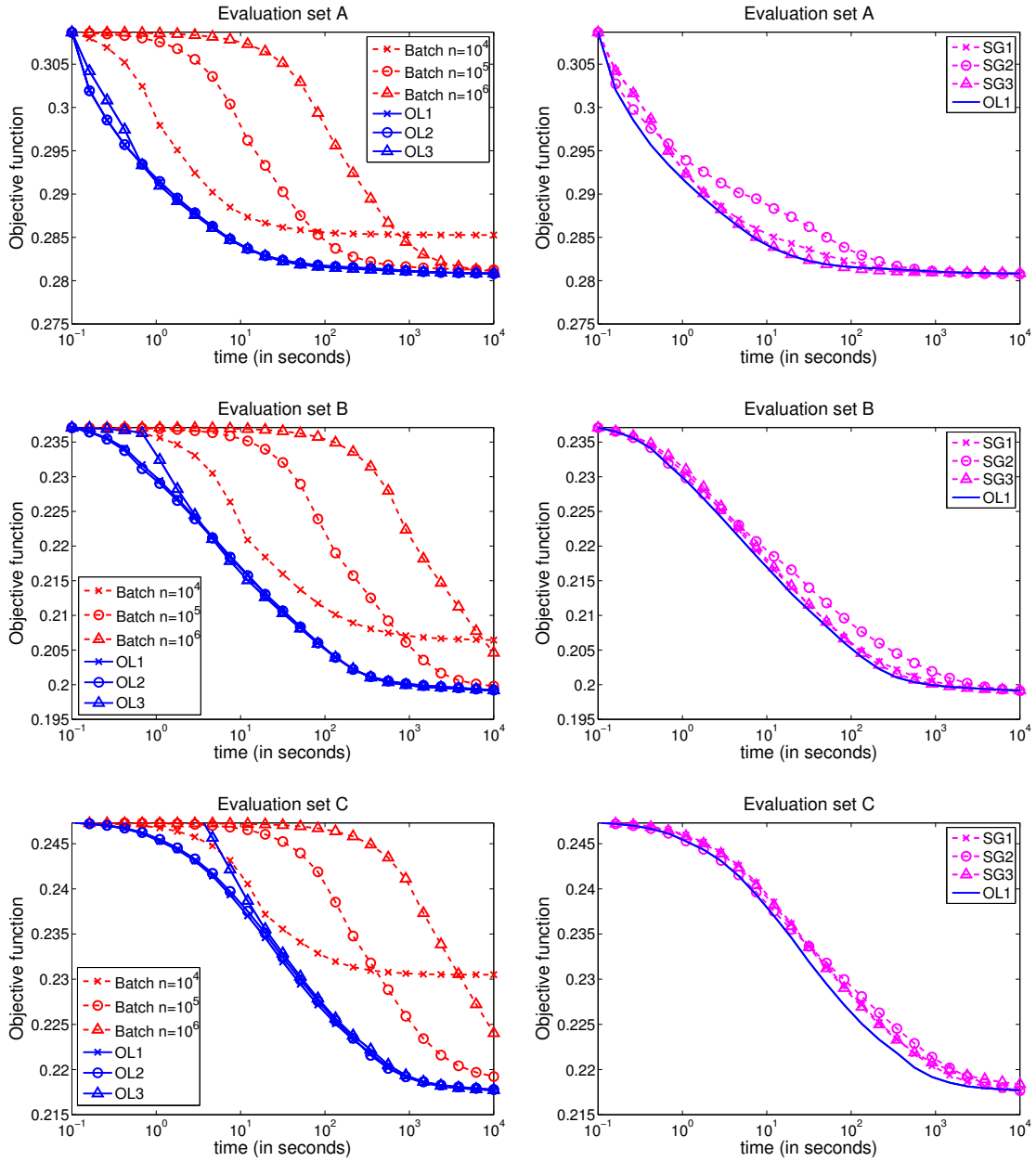



Figure 2.1: Left: Comparison between our method and the batch approach for dictionary learning. Right: Comparison between our method and stochastic gradient descent. The results are reported for three data sets as a function of computation time on a logarithmic scale. Note that the times of computation that are less than $0.1s$ are not reported. See text for details.

led to very similar values of the objective function and the variance of the experiments was always insignificant after 10 seconds of computations.

### 2.6.2 Non Negative Matrix Factorization and Non Negative Sparse Coding

In this section, we compare our method with the classical algorithm of Lee and Seung (2001) for NMF and the non-negative sparse coding algorithm of Hoyer (2002) for NNSC. The experiments have been carried out on three data sets with different sizes:

- Data set D is composed of $n = 2,429$ face images of size $m = 19 \times 19$ pixels from the the MIT-CBCL Face Database #1 (Sung, 1996).
- Data set E is composed of $n = 2,414$ face images of size $m = 192 \times 168$ pixels from the Extended Yale B Database (Georghiades et al., 2001; Lee et al., 2005).
- Data set F is composed of $n = 100,000$ natural image patches of size $m = 16 \times 16$ pixels from the Pascal VOC'06 image database (Everingham et al., 2007).

We have used the Matlab implementations of NMF and NNSC of P. Hoyer, which are freely available at `http://www.cs.helsinki.fi/u/phoyer/software.html`. Even though our C++ implementation has a built-in advantage in terms of speed over these Matlab implementations, most of the computational time of NMF and NNSC is spent on large matrix multiplications, which are typically well optimized in Matlab. All the experiments have been run for simplicity on a single-CPU, single-core 2.4GHz machine, without using the parameters $\rho$ and $t_0$ presented in Section 2.3.4—that is, $\rho = 0$ and $t_0 = 0$. As in Section 2.6.1, a minibatch of size $\eta = 512$ is chosen. Following the original experiment of Lee and Seung (2001) on data set D, we have chosen to learn $p = 49$ basis vectors for the face images data sets D and E, and we have chosen $p = 64$ for data set F. Each input vector is normalized to have unit $\ell_2$-norm.

The experiments we present in this section compare the value of the objective function on the data sets obtained with the different algorithms as a function of the computation time. Since our algorithm learns the matrix $\mathbf{D}$ but does not provide the matrix $\boldsymbol{\alpha}$, the computation times reported for our approach include two steps: First, we run our algorithm to obtain $\mathbf{D}$. Second, we run one sparse coding step over all the input vectors to obtain $\boldsymbol{\alpha}$. Figure 2.2 presents the results for NMF and NNSC. The gradient step for the algorithm of Hoyer (2002) was optimized for the best performance and $\lambda$ was set to $\frac{1}{\sqrt{m}}$. Both $\mathbf{D}$ and $\boldsymbol{\alpha}$ were initialized randomly. The values reported are those obtained for more than 0.1s of computation. Since the random initialization provides an objective value which is by far greater than the value obtained at convergence, the curves are all truncated to present significant objective values. All the results are obtained using the average of 3 experiments with different initializations. As shown on Figure 2.2, our algorithm provides a significant improvement in terms of speed compared to the other tested methods, even though the results for NMF and NNSC could be improved a bit using a C++ implementation.

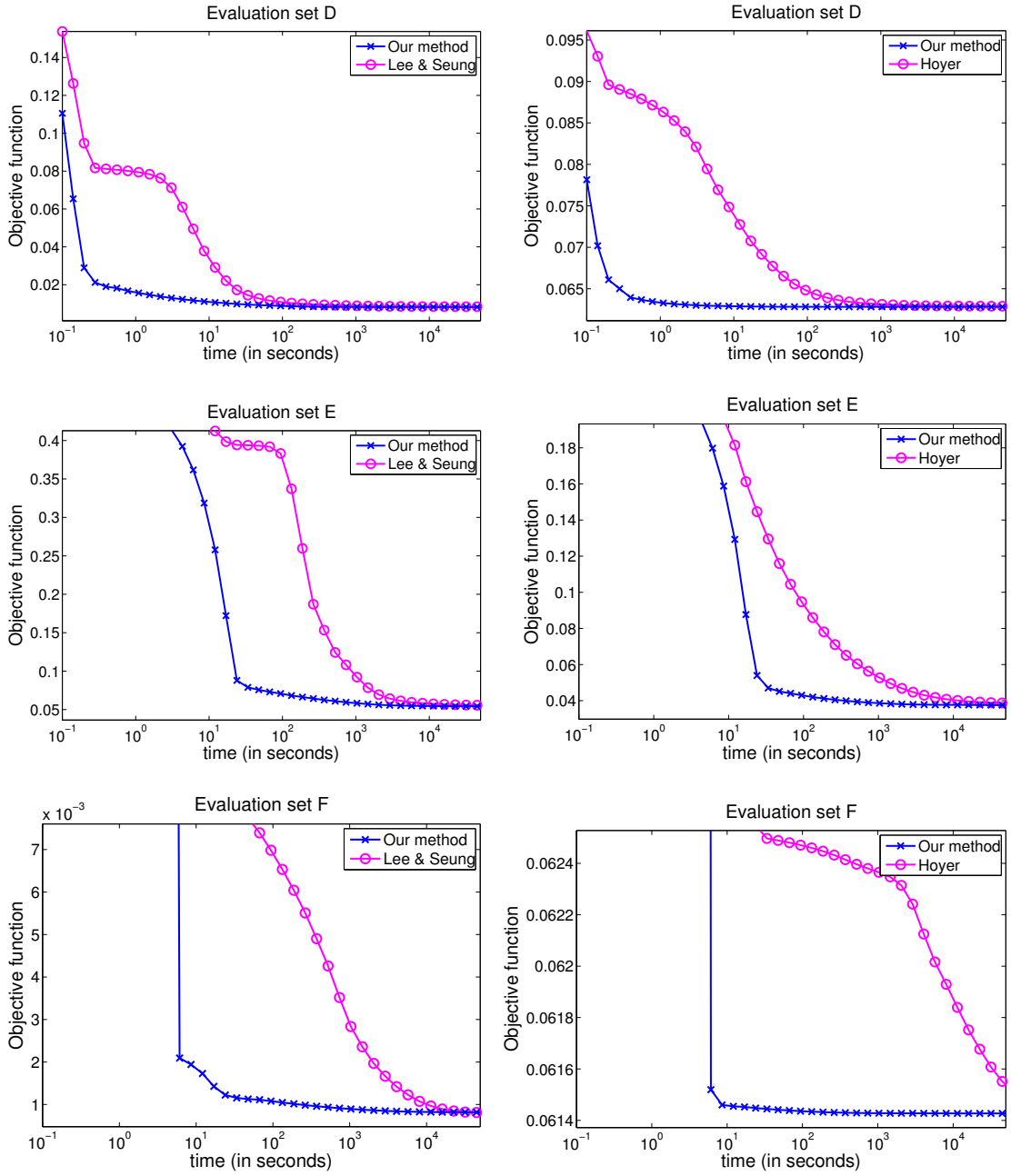



Figure 2.2: Left: Comparison between our method and the approach of Lee and Seung (2001) for NMF. Right: Comparison between our method and the approach of Hoyer (2002) for NNSC. The value of the objective function is reported for three data sets as a function of computation time on a logarithmic scale.

### 2.6.3 Sparse Principal Component Analysis

We present here the application of our method addressing SPCA with various types of data: faces, natural image patches, and genomic data.

#### Faces and Natural Patches

In this section, we compare qualitatively the results obtained by PCA, NMF, our dictionary learning and our sparse principal component analysis algorithm on the data sets used in Section 2.6.2. For dictionary learning, PCA and SPCA, the input vectors are first centered and normalized to have a unit norm. Visual results are presented on figures 2.3, 2.4 and 2.5, respectively for the data sets D, E and F. The parameter $\lambda$ for dictionary learning and SPCA was set so that the decomposition of each input signal has approximately 10 nonzero coefficients. The results for SPCA are presented for various values of the parameter $\gamma$, yielding different levels of sparsity. The scalar $\tau$ indicates the percentage of nonzero values of the dictionary.

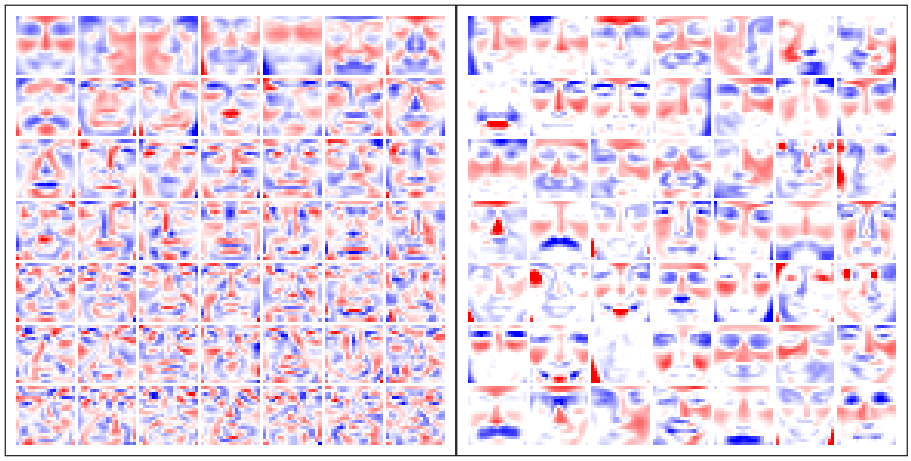
Each image is composed of $p$ small images each representing one learned feature vector. Negative values are blue, positive values are red and the zero values are represented in white. Confirming earlier observations from Lee and Seung (2001), PCA systematically produces features spread out over the images, whereas NMF produces more localized features on the face databases D and E. However, neither PCA, nor NMF are able to learn localized features on the set of natural patches F. On the other hand, the dictionary learning technique is able to learn localized features on data set F, and SPCA is the only tested method that allows controlling the level of sparsity among the learned matrices.

#### Genomic Data

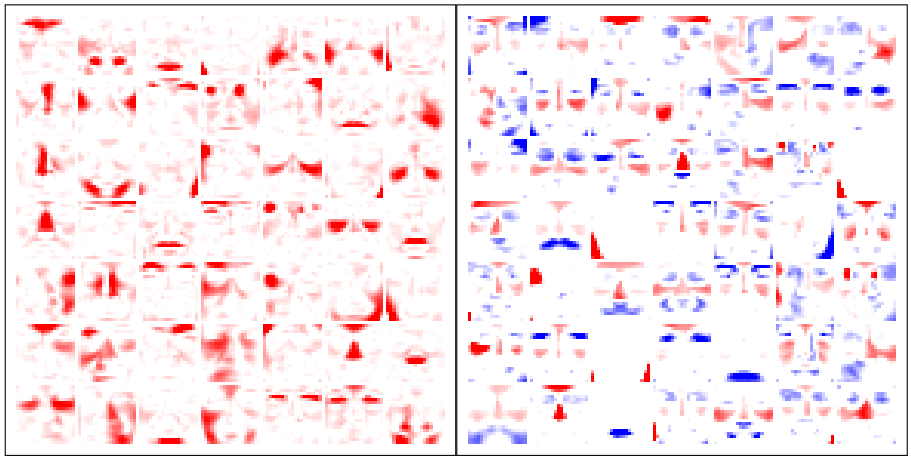
This experiment follows Witten et al. (2009) and demonstrates that our matrix decomposition technique can be used for analyzing genomic data. Gene expression measurements and DNA copy number changes (comparative genomic hybridization CGH) are two popular types of data in genomic research, which can be used to characterize a set of abnormal tissue samples for instance. When these two types of data are available, a recent line of research tries to analyze the correlation between them—that is, to determine sets of expression genes which are correlated with sets of chromosomal gains or losses (see Witten et al., 2009 and references therein). Let us suppose that for $n$ tissue samples, we have a matrix $\mathbf{X}$ in $\mathbb{R}^{n\times p}$ of gene expression measurements and a matrix $\mathbf{Y}$ in $\mathbb{R}^{n\times q}$ of CGH measurements. In order to analyze the correlation between these two sets of data, recent works have suggested the use of canonical correlation analysis (Hotelling, 1936), which solves[6]

$$\min_{\mathbf{u}\in\mathbb{R}^p,\mathbf{v}\in\mathbb{R}^q} \operatorname{cov}(\mathbf{Xu},\mathbf{Yv}) \quad \text{s.t.} \quad \|\mathbf{Xu}\|_2 \leq 1 \quad \text{and} \quad \|\mathbf{Yv}\|_2 \leq 1.$$

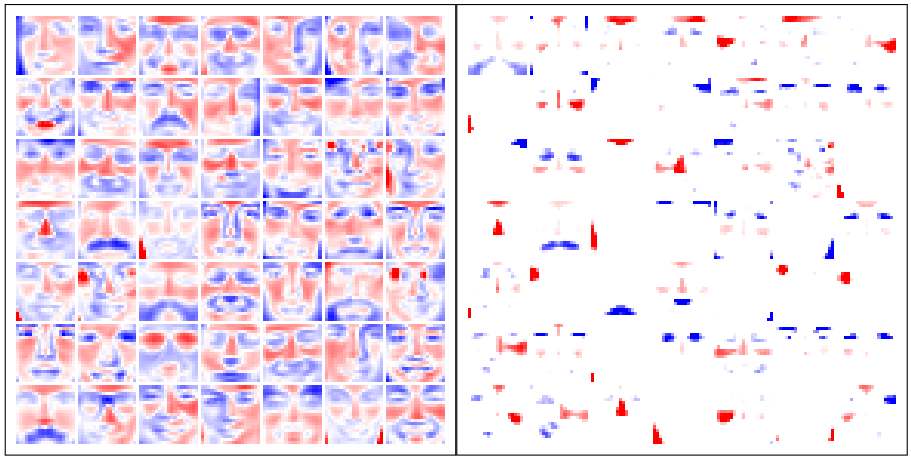
[6]Note that when more than one couple of factors are needed, two sequences $\mathbf{u}^1,\mathbf{u}^2,\ldots$ and $\mathbf{v}^1,\mathbf{v}^2,\ldots$ of factors can be obtained recursively subject to orthogonality constraints of the sequences $\mathbf{Xu}^1,\mathbf{Xu}^2,\ldots$ and $\mathbf{Yv}^1,\mathbf{Yv}^2,\ldots$.



(a) PCA (b) SPCA, $\tau = 70\%$

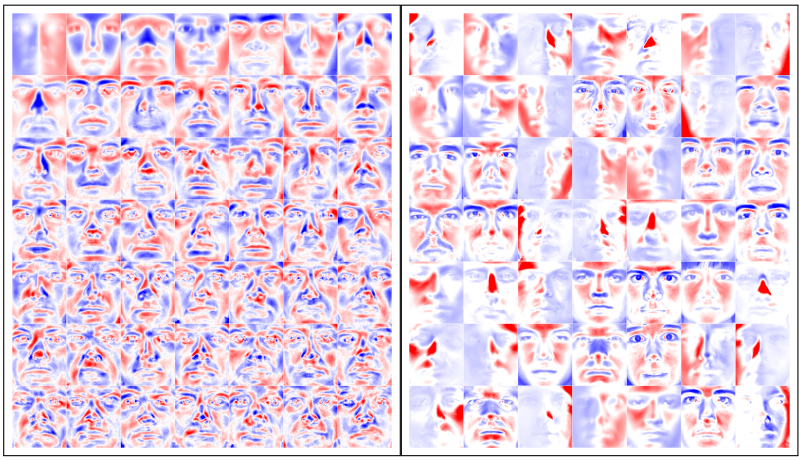


(c) NMF (d) SPCA, $\tau = 30\%$

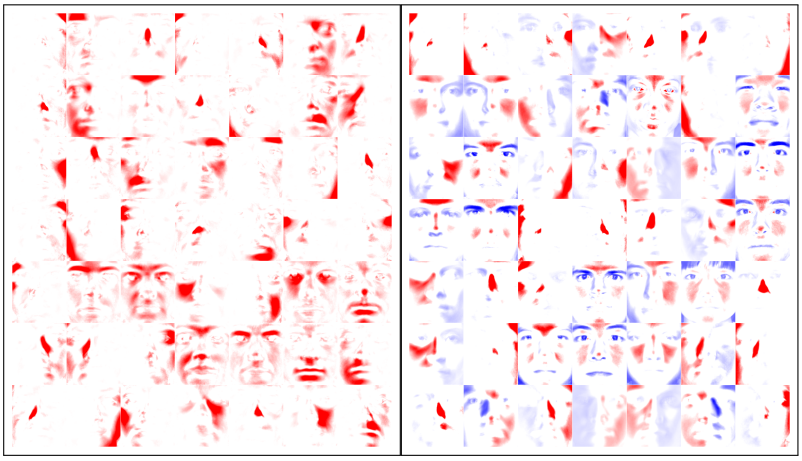


(e) Dictionary Learning (f) SPCA, $\tau = 10\%$

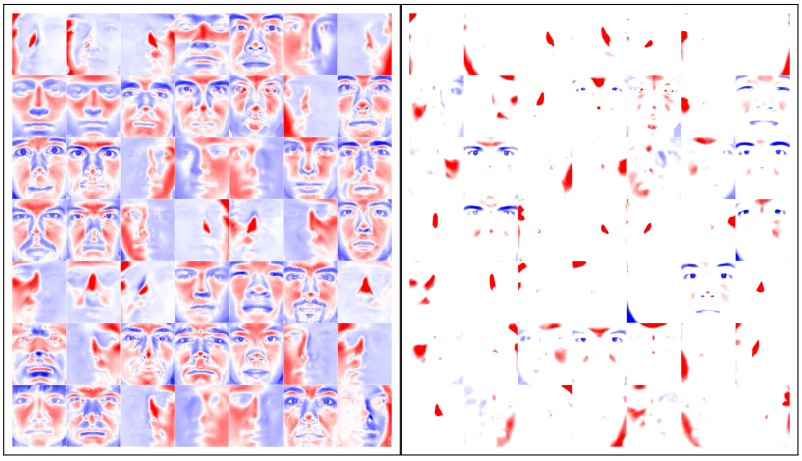
Figure 2.3: Results obtained by PCA, NMF, dictionary learning, SPCA for data set D.



(a) PCA (b) SPCA, $\tau = 70\%$

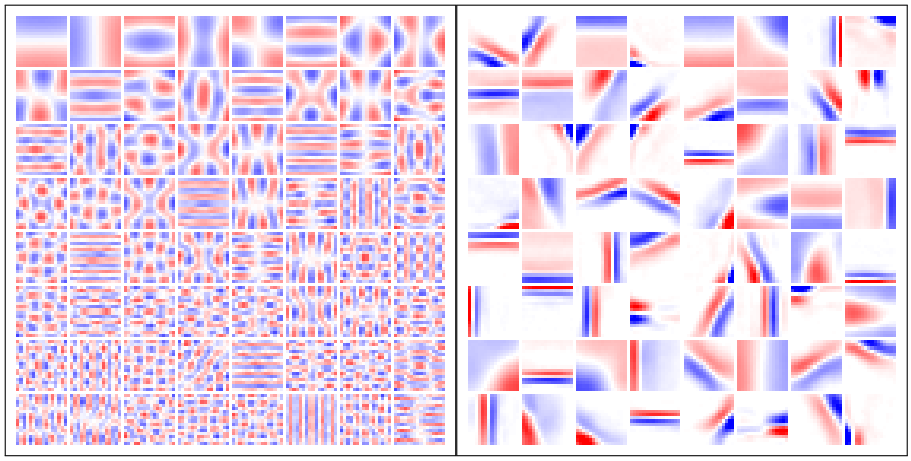


(c) NMF (d) SPCA, $\tau = 30\%$

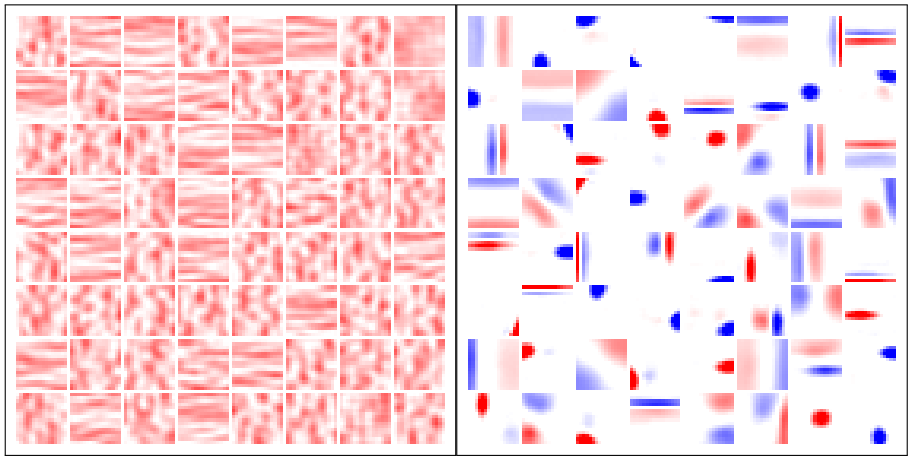


(e) Dictionary Learning (f) SPCA, $\tau = 10\%$

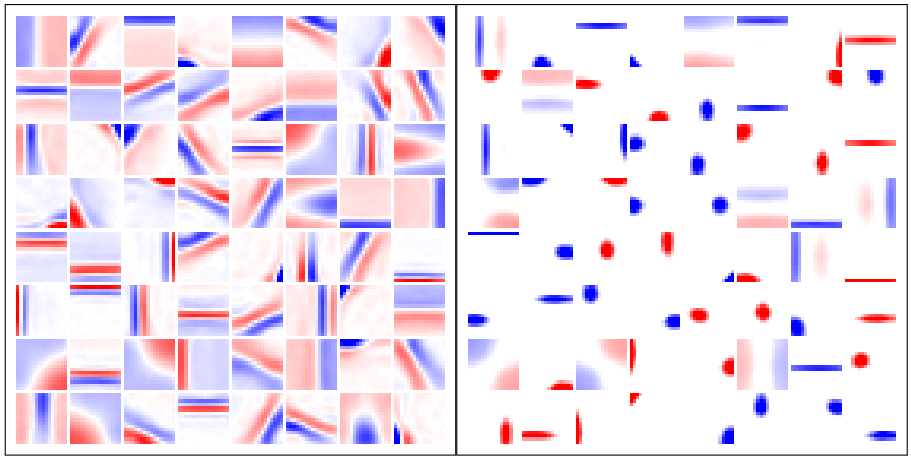
Figure 2.4: Results obtained by PCA, NMF, dictionary learning, SPCA for data set E.



(a) PCA (b) SPCA, $\tau = 70\%$



(c) NMF (d) SPCA, $\tau = 30\%$



(e) Dictionary Learning (f) SPCA, $\tau = 10\%$

Figure 2.5: Results obtained by PCA, NMF, dictionary learning, SPCA for data set F.

When $\mathbf{X}$ and $\mathbf{Y}$ are centered and normalized, it has been further shown that with this type of data, good results can be obtained by treating the covariance matrices $\mathbf{X}^\top\mathbf{X}$ and $\mathbf{Y}^\top\mathbf{Y}$ as diagonal, leading to a rank-one matrix decomposition problem

$$\min_{\mathbf{u}\in\mathbb{R}^p,\mathbf{v}\in\mathbb{R}^q} \|\mathbf{X}^\top\mathbf{Y} - \mathbf{u}\mathbf{v}^\top\|_{\mathrm{F}}^2 \text{ s.t. } \|\mathbf{u}\|_2 \leq 1, \text{ and } \|\mathbf{v}\|_2 \leq 1.$$

Furthermore, as shown by Witten et al. (2009), this method can benefit from sparse regularizers such as the $\ell_1$ norm for the gene expression measurements and a fused lasso for the CGH arrays, which are classical choices used for these data. The formulation we have chosen is slightly different from the one used by Witten et al. (2009) and can be addressed using our algorithm:

$$\min_{\mathbf{u}\in\mathbb{R}^p,\mathbf{v}\in\mathbb{R}^q} \|\mathbf{Y}^\top\mathbf{X} - \mathbf{v}\mathbf{u}^\top\|_{\mathrm{F}}^2 + \lambda\|\mathbf{u}\|_2 \text{ s.t. } \|\mathbf{v}\|_2^2 + \gamma_1\|\mathbf{v}\|_1 + \gamma_2\,\mathrm{FL}(\mathbf{v}) \leq 1. \tag{2.13}$$

In order to assess the effectivity of our method, we have conducted the same experiment as Witten et al. (2009) using the breast cancer data set described by Chin et al. (2006), consisting of $q = 2,148$ gene expression measurements and $p = 16,962$ CGH measurements for $n = 89$ tissue samples. The matrix decomposition problem of Eq. (2.13) was addressed once for each of the 23 chromosomes, using each time the CGH data available for the corresponding chromosome, and the gene expression of all genes. Following the original choice of Witten et al. (2009), we have selected a regularization parameter $\lambda$ resulting in about 25 non-zero coefficients in $\mathbf{u}$, and selected $\gamma_1 = \gamma_2 = 1$, which results in sparse and piecewise-constant vectors $\mathbf{v}$. The original matrices $(\mathbf{X}, \mathbf{Y})$ are divided into a training set $(\mathbf{X}^{tr}, \mathbf{Y}^{tr})$ formed with 3/4 of the $n$ samples, keeping the rest $(\mathbf{X}^{te}, \mathbf{Y}^{te})$ for testing. This experiment is repeated for 10 random splits, for each chromosome a couple of factors $(\mathbf{u}, \mathbf{v})$ are computed, and the correlations $\mathrm{corr}(\mathbf{X}^{tr}\mathbf{u}, \mathbf{Y}^{tr}\mathbf{v})$ and $\mathrm{corr}(\mathbf{X}^{te}\mathbf{u}, \mathbf{Y}^{te}\mathbf{v})$ are reported on Figure 2.6. The average standard deviation of the experiments results was 0.0339 for the training set and 0.1391 for the test set.

Comparing with the original curves reported by Witten et al. (2009) for their penalized matrix decomposition (PMD) algorithm, our method exhibits in general a performance similar as PMD.[7] Nevertheless, the purpose of this section is more to demonstrate that our method can be used with genomic data than comparing it carefully with PMD. To draw substantial conclusions about the performance of both methods, more experiments would of course be needed.

### 2.6.4 Application to Large-Scale Image Processing

We demonstrate in this section that our algorithm can be used for a difficult large-scale image processing task, namely, removing the text (*inpainting*) from the damaged 12-Megapixel image of Figure 2.7. Using a multi-threaded version of our implementation, we have learned a dictionary with 256 elements from the roughly $7 \times 10^6$ undamaged

[7] The curves for PMD were generated with the R software package available at http://cran.r-project.org/web/packages/PMA/index.html and a script provided by Witten et al. (2009).

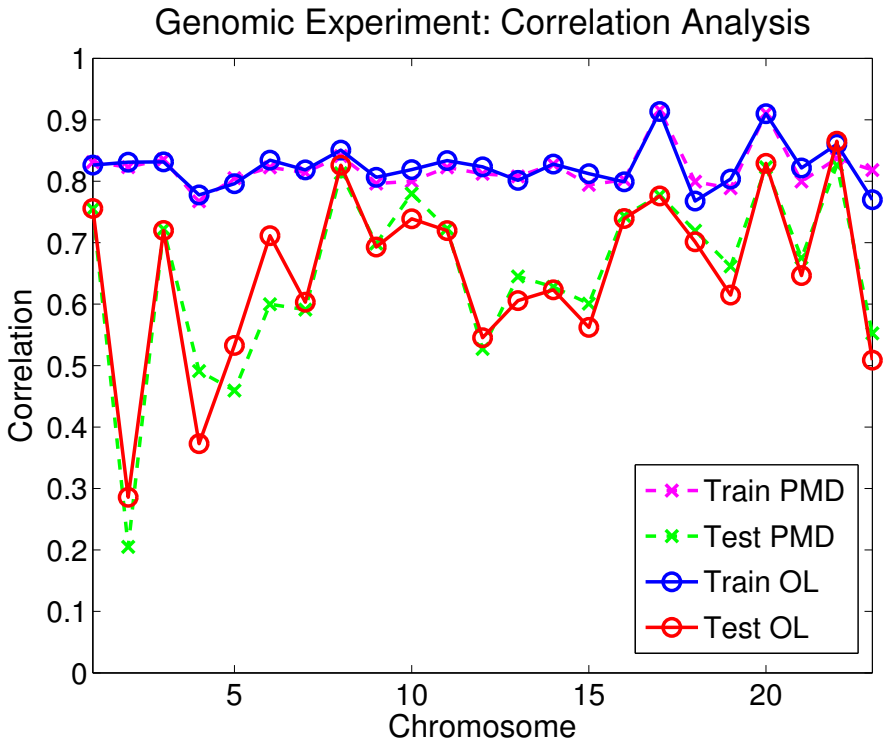



Figure 2.6: SPCA was applied to the covariance matrix obtained from the breast cancer data (Chin et al., 2006). A fused lasso regularization is used for the CGH data. 3/4 of the $n$ samples are used as a training set, keeping the rest for testing. Average correlations from 10 random splits are reported for each of the 23 chromosomes, for PMD (Witten et al., 2009) and our method denoted by OL.

$12 \times 12$ color patches in the image with two epochs in about 8 minutes on a 2.4GHz machine with eight cores. Once the dictionary has been learned, the text is removed using the sparse coding technique for inpainting of Mairal et al. (2008b), and we obtain a convincing result, where artefacts are hardly visible. Our intent here is of course *not* to evaluate our learning procedure in inpainting tasks, which would require a thorough comparison with state-the-art techniques on standard data sets. Instead, we just wish to demonstrate that it can indeed be applied to a realistic, non-trivial image processing task on a large image. Indeed, to the best of our knowledge, this is the first time that dictionary learning is used for image restoration on such large-scale data. For comparison, the dictionaries used for inpainting in Mairal et al. (2008b) are learned (in batch mode) on 200,000 patches only.

Figure 2.7: Inpainting example on a 12-Megapixel image. Top: Damaged and restored images. Bottom: Zooming on the damaged and restored images. Note that the pictures presented here have been scaled down for display, and are best seen in color by zooming on a computer screen.

## 2.7 Conclusion

We have introduced in this work new stochastic online algorithm for learning dictionaries adapted to sparse coding tasks, and proven its convergence. Experiments demonstrate that it is significantly faster than batch alternatives such as Engan et al. (1999), Aharon et al. (2006) and Lee et al. (2007) on large data sets that may contain millions of training examples, yet it does not require a careful learning rate tuning like regular stochastic gradient descent methods. Moreover, we have extended it to other matrix factorization problems such as non negative matrix factorization, and we have proposed a formulation for sparse principal component analysis which can be solved efficiently using our method. Our approach has already shown to be useful for image restoration tasks such as denoising (Mairal et al., 2009c); more experiments are of course needed to better assess its promise in bioinformatics and signal processing. Beyond this, we plan to use the proposed learning framework for sparse coding in computationally demanding

video restoration tasks (Protter and Elad, 2009), with dynamic data sets whose size is not fixed, and extending this framework to different loss functions (Mairal et al., 2009b) to address discriminative tasks such as image classification, which are more sensitive to overfitting than reconstructive ones.

# 3

# Network Flow Algorithms for Structured Sparsity

**Chapter abstract:** We consider a class of learning problems that involve a structured sparsity-inducing norm defined as the sum of $\ell_\infty$-norms over groups of variables. Whereas a lot of effort has been put in developing fast optimization methods when the groups are disjoint or embedded in a specific hierarchical structure, we address here the case of general overlapping groups. To this end, we show that the corresponding optimization problem is related to network flow optimization. More precisely, the proximal problem associated with the norm we consider is dual to a *quadratic min-cost flow problem*. We propose an efficient procedure which computes its solution exactly in polynomial time. Our algorithm scales up to millions of variables, and opens up a whole new range of applications for structured sparse models. We present several experiments on image and video data, demonstrating the applicability and scalability of our approach for various problems.

The work presented in this chapter was achieved with the collaboration of Rodolphe Jenatton, Guillaume Obozinski and Francis Bach, with equal contribution between Rodolphe Jenatton and myself. The main topic of this chapter is optimization and proximal methods. Therefore, Section 1.4.3 should be read before this chapter. The reader is also strongly encouraged to have a look at Section 1.4.8. The material of this chapter is based on the following publication:

J. Mairal*, R. Jenatton*, G. Obozinski, F. Bach. Network Flow Algorithms for Structured Sparsity. Advances in Neural Information Processing Systems. 2010

and introduces some material from

R. Jenatton*, J. Mairal*, G. Obozinski, F. Bach. Proximal Methods for Sparse Hierarchical Dictionary Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*. 2010

R. Jenatton*, J. Mairal*, G. Obozinski, F. Bach. Proximal Methods for Hierarchical Sparse Coding. *submitted*. arXiv:1009.2139v2, 2010 (long version of the previous article)

(*equal contributions)

## 3.1 Introduction

Sparse linear models have become a popular framework for dealing with various unsupervised and supervised tasks in machine learning and signal processing. In such models, linear combinations of small sets of variables are selected to describe the data. Regularization by the $\ell_1$-norm has emerged as a powerful tool for addressing this combinatorial variable selection problem, relying on both a well-developed theory (see Bickel et al., 2009, and references therein) and efficient algorithms (Efron et al., 2004; Nesterov, 2007; Beck and Teboulle, 2009).

The $\ell_1$-norm primarily encourages sparse solutions, regardless of the potential structural relationships (e.g., spatial, temporal or hierarchical) existing between the variables. Much effort has recently been devoted to designing sparsity-inducing regularizations capable of encoding higher-order information about allowed patterns of non-zero coefficients (Jenatton et al., 2009; Jacob et al., 2009; Zhao et al., 2009; Huang et al., 2009; Baraniuk et al., 2010), with successful applications in bioinformatics (Jacob et al., 2009; Kim and Xing, 2010), topic modeling (Jenatton et al., 2010a) and computer vision (Huang et al., 2009). Other attempts for modelling relations between variables include the use of graphical models by Cevher et al. (2009); Garrigues and Olshausen (2008).

By considering sums of norms of appropriate subsets, or *groups*, of variables, these regularizations control the sparsity patterns of the solutions. The underlying optimization problem is usually difficult, in part because it involves nonsmooth components. Proximal methods have proven to be effective in this context, essentially because of their fast convergence rates and their ability to deal with large problems (Nesterov, 2007; Beck and Teboulle, 2009). While the settings where the penalized groups of variables do not overlap (Roth and Fischer, 2008) or are embedded in a tree-shaped hierarchy (Jenatton et al., 2010a) have already been studied, sparsity-inducing regularizations of general overlapping groups have, to the best of our knowledge, never been considered within the proximal method framework.

### 3.1.1 Contributions

This work makes the following contributions:

- It shows that the *proximal operator* associated with the structured norm we consider can be computed by solving a *quadratic min-cost flow* problem, thereby establishing a connection with the network flow optimization literature.
- It presents a fast and scalable procedure for solving a large class of structured sparse regularized problems, which, to the best of our knowledge, have not been addressed efficiently before.
- It shows that the dual norm of the sparsity-inducing norm we consider can also be evaluated efficiently, which enables us to compute duality gaps for the corresponding optimization problems.

- It demonstrates that our method is relevant for various applications, from video background subtraction to estimation of hierarchical structures for dictionary learning of natural image patches.

## 3.2 Related Work and Problem Statement

We consider in this work convex optimization problems of the form

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} f(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha}), \tag{3.1}$$

where $f : \mathbb{R}^p \to \mathbb{R}$ is a convex differentiable function with uniformly Lipschitz gradient and $\Omega : \mathbb{R}^p \to \mathbb{R}$ is a convex, nonsmooth, sparsity-inducing regularization function. When one knows *a priori* that the solutions of this learning problem only have a few non-zero coefficients, $\Omega$ is often chosen to be the $\ell_1$-norm, leading for instance to the Lasso (Tibshirani, 1996). When these coefficients are organized in groups, a penalty encoding explicitly this prior knowledge can improve the prediction performance and/or interpretability of the learned models (Roth and Fischer, 2008; Yuan and Lin, 2006; Huang et al., 2009; Obozinski et al., 2009). Such a penalty, might for example take the form

$$\Omega(\boldsymbol{\alpha}) \triangleq \sum_{g\in\mathcal{G}} \eta_g \|\boldsymbol{\alpha}_g\|, \tag{3.2}$$

where $\mathcal{G}$ is a set of groups of indices, $\boldsymbol{\alpha}_j$ denotes the $j$-th coordinate of $\boldsymbol{\alpha}$ for $j$ in $[\![1;p]\!]$, the vector $\boldsymbol{\alpha}_g$ in $\mathbb{R}^{|g|}$ records the coefficients of $\boldsymbol{\alpha}$ indexed by $g$ in $\mathcal{G}$, and the scalars $\eta_g$ are positive weights. $\|.\|$ could be any norm, but it is often taken to be the $\ell_2$- or $\ell_\infty$-norm in practice, which when $\mathcal{G}$ is the set of singletons of $[\![1;p]\!]$, gets back to the $\ell_1$-norm. For example, we present in the next section an algorithm for dealing with groups embedded in a particular hierarchical structure that works for the $\ell_2$ or $\ell_\infty$-norms. The more general algorithm that we show later and which is the main topic of this chapter requires a sum of $\ell_\infty$-norms, which are piecewise linear, a property that we take advantage of in this work.

If $\mathcal{G}$ is a *partition* of $[\![1;p]\!]$—that is, the groups do not overlap, variables are selected in groups rather than individually. When the groups overlap, $\Omega$ is still a norm and sets groups of variables to zero together (Jenatton et al., 2009). The latter setting has first been considered for hierarchies (Zhao et al., 2009) and will be presented in the next section, and then extended to general group structures (Jenatton et al., 2009). Note that other types of structured sparse models have also been introduced, either through a different norm (Jacob et al., 2009), or through non-convex criteria (Huang et al., 2009; Baraniuk et al., 2010). Solving Eq. (3.1) in this context becomes challenging and is the topic of this work. Following Jenatton et al. (2010a) who tackled the case of hierarchical groups, we propose to approach this problem with proximal methods.

### 3.2.1 Proximal Methods

We suppose known proximal methods, which have been presented in Section 1.4.3. We only recall that using these methods requires to efficiently solve the *proximal operator* associated with the regularization $\lambda\Omega$, which is the function that maps a vector $\mathbf{u}$ in $\mathbb{R}^p$ to the (unique, by strong convexity) solution of

$$\min_{\mathbf{v}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 + \lambda\Omega(\mathbf{v}). \tag{3.3}$$

Solving efficiently this problem is the topic of this chapter. In the simple setting where $\mathcal{G}$ is the set of singletons and the $\eta_g$ are all set to 1, $\Omega$ is the $\ell_1$-norm, and the proximal operator is the elementwise soft-thresholding operator $\mathbf{v}_j \leftarrow \text{sign}(\mathbf{u}_j)\max(|\mathbf{u}_j|-\lambda,0)$ for $j \in [\![1;p]\!]$. Similarly, when the groups in $\mathcal{G}$ do not overlap, the proximal operator can be computed in closed form (Liu et al., 2009). The case of a tree-shaped hierarchical structure has recently been tackled by Jenatton et al. (2010a) based on a dual optimization problem.

The approach we develop extends Jenatton et al. (2010a) to the case of general overlapping groups when $\Omega$ is a weighted sum of $\ell_\infty$-norms[1], extending the applicability of these regularizations to larger problems.

We first recall the approach of Jenatton et al. (2010a) in the next section, and then show that, for a general set $\mathcal{G}$ of overlapping groups, a convex dual of the problem of Eq. (3.3) can be reformulated as a *quadratic min-cost flow problem*.

### 3.2.2 Solving The Hierarchical Case

We present in this section the approach of Jenatton et al. (2010a) for dealing with the case of hierarchies. To simplify the presentation, we present the regularization in the context of a tree-structured dictionary for solving the following problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}),$$

with $\Omega$ is a specific structured sparsity-inducing norm, which we will explicit in the sequel. $\mathbf{x}$ in $\mathbb{R}^m$ is a signal, $\mathbf{D}$ in $\mathbb{R}^{m\times p}$ a dictionary, and $\lambda$ is the regularization parameter. Let us now explicit the norm $\Omega$.

### 3.2.3 Hierarchical Sparsity Inducing Norms

We organize the $p$ entries of $\boldsymbol{\alpha}$ in a rooted-tree $\mathcal{T}$ composed of $p$ nodes, one for each dictionary element $\mathbf{d}^j$, $j \in [\![1;p]\!]$. We will identify these indices $j$ in $[\![1;p]\!]$ and the nodes of $\mathcal{T}$. We want to exploit the structure of $\mathcal{T}$ in the following sense: the decomposition of any vector $\mathbf{x}$ can involve a dictionary element $\mathbf{d}^j$ *only if the ancestors of* $\mathbf{d}^j$ *in* $\mathcal{T}$ *are themselves part of the decomposition*. Equivalently, one can say that when a dictionary

[1]For hierarchies, the approach of Jenatton et al. (2010a) applies also to the case of where $\Omega$ is a weighted sum of $\ell_2$-norms.

element $\mathbf{d}^j$ is not involved in the decomposition of a vector $\mathbf{x}$ then *its descendants in* $\mathcal{T}$ *should not be part of the decomposition.* While these two views are equivalent, the latter leads to an intuitive penalization term.

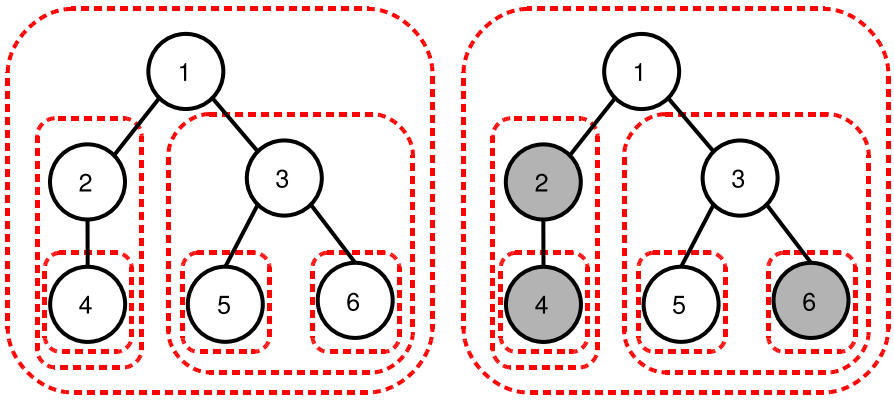



Figure 3.1: Left: example of a tree-structured set of groups $\mathcal{G}$ (dashed contours in red), corresponding to a tree $\mathcal{T} = \{1, \ldots, 6\}$ (in black). Right, example of a sparsity pattern: the groups $\{2, 4\}, \{4\}$ and $\{6\}$ are set to zero, so that the corresponding nodes (in gray) that form subtrees of $\mathcal{T}$ are removed. The remaining nonzero variables $\{1, 3, 5\}$ are such that, if a node is selected, the same goes for all its ancestors.

To obtain models with the desired property, one considers for all $j$ in $\mathcal{T}$, the group $g_j \subseteq \{1, \ldots, p\}$ of dictionary elements that only contains $j$ and all its descendants, and penalizes the number of such groups that are involved in the decomposition of $\mathbf{x}$ (a group being "involved in the decomposition" meaning here that at least one of its dictionary element is part of the decomposition). We call $\mathcal{G}$ this set of groups (Figure 3.1).

While this penalization is non-convex, a convex proxy has been introduced by Zhao et al. (2009) and was further considered by Bach (2009) and Kim and Xing (2010) in the context of regression. It takes the same form as Eq. (3.2)—that is, for any vector $\boldsymbol{\alpha}$ in $\mathbb{R}^p$, it can be written $\Omega(\boldsymbol{\alpha}) = \sum_{g \in \mathcal{G}} \eta_g \|\boldsymbol{\alpha}_g\|$, where $\|.\|$ stands either for the $\ell_\infty$ or $\ell_2$ norm, and $(\eta_g)_{g \in \mathcal{G}}$ denotes some positive weights[2]. As analyzed by Zhao et al. (2009), when penalizing by $\Omega$, some of the vectors $\boldsymbol{\alpha}_g$ are set to zero for some $g \in \mathcal{G}$. Therefore, the components of $\boldsymbol{\alpha}$ corresponding to some entire subtrees of $\mathcal{T}$ are set to zero, which is exactly the desired effect (Figure 3.1).

Note that even though we have presented for simplicity reasons this hierarchical norm in the context of a single tree with a single element at each node, it can be extended easily to the case of forests of trees, and/or trees containing several dictionary elements at each node. More generally, this formulation can be extended with the notion of *tree-structured* groups, which we now present.

[2]For a complete definition of $\Omega$ for any $\ell_q$ norm, a discussion of the choice of $q$, and a strategy for choosing the weights $\eta_g$, see (Zhao et al., 2009; Kim and Xing, 2010).

**Definition 1 (Tree-structured set of groups.)**
*A set of groups $\mathcal{G} = \{g\}_{g\in\mathcal{G}}$ is said to be tree-structured in $\{1,\ldots,p\}$, if $\bigcup_{g\in\mathcal{G}} g = \{1,\ldots,p\}$ and for all $g,h \in \mathcal{G}$, $(g\cap h \neq \emptyset) \Rightarrow (g \subseteq h \text{ or } h \subseteq g)$. For such a set of groups, there exists a (non-unique) total order relation $\preceq$ such that:*

$$g \preceq h \;\Rightarrow\; \{g \subseteq h \quad or \quad g\cap h = \emptyset\}.$$

Sparse hierarchical norms having been introduced, we now address the optimization dealing with such norms.

### 3.2.4 Optimization for Hierarchies

Within the context of regression, several optimization methods to cope with the regularization $\Omega$ have already been proposed. A boosting-like technique with a path-following strategy is used by Zhao et al. (2009). Kim and Xing (2010) use a reweighted least-squares scheme when $\|.\|$ is the $\ell_2$ norm. The same approach is considered by Zhao et al. (2009), but built upon an active set strategy. In this chapter, we propose to perform the updates of the vectors $\boldsymbol{\alpha}_i$ based on a proximal approach which we have introduced in Section 1.4.3.

#### Proximal Operator for the Norm $\Omega$

We now address the general optimization problem presented in Eq. 3.1 with the help of proximal methods, which we have recalled in Section 3.2.1, and we show that Eq. (3.3) can be solved with a primal-dual approach. The procedure solves a dual formulation of Eq. (3.3) involving the dual norm[3] of $\|.\|$, denoted by $\|.\|_*$, and defined by $\|\boldsymbol{\kappa}\|_* = \max_{\|\mathbf{z}\|\leq 1} \mathbf{z}^\top\boldsymbol{\kappa}$ for any vector $\boldsymbol{\kappa}$ in $\mathbb{R}^p$. The formulation is described in the next lemma that relies on conic duality (Boyd and Vandenberghe, 2004). The rationale for using conic duality is to come up with a dual problem without overlapping variables.

**Lemma 3 (Dual of the proximal problem)**
*Let $\mathbf{u} \in \mathbb{R}^p$ and let us consider the problem*

$$\begin{aligned} &\max_{\boldsymbol{\xi}\in\mathbb{R}^{p\times|\mathcal{G}|}} -\frac{1}{2}(\|\mathbf{u} - \sum_{g\in\mathcal{G}} \boldsymbol{\xi}^g\|_2^2 - \|\mathbf{u}\|_2^2) \\ &s.t.\ \forall g \in \mathcal{G},\ \|\boldsymbol{\xi}^g\|_* \leq \lambda\eta_g \text{ and } \boldsymbol{\xi}^g_j = 0 \text{ if } j \notin g, \end{aligned} \tag{3.4}$$

*where $\boldsymbol{\xi} = (\boldsymbol{\xi}^g)_{g\in\mathcal{G}}$ and $\boldsymbol{\xi}^g_j$ denotes the $j$-th coordinate of the vector $\boldsymbol{\xi}^g$ in $\mathbb{R}^p$. Then, problems (3.3) and (3.4) are dual to each other and strong duality holds. In addition, the pair of primal-dual variables $\{\mathbf{v},\boldsymbol{\xi}\}$ is optimal if and only if $\boldsymbol{\xi}$ is a feasible point of*

[3] It is easy to show that the dual norm of the $\ell_2$ norm is the $\ell_2$ norm itself. The dual norm of the $\ell_\infty$ is the $\ell_1$ norm.

*the optimization problem (3.4), and*

$$\mathbf{v} = \mathbf{u} - \sum_{g \in \mathcal{G}} \boldsymbol{\xi}^g,$$
$$\text{and for all } g \in \mathcal{G}, \begin{cases} \mathbf{v}_{|g}^\top \boldsymbol{\xi}^g = \|\mathbf{v}_{|g}\| \|\boldsymbol{\xi}^g\|_* \text{ and } \|\boldsymbol{\xi}^g\|_* = \lambda \eta_g, \\ \text{or} \quad \mathbf{v}_{|g} = 0. \end{cases} \tag{3.5}$$

In this lemma, we use the notation $\mathbf{v}_{gi}$ which is the vector of size $p$ whose coordinates are equal to those of $\mathbf{v}$ for indices in the set $g$, and 0 otherwise. Note the difference with the notation $\mathbf{v}_g$, which is often used in works on structured sparsity, where $\mathbf{v}_g$ is a vector of size $|g|$. The proof of the lemma, as well as all the ones of the subsequent lemmas and propositions are given in Appendix B for readability purposes. Note that we focus here on specific tree-structured groups, but the previous lemma is valid regardless of the nature of $\mathcal{G}$.

After removing the constant terms, the dual problem can be rewritten as

$$\min_{\boldsymbol{\xi} \in \mathbb{R}^{p \times |\mathcal{G}|}} \frac{1}{2} \|\mathbf{u} - \sum_{g \in \mathcal{G}} \boldsymbol{\xi}^g\|_2^2 \quad \text{s.t.} \quad \forall g \in \mathcal{G}, \ \|\boldsymbol{\xi}^g\|_* \leq \lambda \eta_g \text{ and } \boldsymbol{\xi}_j^g = 0 \text{ if } j \notin g.$$

The structure of the dual problem of Eq. (3.4), i.e., the separability of the (convex) constraints for each vector $\boldsymbol{\xi}^g$, $g \in \mathcal{G}$, makes it possible to use block coordinate ascent (Bertsekas, 1999). Such a procedure is presented in Algorithm 3. It optimizes sequentially Eq. (3.4) with respect to the variable $\boldsymbol{\xi}^g$, while keeping fixed the other variables $\boldsymbol{\xi}^h$, for $h \neq g$. It is easy to see from Eq. (3.4) that such an update for a group $g$ in $\mathcal{G}$ amounts to the orthogonal projection of the vector $\mathbf{u}_{|g} - \sum_{h \neq g} \xi_{|g}^h$ onto the ball of radius $\lambda \eta_g$ of the dual norm $\|.\|_*$. We denote this projection by $\Pi^*_{\lambda \eta_g}$.

**Algorithm 3** Block coordinate ascent in the dual

Inputs: $\mathbf{u} \in \mathbb{R}^p$ and set of groups $\mathcal{G}$.
Outputs: $(\mathbf{v}, \boldsymbol{\xi})$ (primal-dual solutions).
Initialization: $\mathbf{v} = \mathbf{u}$, $\boldsymbol{\xi} = 0$.
**while** ( *maximum number of iterations not reached* ) **do**
  **for** $g \in \mathcal{G}$ **do**
    $\mathbf{v} \leftarrow \mathbf{u} - \sum_{h \neq g} \boldsymbol{\xi}^h$.
    $\boldsymbol{\xi}^g \leftarrow \Pi^*_{\lambda \eta_g}(\mathbf{v}_{|g})$.
  **end for**
**end while**
$\mathbf{v} \leftarrow \mathbf{u} - \sum_{g \in \mathcal{G}} \boldsymbol{\xi}^g$.

### Convergence in One Pass for the $\ell_2$- and $\ell_\infty$-Norms

In general, Algorithm 3 is not guaranteed to solve exactly Eq. (3.3) in a finite number of iterations. However, when $\|.\|$ is the $\ell_2$- or $\ell_\infty$-norm, and provided that the groups

in $\mathcal{G}$ are appropriately ordered, we now prove that only *one pass* of Algorithm 3, i.e., only one iteration over all groups, is sufficient to obtain the exact solution of Eq. (3.3).

Before stating this result, we need to introduce two lemmas. The first lemma characterizes the projections onto norm balls.

**Lemma 4** (**Projection on the dual ball**)
*Let* $\mathbf{v} \in \mathbb{R}^p$ *and* $t > 0$. *We have*

$$\boldsymbol{\kappa} = \Pi_t^*(\mathbf{v})$$

*if and only if* $\begin{cases} \text{if } \|\mathbf{v}\|_* \leq t, & \boldsymbol{\kappa} = \mathbf{v}, \\ \text{otherwise,} & \|\boldsymbol{\kappa}\|_* = t \quad \text{and} \quad \boldsymbol{\kappa}^\top(\mathbf{v} - \boldsymbol{\kappa}) = \|\boldsymbol{\kappa}\|_* \|\mathbf{v} - \boldsymbol{\kappa}\|. \end{cases}$

The second lemma shows that, given two nested groups $g, h$ such that $g \subseteq h \subseteq \{1, \ldots, p\}$, if $\boldsymbol{\xi}^g$ is updated before $\boldsymbol{\xi}^h$ in Algorithm 3, then the optimality condition of $\boldsymbol{\xi}^g$ is not perturbed by the update of $\boldsymbol{\xi}^h$. In other words, this lemma indicates that the correct order to consider in Algorithm 3 is $\preceq$ (see Definition 1).

**Lemma 5** (**Projections with nested groups**)
*Let* $\|.\|$ *denote either the* $\ell_2$ *or* $\ell_\infty$ *norm, and* $g$ *and* $h$ *be two nested groups—that is,* $g \subseteq h \subseteq \{1, \ldots, p\}$. *Let* $\mathbf{v}$ *be a vector in* $\mathbb{R}^p$, *and let us consider the successive projections*

$$\boldsymbol{\kappa}^g \triangleq \Pi_{t_g}^*(\mathbf{v}_{|g}) \quad \text{and} \quad \boldsymbol{\kappa}^h \triangleq \Pi_{t_h}^*(\mathbf{v}_{|h} - \boldsymbol{\kappa}^g)$$

*with* $t_g, t_h > 0$. *Then, we have as well*

$$\boldsymbol{\kappa}^g = \Pi_{t_g}^*(\mathbf{v}_{|g} - \boldsymbol{\kappa}_{|g}^h).$$

The previous lemma establishes the convergence in one pass of Algorithm 3 in the case where $\mathcal{G}$ contains only two nested groups $g \subseteq h$, provided that $\boldsymbol{\xi}^g$ is computed before $\boldsymbol{\xi}^h$. In the following proposition, this lemma is extended to general tree-structured sets of groups $\mathcal{G}$:

**Proposition 7** (**Convergence in one pass**)
*Suppose that the groups in* $\mathcal{G}$ *are ordered according to* $\preceq$ *and that the norm* $\|.\|$ *is either the* $\ell_2$ *or* $\ell_\infty$ *norm. Then, after initializing* $\boldsymbol{\xi}$ *to* 0, ***one pass*** *of Algorithm 3 with the order* $\preceq$ *gives the solution of 3.4.*

Interestingly, this property that is true for the $\ell_2$ or $\ell_\infty$-norms is not true for other $\ell_q$-norms, and in fact counter examples exists for $q \neq 2, \infty$ (see Jenatton et al., 2010b).

### Efficient Implementation of the Proximal Operator

Since one pass of Algorithm 3 involves $p$ projections on the dual balls (respectively the $\ell_2$ and the $\ell_1$ balls for the $\ell_2$- and $\ell_\infty$-norms) of vectors in $\mathbb{R}^p$, a naive implementation leads to a polynomial complexity in $O(p^2)$, since each of these projections can be obtained in $O(p)$ operations (Brucker, 1984; Maculan and de Paula, 1989). However, we show that in these cases, the primal solution $\mathbf{v}$, which is the quantity of interest, can be obtained with a smaller complexity, without considering explicitly the dual variable $\boldsymbol{\xi}$.

We present a fast recursive implementation in Algorithm 4 for the $\ell_2$-norm case. Two new notations are used: For a group $g$ in $\mathcal{G}$, we denote by $\text{root}(g)$ the indices of the variables that are at the root of the subtree corresponding to $g$, and by $\text{children}(g)$ the set of groups that are the children of $\text{root}(g)$ in the tree. For instance, in the tree presented in Figure 3.1, $\text{root}(\{3,5,6\}) = \{3\}$, $\text{root}(\{1,2,3,4,5,6\}) = \{1\}$, $\text{children}(\{3,5,6\}) = \{\{5\},\{6\}\}$, and $\text{children}(\{1,2,3,4,5,6\}) = \{\{2,4\},\{3,5,6\}\}$. Note that all the groups of $\text{children}(g)$ are necessarily included in $g$.

**Algorithm 4** Fast implementation of Algorithm 3 when $\|.\|$ is the $\ell_2$-norm.

**Require:** $\mathbf{u} \in \mathbb{R}^p$ (input vector), set of groups $\mathcal{G}$, $(\eta_g)_{g\in\mathcal{G}}$ (positive weights), and $g_0$ (root of the tree).

1: Variables: $\boldsymbol{\rho} = (\rho_g)_{g\in\mathcal{G}}$ in $\mathbb{R}^{|\mathcal{G}|}$ (scaling factors); $\mathbf{v}$ in $\mathbb{R}^p$ (output, primal variable).
2: `computeNorm`($g_0$).
3: `recursiveScaling`($g_0$,1).
4: **return** $\mathbf{v}$ (primal solution).

**Procedure** `computeNorm`($g$)

1: Compute the squared norm of the group:

$$\gamma_g \leftarrow \|\mathbf{u}_{\text{root}(g)}\|_2^2 + \sum_{h\in\text{children}(g)} \texttt{computeNorm}(h).$$

2: Compute the scaling factor of the group: $\rho_g \leftarrow \max(0, 1 - \lambda\eta_g/\sqrt{\gamma_g})$.
3: **return** $\gamma_g\rho_g^2$.

**Procedure** `recursiveScaling`($g$,$t$)

1: $\rho_g \leftarrow t\rho_g$.
2: $\mathbf{v}_{\text{root}(g)} \leftarrow \rho_g\mathbf{u}_{\text{root}(g)}$.
3: **for** $h \in \text{children}(g)$ **do**
4: `recursiveScaling`($h$,$\rho_g$).
5: **end for**

The successive projections on the ball of the $\ell_2$-norm in Algorithm 3 amount to performing sequences of scaling operations. The intuition behind the recursive implementation presented in Algorithm 4 is that part of the computations corresponding to these scaling operations can be factorized thanks to the tree structure. The next lemma ensures the correctness of this implementation and gives its complexity. More details can be found in the proof, which is relegated to the appendix.

**Lemma 6 (Correctness and complexity of Algorithm 4)**
*When $\|.\|$ is chosen to be the $\ell_2$-norm, Algorithm 4 gives the solution of the primal problem Eq. (3.3) in $O(p)$ operations.*

We then give a simple recursive implementation of Algorithm 3 when $\|.\|$ is chosen to be the $\ell_\infty$-norm. The details of the procedure are described in Algorithm 5.

The next lemma ensures the correctness of this algorithm and gives its complexity:

**Algorithm 5** Fast implementation of Algorithm 3 when $\|.\|$ is the $\ell_\infty$-norm.

**Require:** $\mathbf{u} \in \mathbb{R}p$, set of groups $\mathcal{G}$, $(\eta_g)_{g\in\mathcal{G}}$ (positive weights), and $g_0$ (index of the group that contains everybody).

1: Variables: $\boldsymbol{\rho} = (\rho_g)_{g\in\mathcal{G}}$ in $\mathbb{R}^{|\mathcal{G}|}$, $\mathbf{v}$ in $\mathbb{R}^p$ (output).
2: Initialization: $\mathbf{v} \leftarrow \mathbf{u}$.
3: `recursiveProjection`$(g_0)$.
4: **return** $\mathbf{v}$ (primal solution).

**Procedure** `recursiveProjection`$(g)$

1: **for** $h \in$ children$(g)$ **do**
2: `recursiveProjection`$(h)$.
3: **end for**
4: $\mathbf{v}_{|g} \leftarrow \mathbf{v}_{|g} - \Pi^*_{\lambda\eta_g}(\mathbf{v}_{|g})$.

**Lemma 7** (**Correctness and complexity of Algorithm 5**)
*When $\|.\|$ is chosen to be the $\ell_\infty$-norm, Algorithm 5 gives the solution of the primal problem Eq. (3.3) in $O(pd)$ operations, where $d$ is the depth of the tree.*

Note that the claim of having a linear complexity in the $\ell_\infty$-case is slightly abusive, since $d$ could depend of $p$ as well. For instance, in an unbalanced case, the worse case could be $d = O(p)$, in a balanced tree, one could have $d = O(\log(p))$. In practice, the structures we have considered experimentally are relatively flat, with a depth not exceeding $d = 5$.

## 3.3 Proposed Approach

The dual formulation to problem (3.3) has been introduced, we now look at the general case of overlapping groups, and stop assuming a tree-structure. From now on, however, $\Omega$ as to be a sum of $\ell_\infty$-norms. Specializing the dual formulation to this case, we can rewrite it in the following way

**Lemma 8** (Dual of the proximal problem for the $\ell_\infty$-norm)
*Let $\mathbf{u} \in \mathbb{R}^p$ and consider the problem*

$$\min_{\boldsymbol{\xi}\in\mathbb{R}^{p\times|\mathcal{G}|}} \frac{1}{2}\|\mathbf{u} - \sum_{g\in\mathcal{G}} \boldsymbol{\xi}^g\|_2^2 \quad s.t. \quad \forall g \in \mathcal{G},\ \|\boldsymbol{\xi}^g\|_1 \leq \lambda\eta_g \quad \text{and} \quad \boldsymbol{\xi}^g_j = 0 \ \text{ if } j \notin g, \tag{3.6}$$

*where $\boldsymbol{\xi} = (\boldsymbol{\xi}^g)_{g\in\mathcal{G}}$ and $\boldsymbol{\xi}^g_j$ denotes the $j$-th coordinate of the vector $\boldsymbol{\xi}^g$ in $\mathbb{R}^p$. Then, denoting by $\mathbf{v}^\star$ the solution of Eq. (3.3), and $\boldsymbol{\xi}^\star$ a solution of Eq. (3.6), the following relation holds: $\mathbf{v}^\star = \mathbf{u} - \sum_{g\in\mathcal{G}} \boldsymbol{\xi}^{\star g}$.*

To interpret this dual problem as a network flow problem, we introduce the appropriate framework.

### 3.3.1 Graph Model

Without loss of generality[4], we assume in the rest of the chapter that the scalars $\mathbf{u}_j$ are all non-negative, and add non-negativity constraints on $\boldsymbol{\xi}$.

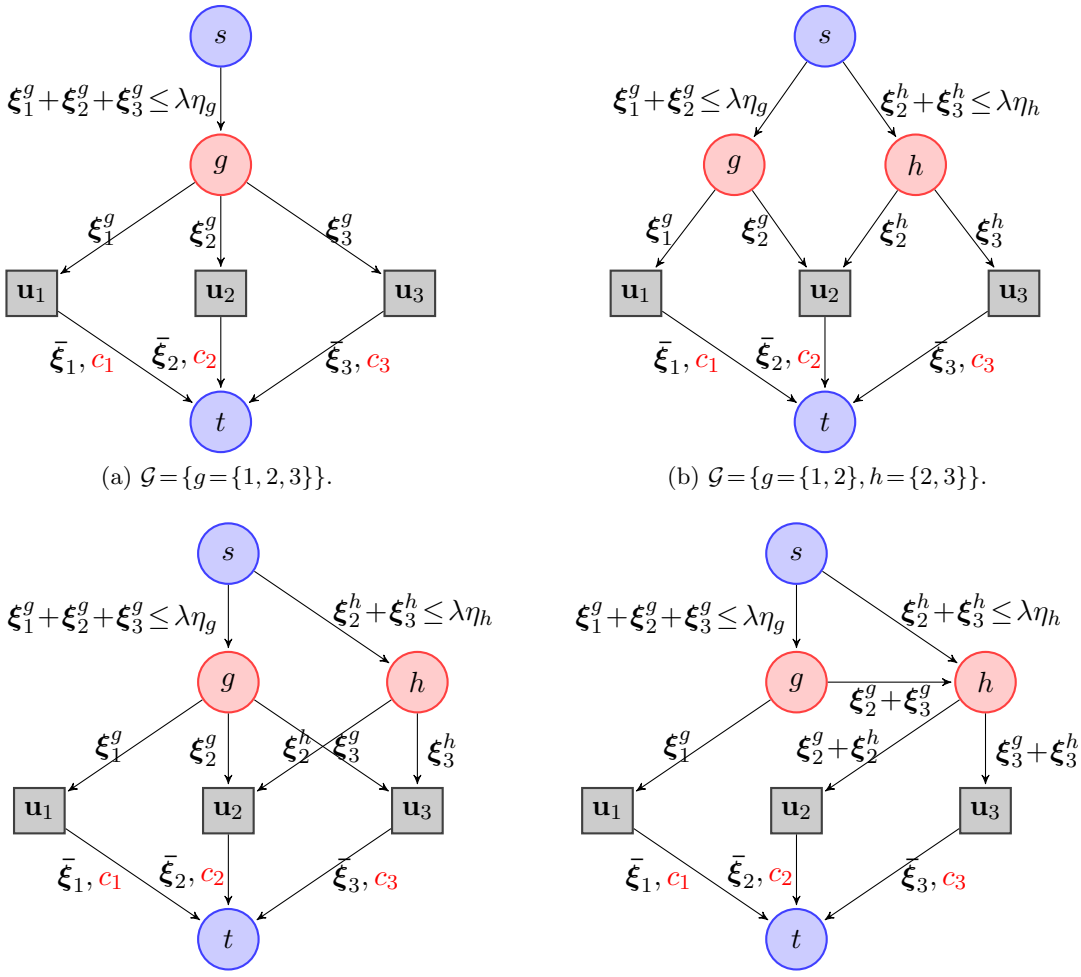
Let $G$ be a directed graph $G = (V, E, s, t)$, where $V$ is a set of vertices, $E \subseteq V \times V$ a set of arcs, $s$ a source, and $t$ a sink. Let $c$ and $c'$ be two functions on the arcs, $c : E \to \mathbb{R}$ and $c' : E \to \mathbb{R}^+$, where $c$ is a *cost function* and $c'$ is a non-negative *capacity function*. A *flow* is a non-negative function on arcs that satisfies capacity constraints on all arcs (the value of the flow on an arc is less than or equal to the arc capacity) and conservation constraints on all vertices (the sum of incoming flows at a vertex is equal to the sum of outgoing flows) except for the source and the sink.

We introduce a unique *canonical* graph $G$ associated with our optimization problem, which is characterized by the following construction:
(i) $V$ is the union of two sets of vertices $V_u$ and $V_{gr}$, where $V_u$ contains exactly one vertex for each index $j$ in $[\![1; p]\!]$, and $V_{gr}$ contains exactly one vertex for each group $g$ in $\mathcal{G}$. We thus have $|V| = |\mathcal{G}| + p$. For simplicity, we identify groups and indices with the vertices of the graph.
(ii) For every group $g$ in $\mathcal{G}$, $E$ contains an arc $(s, g)$. These arcs have capacity $\lambda\eta_g$ and zero cost.
(iii) For every group $g$ in $\mathcal{G}$, and every index $j$ in $g$, $E$ contains an arc $(g, j)$ with zero cost and infinite capacity. We denote by $\boldsymbol{\xi}_j^g$ the flow on this arc.
(iv) For every index $j$ in $[\![1; p]\!]$, $E$ contains an arc $(j, t)$ with infinite capacity and a cost $c_j \triangleq \frac{1}{2}(\mathbf{u}_j - \bar{\boldsymbol{\xi}}_j)^2$, where $\bar{\boldsymbol{\xi}}_j$ is the flow on $(j, t)$. Note that by flow conservation, we necessarily have $\bar{\boldsymbol{\xi}}_j = \sum_{g \in \mathcal{G}} \boldsymbol{\xi}_j^g$.

Examples of canonical graphs are given in Figures 3.2a-3.2c, The flows $\boldsymbol{\xi}_j^g$ associated with $G$ can now be identified with the variables of problem (3.6): indeed, the sum of the costs on the edges leading to the sink is equal to the objective function of (3.6), while the capacities of the arcs $(s, g)$ match the constraints on each group. This shows that finding a flow *minimizing the sum of the costs* on such a graph is equivalent to solving problem (3.6).

When some groups are included in others, a *canonical* graph can be simplified to yield a graph with a smaller number of edges. Specifically if $h$ and $g$ are groups with $h \subset g$, the edges $(g, j)$ for $j \in h$ carrying a flow $\boldsymbol{\xi}_j^g$ can be removed and replaced by a single edge $(g, h)$ of infinite capacity and zero cost, carrying the flow $\sum_{j \in h} \boldsymbol{\xi}_j^g$ which is then redistributed after node $h$. This simplification is illustrated in Figure 3.2d, where the graph is equivalent to the one of Figure 3.2c. Remarkably it does not change the optimal value of $\bar{\boldsymbol{\xi}}^\star$, which is the quantity of interest for computing the optimal primal variable $\mathbf{v}^\star$. These modifications are useful in practice, as they reduce the number of edges in the graph and improve the speed of the algorithms we are going to present.

[4] Let $\boldsymbol{\xi}^\star$ denote a solution of Eq. (3.6). The optimality conditions of Eq. (3.6) derived by Jenatton et al. (2010a) show that for all $j$ in $[\![1; p]\!]$, the signs of the non-zero coefficients $\boldsymbol{\xi}_j^{\star g}$ for $g$ in $\mathcal{G}$ are the same as the signs of the entries $\mathbf{u}_j$. To solve Eq. (3.6), one can therefore flip the signs of the negative variables $\mathbf{u}_j$, solve the modified dual formulation (with non-negative variables), which gives the magnitude of the coefficients $\boldsymbol{\xi}_j^{\star g}$ (the signs of these being known beforehand).




(a) $\mathcal{G}=\{g=\{1,2,3\}\}$.

(b) $\mathcal{G}=\{g=\{1,2\}, h=\{2,3\}\}$.

(c) $\mathcal{G}=\{g=\{1,2,3\}, h=\{2,3\}\}$.

(d) $\mathcal{G}=\{g=\{1\}\cup h, h=\{2,3\}\}$.

Figure 3.2: Graph representation of simple proximal problems with different group structures $\mathcal{G}$. The three indices $1,2,3$ are represented as grey squares, and the groups $g,h$ in $\mathcal{G}$ as red discs. The source is linked to every group $g,h$ with respective maximum capacity $\lambda\eta_g, \lambda\eta_h$ and zero cost. Each variable $\mathbf{u}_j$ is linked to the sink $t$, with an infinite capacity, and with a cost $c_j \triangleq \frac{1}{2}(\mathbf{u}_j - \bar{\xi}_j)^2$. All other arcs in the graph have zero cost and infinite capacity, and represent inclusion relations, children being included in their parents. Note that the graphs (c) and (d) correspond to a special case of tree-structured hierarchy in the sense of (Jenatton et al., 2010a). Their min-cost flow problems are equivalent.

Formally, such a notion of equivalence between graphs can be summarized by the following lemma.

**Lemma 9 (Equivalence to canonical graphs.)**
*Let $G = (V, E, s, t)$ be the canonical graph corresponding to a group structure $\mathcal{G}$ with weights $(\eta_g)_{g\in\mathcal{G}}$. Let $G' = (V, E', s, t)$ be a graph sharing the same set of vertices, source and sink as $G$, but with a different arc set $E'$. We say that $G'$ is equivalent to $G$ if and only if the following conditions hold:*

- *Arcs of $E'$ outgoing from the source are the same as in $E$, with the same costs and capacities.*
- *Arcs of $E'$ going to the sink are the same as in $E$, with the same costs and capacities.*
- *For every arc $(g, j)$ in $E$, with $(g, j)$ in $V_{gr} \times V_u$, there exists a unique path in $E'$ from $g$ to $j$ with zero costs and infinite capacities on every arc of the path.*
- *Conversely, if there exists a path in $E'$ between a vertex $g$ in $V_{gr}$ and a vertex $j$ in $V_u$, then there exists an arc $(g, j)$ in $E$.*

*Then, the cost of the optimal min-cost flow on $G$ and $G'$ are the same. Moreover, the values of the optimal flow on the arcs $(j, t)$, $j$ in $V_u$, are the same on $G$ and $G'$.*

The proof of this lemma, as well as all other proofs of this chapter, are given in Appendix B.

### 3.3.2 Proposed Algorithm

Quadratic min-cost flow problems have been well studied in the operations research literature (Hochbaum and Hong, 1995). One of the simplest cases, where $\mathcal{G}$ contains a single group $g$ as in Figure 3.2a, can be solved by an orthogonal projection on the $\ell_1$-ball of radius $\lambda\eta_g$.

It has been shown, both in machine learning (Duchi et al., 2008) and operations research (Hochbaum and Hong, 1995; Brucker, 1984), that such a projection can be done in $O(p)$ operations. In the latter community, this problem is known to be a particular instance of the continuous quadratic knapsack problem. When the group structure is a tree (i.e., the only overlaps possible are inclusions), as in Figure 3.2d (Note that in that case the graph induced on $V$ is a tree), strategies developed in the two communities are also similar (Jenatton et al., 2010a; Hochbaum and Hong, 1995), and solve the problem in $O(pd)$ operations, where $d$ is the depth of the tree.

The general case of overlapping groups is more difficult. Hochbaum and Hong (1995) have shown that *quadratic min-cost flow problems* can be reduced to a specific *parametric max-flow* problem, for which efficient algorithms exists (Gallo et al., 1989).[5] While this

[5]By definition, a parametric max-flow problem consists in solving, for every value of a parameter, a max-flow problem on a graph whose arc capacities depend on this parameter.

approach could be used to solve Eq. (3.6), it ignores the fact that our graphs have non-zero costs only on edges leading to the sink. To take advantage of this specificity, we propose the dedicated Algorithm 6. Our method clearly shares some similarities with a simplified version of Gallo et al. (1989) presented by Babenko and Goldberg (2006), namely a divide and conquer strategy. We perform an empirical comparison between our algorithm and algorithms described by Gallo et al. (1989) in Section 3.5.2.

**Algorithm 6** Computation of the proximal operator for overlapping groups.

1: Inputs: $\mathbf{u} \in \mathbb{R}^p$, a set of groups $\mathcal{G}$, positive weights $(\eta_g)_{g\in\mathcal{G}}$, and $\lambda$ (regularization parameter).
2: Build a graph $G_0 = (V_0, E_0, s, t)$ as explained in Section 3.3.1.
3: Compute the optimal flow: $\bar{\boldsymbol{\xi}} \leftarrow \texttt{computeFlow}(V_0, E_0)$.
4: **Return:** $\mathbf{v} = \mathbf{u} - \bar{\boldsymbol{\xi}}$ (optimal solution of the proximal problem).

**Function** `computeFlow`$(V = V_u \cup V_{gr}, E)$

1: Projection step: $\boldsymbol{\gamma} \leftarrow \arg\min_{\gamma} \sum_{j\in V_u} \frac{1}{2}(\mathbf{u}_j - \boldsymbol{\gamma}_j)^2$ s.t. $\sum_{j\in V_u} \boldsymbol{\gamma}_j \leq \lambda \sum_{g\in V_{gr}} \eta_g$.
2: For all nodes $j$ in $V_u$, set $\boldsymbol{\gamma}_j$ to be the capacity of the arc $(j, t)$.
3: Max-flow step: Update $(\bar{\boldsymbol{\xi}}_j)_{j\in V_u}$ by computing a max-flow on the graph $(V, E, s, t)$.
4: **if** $\exists\, j \in V_u$ s.t. $\bar{\boldsymbol{\xi}}_j \neq \boldsymbol{\gamma}_j$ **then**
5: Denote by $(s, V^+)$ and $(V^-, t)$ the two disjoint subsets of $(V, s, t)$ separated by the minimum $(s, t)$-cut of the graph, and remove the arcs between $V^+$ and $V^-$. Call $E^+$ and $E^-$ the two remaining disjoint subsets of $E$ corresponding to $V^+$ and $V^-$.
6: $(\bar{\boldsymbol{\xi}}_j)_{j\in V_u^+} \leftarrow \texttt{computeFlow}(V^+, E^+)$.
7: $(\bar{\boldsymbol{\xi}}_j)_{j\in V_u^-} \leftarrow \texttt{computeFlow}(V^-, E^-)$.
8: **end if**
9: **Return:** $(\bar{\boldsymbol{\xi}}_j)_{j\in V_u}$.

Informally, `computeFlow`$(V_0, E_0)$ returns the optimal flow vector $\bar{\boldsymbol{\xi}}$, proceeding as follows: This function first solves a relaxed version of problem Eq. (3.6) obtained by replacing the sum of the vectors $\boldsymbol{\xi}^g$ by a single vector $\boldsymbol{\gamma}$ whose $\ell_1$-norm should be less than, or equal to, the sum of the constraints on the vectors $\boldsymbol{\xi}^g$. The optimal vector $\boldsymbol{\gamma}$ therefore gives a lower bound $\|\mathbf{u} - \boldsymbol{\gamma}\|_2^2/2$ on the optimal cost. Then, the maximum-flow step (Goldberg and Tarjan, 1986) tries to find a feasible flow such that the vector $\bar{\boldsymbol{\xi}}$ matches $\boldsymbol{\gamma}$. If $\bar{\boldsymbol{\xi}} = \boldsymbol{\gamma}$, then the cost of the flow reaches the lower bound, and the flow is optimal. If $\bar{\boldsymbol{\xi}} \neq \boldsymbol{\gamma}$, the lower bound cannot be reached, and we construct a minimum $(s, t)$-cut of the graph (Ford and Fulkerson, 1956) that defines two disjoints sets of nodes $V^+$ and $V^-$; $V^+$ is the part of the graph that can potentially receive more flow from the source, whereas all arcs linking $s$ to $V^-$ are saturated. The properties of a min $(s, t)$-cut (Bertsekas, 1991) imply that there are no arcs from $V^+$ to $V^-$ (arcs inside $V$ have infinite capacity by construction), and that there is no flow on arcs from $V^-$ to $V^+$. At this point, it is possible to show that the value of the optimal min-cost flow on these arcs is also zero. Thus, removing them yields an equivalent optimization problem, which

can be decomposed into two independent problems of smaller size and solved recursively by the calls to `computeFlow`$(V^+, E^+)$ and `computeFlow`$(V^-, E^-)$. Note that when $\Omega$ is the $\ell_1$-norm, our algorithm solves problem (3.6) during the first projection step in line 1 and stops.

We now prove that our algorithm converges and that it finds the optimal solution of the proximal problem. This requires that we introduce the optimality conditions for problem (3.6) derived by Jenatton et al. (2010a), since our convergence proof essentially checks that these conditions are satisfied upon termination of the algorithm.

**Lemma 10 (Optimality conditions of the problem (3.6), Jenatton et al., 2010a)**
*The primal-dual variables* $(\mathbf{v}, \boldsymbol{\xi})$ *are respectively solutions of the primal (3.3) and dual problems (3.6) if and only if the dual variable* $\boldsymbol{\xi}$ *is feasible for the problem (3.6) and*

$$\begin{aligned} &\mathbf{v} = \mathbf{u} - \textstyle\sum_{g \in \mathcal{G}} \boldsymbol{\xi}^g, \\ &\forall g \in \mathcal{G}, \quad \begin{cases} \mathbf{v}_g^\top \boldsymbol{\xi}_g^g = \|\mathbf{v}_g\|_\infty \|\boldsymbol{\xi}^g\|_1 \quad \textit{and} \quad \|\boldsymbol{\xi}^g\|_1 = \lambda \eta_g, \\ \textit{or} \quad \mathbf{v}_g = 0. \end{cases} \end{aligned}$$

This lemma is only an application of Lemma 7. Its proof is therefore immediate. Note that these optimality conditions provide an intuitive view of our min-cost flow problem. Solving the min-cost flow problem is equivalent to sending the maximum amount of flow in the graph under the capacity constraints, while respecting the rule that *the flow outgoing from a group $g$ should always be directed to the variables $\mathbf{u}_j$ with maximum residual* $\mathbf{u}_j - \sum_{g \in \mathcal{G}} \boldsymbol{\xi}_j^g$.

Before proving the convergence and correctness of our algorithm, we also recall classical properties of the min capacity cuts, which we intensively use in the proofs of this chapter. The procedure `computeFlow` of our algorithm finds a minimum $(s,t)$-cut of a graph $G = (V, E, s, t)$, dividing the set $V$ into two disjoint parts $V^+$ and $V^-$. $V^+$ is by construction the sets of nodes in $V$ such that there exists a non-saturating path from $s$ to $V$, while all the paths from $s$ to $V^-$ are saturated. Conversely, arcs from $V^+$ to $t$ are all saturated, whereas there can be non-saturated arcs from $V^-$ to $t$. Moreover, the following properties hold

- There is no arc going from $V^+$ to $V^-$. Otherwise the value of the cut would be infinite. (Arcs inside $V$ have infinite capacity by construction of our graph).
- There is no flow going from $V^-$ to $V^+$ (see properties of the minimum $(s,t)$-cut Bertsekas, 1991).
- The cut goes through all arcs going from $V^+$ to $t$, and all arcs going from $s$ to $V^-$.

All these properties are illustrated in Figure 3.3.

Recall that we assume (cf. Section 3.3.1) that the scalars $\mathbf{u}_j$ are all non negative, and that we add non-negativity constraints on $\boldsymbol{\xi}$. With the optimality conditions of Lemma 10 in hand, we can show our first convergence result.

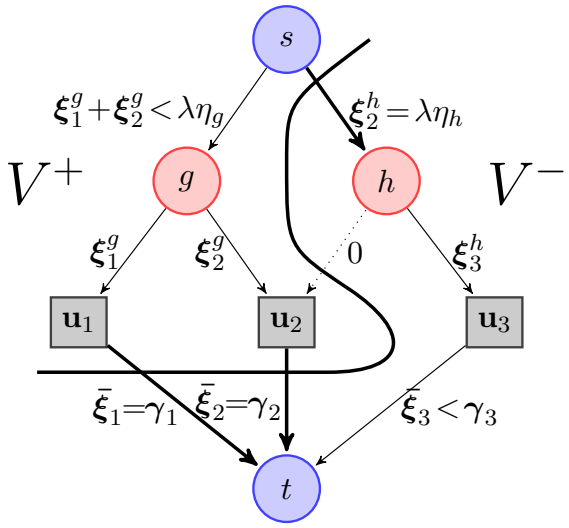



Figure 3.3: Cut computed by our algorithm. $V^+ = V_u^+ \cup V_{gr}^+$, with $V_{gr}^+ = \{g\}$, $V_u^+ = \{1, 2\}$, and $V^- = V_u^- \cup V_{gr}^-$, with $V_{gr}^- = \{h\}$, $V_u^- = \{3\}$. Arcs going from $s$ to $V^-$ are saturated, as well as arcs going from $V^+$ to $t$. Saturated arcs are in bold. Arcs with zero flow are dotted.

**Proposition 8** (**Convergence of Algorithm 6**)
*Algorithm 6 converges in a finite and polynomial number of operations.*

The proof is given in Appendix B. After proving the convergence, we prove that the algorithm is correct with the next proposition.

**Proposition 9** (**Correctness of Algorithm 6**)
*Algorithm 6 solves the proximal problem of Eq. (3.3).*

### Implementation Details and Extensions

We describe hereafter several implementation details that are crucial to the efficiency of the algorithm:

- **Exploiting maximal connected components**: When there exists no arc between two subsets of $V$, it is possible to process them independently to solve the global min-cost flow problem.
  To that effect, before calling the function `computeFlow`$(V, E)$, we look for maximal connected components $(V_1, E_1), \ldots, (V_N, E_N)$ and call sequentially the procedure `computeFlow`$(V_i, E_i)$ for $i$ in $[\![1; N]\!]$.

- **Efficient max-flow algorithm**: We have implemented the "push-relabel" algorithm of (Goldberg and Tarjan, 1986) to solve our max-flow problems, using classical heuristics that significantly speed it up in practice (see (Goldberg and Tarjan,

1986; Cherkassky and Goldberg, 1997)). Our implementation uses the so-called "highest-active vertex selection rule, global and gap heuristics" (see (Goldberg and Tarjan, 1986; Cherkassky and Goldberg, 1997)), and has a worst-case complexity of $O(|V|^2|E|^{1/2})$ for a graph $(V, E, s, t)$. Note that this algorithm uses the concept of *pre-flow*that relaxes the definition of flows by allowing vertices to have a positive excess.

- **Using flow warm-restarts**: The main advantage of the push-relabel method for us is that it can be initialized with any valid pre-flow. Since we use this algorithm a large number of times (in the inner loop in our algorithm, and each time we call the proximal operator), we can easily use the value of the pre-flow obtained during the previous iteration to build a valid pre-flow for the current one.

- **Improved projection step**: The first line of the procedure `computeFlow` can be replaced by

$$\boldsymbol{\gamma} \leftarrow \arg\min_{\boldsymbol{\gamma}} \sum_{j \in V_u} \frac{1}{2}(\mathbf{u}_j - \boldsymbol{\gamma}_j)^2 \quad \text{s.t.} \quad \sum_{j \in V_u} \boldsymbol{\gamma}_j \leq \lambda \sum_{g \in V_{gr}} \eta_g \text{ and } |\boldsymbol{\gamma}_j| \leq \lambda \sum_{g \ni j} \eta_g.$$

  The idea is that the structure of the graph will not allow $\bar{\boldsymbol{\xi}}_j$ to be greater than $\lambda \sum_{g \ni j} \eta_g$ after the max-flow step. Adding these additional constraints leads to better performance when the graph is not well balanced. This modified projection step can still be computed in linear time (Brucker, 1984).

## 3.4 Computation of the Dual Norm

We show here how to compute efficiently the dual norm of $\Omega$, and use it here to monitor the convergence of the proximal method through a duality gap, and define a proper optimality criterion for problem (3.1). The duality gap can be derived from standard Fenchel duality arguments (Borwein and Lewis, 2006) and is equal to $f(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha}) + f^*(-\boldsymbol{\kappa})$, for $\boldsymbol{\alpha}, \boldsymbol{\kappa} \in \mathbb{R}^p$ with $\Omega^*(\boldsymbol{\kappa}) \leq \lambda$, where $f^*(\boldsymbol{\kappa}) \triangleq \sup_{\mathbf{z}}[\mathbf{z}^\top \boldsymbol{\kappa} - f(\mathbf{z})]$ is the Fenchel conjugate of $f$. Therefore, efficiently computing this duality gap requires efficiently evaluating $\Omega^*$, in order to find a feasible dual variable $\boldsymbol{\kappa}$. Similarly to the proximal operator, the computation of dual norm $\Omega^*$ can itself shown to solve another network flow problem, based on the following variational formulation, which extends a previous result from Jenatton et al. (2009):

**Lemma 11 (Dual Formulation of the Dual-norm $\Omega^\star$)**
*Let $\boldsymbol{\kappa} \in \mathbb{R}^p$. We have*

$$\Omega^*(\boldsymbol{\kappa}) = \min_{\boldsymbol{\xi} \in \mathbb{R}^{p \times |\mathcal{G}|}, \tau} \tau \quad s.t. \quad \sum_{g \in \mathcal{G}} \boldsymbol{\xi}^g = \boldsymbol{\kappa}, \ and \ \forall g \in \mathcal{G}, \ \|\boldsymbol{\xi}^g\|_1 \leq \tau\eta_g \ \ with \ \ \boldsymbol{\xi}^g_j = 0 \ if \ j \notin g. \tag{3.7}$$

From a graph viewpoint, let us parameterize the capacities on the arcs $(s, g)$, $g \in \mathcal{G}$, by $\tau\eta_g$, and fix the capacities on the arcs $(i, t)$, $i$ in $[\![1; p]\!]$, to $\boldsymbol{\kappa}_i$. The above problem

amounts to finding the smallest value of $\tau$, such that there exists a flow saturating the capacities $\boldsymbol{\kappa}_i$ on the arcs leading to the sink $t$ (i.e., $\bar{\boldsymbol{\xi}} = \boldsymbol{\kappa}$). The corresponding algorithm is presented below (see Algorithm 7), and is proven to be correct in the supplemental material.

**Algorithm 7** Computation of the dual norm

1: Inputs: $\boldsymbol{\kappa} \in \mathbb{R}^p$, a set of groups $\mathcal{G}$, positive weights $(\eta_g)_{g \in \mathcal{G}}$.
2: Build a graph $G_0 = (V_0, E_0, s, t)$ as explained in Section 3.3.
3: $\tau \leftarrow \texttt{dualNorm}(V_0, E_0)$.
4: **Return:** $\tau$ (value of the dual norm).

**Function** `dualNorm`$(V = V_u \cup V_{gr}, E)$

1: $\tau \leftarrow (\sum_{j \in V_u} \boldsymbol{\kappa}_j)/(\sum_{g \in V_{gr}} \eta_g)$ and set the capacities of arcs $(s, g)$ to $\tau \eta_g$ for all $g$ in $V_{gr}$.
2: Max-flow step: Update $(\bar{\boldsymbol{\xi}}_j)_{j \in V_u}$ by computing a max-flow on the graph $(V, E, s, t)$.
3: **if** $\exists\, j \in V_u$ s.t. $\bar{\boldsymbol{\xi}}_j \neq \boldsymbol{\kappa}_j$ **then**
4: Define $(V^+, E^+)$ and $(V^-, E^-)$ as in Algorithm 6, and set $\tau \leftarrow \texttt{dualNormAux}(V^-, E^-)$.
5: **end if**
6: **Return:** $\tau$.

We now prove that Algorithm 7 is correct.

**Proposition 10 (Convergence and correctness of Algorithm 7)** *Algorithm 7 computes the value of the dual norm of Eq. (3.7) in a finite and polynomial number of operations.*

The proof is given in Appendix B.

## 3.5 Applications and Experiments

We present in this section experiments to demonstrate the usefulness of our approaches, with experiments on natural image patches, background subtraction, and speed benchmarks.

### 3.5.1 Learning Hierarchical Dictionaries of Natural Image Patches

We start by presenting an experiment from Jenatton et al. (2010a), which learns a dictionary with a hierarchical structure. As far as we know, while much attention has been given to efficiently solving dictionary learning problems (Lee et al., 2007; Mairal et al., 2010b), there are few attempts in the literature to make the model richer by adding structure between dictionary elements (Kavukcuoglu et al., 2009). We propose to use the structured sparsity framework to embed the dictionary elements in a hierarchy.

Hierarchies of latent variables, typically used in neural networks and deep learning architectures (see Bengio, 2009, and references therein) have emerged as a natural struc-

ture in several applications, notably to model text documents. Indeed, in the context of *topic models* (Blei et al., 2003), hierarchical models using Bayesian non-parametric methods have been proposed by Blei et al. (2010). Quite recently, hierarchies have also been considered in the context of kernel methods (Bach, 2009). Structured sparsity has been used to regularize dictionary elements by Jenatton et al. (2010c), but to the best of our knowledge, it has never been used to model dependencies between them.

Let us now consider a set $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n] \in \mathbb{R}^{m \times n}$ of $n$ signals of dimension $m$. As in previous parts of this thesis, we want to learn a dictionary $\mathbf{D} = [\mathbf{d}^1, \ldots, \mathbf{d}^p] \in \mathbb{R}^{m \times p}$, and a matrix of *decomposition coefficients* $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n] \in \mathbb{R}^{p \times n}$, so that $\mathbf{x}^i \approx \mathbf{D}\boldsymbol{\alpha}^i$ for every signal $\mathbf{x}^i$, as measured by any convex loss, e.g., the square loss.

This leads to the following formulation,

$$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A} \in \mathcal{A}} \frac{1}{n} \sum_{i=1}^{n} \Big[ \frac{1}{2} \|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda \Omega(\boldsymbol{\alpha}^i) \Big], \tag{3.8}$$

where $\mathcal{A}$ and $\mathcal{C}$ denote two convex sets and $\Omega$ is a regularization term, usually a norm, whose effect is controlled by the regularization parameter $\lambda > 0$. For instance, the standard *sparse coding* formulation takes $\Omega$ to be the $\ell_1$ norm, $\mathcal{C}$ to be the set of matrices in $\mathbb{R}^{m \times p}$ whose columns are in the unit ball of the $\ell_2$ norm, with $\mathcal{A} = \mathbb{R}^{p \times n}$. However, this classical setting treats each dictionary element independently from the others, and does not exploit possible relationships between them. We address this potential limitation of the $\ell_1$ norm by embedding the dictionary in a tree structure, using the hierarchical norm presented in Section 3.2

Optimization for dictionary learning has already been presented in this thesis. We choose in this chapter the classical alternating between the variables $\mathbf{D}$ and $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n]$, i.e., minimizing over one while keeping the other one fixed, which yields good results in general, even though it can be further accelerated using online learning algorithms, as presented in Chapter 2.

The main difficulty of our problem lies essentially in the optimization of the vectors $\boldsymbol{\alpha}^i$, $i \in [\![1; n]\!]$ for $\mathbf{D}$ fixed, since $n$ may be large, and since it requires to deal with the nonsmooth regularization term $\Omega$. The optimization of the dictionary $\mathbf{D}$ (for $\mathbf{A}$ fixed) is in general easier, and we use Algorithm 2 presented in Section 2.3.1 to that effect.

This experiment studies whether a hierarchical structure can help dictionaries for denoising natural image patches, and in which noise regime the potential gain is significant. We aim at reconstructing *corrupted* patches from a test set, after having learned dictionaries on a training set of *non-corrupted* patches. Though not typical in machine learning, this setting is reasonable in the context of images, where lots of non-corrupted patches are easily available.[6]

We have extracted $100,000$ patches of size $m = 8 \times 8$ pixels from the Berkeley segmentation database of natural images (Martin et al., 2001) which contains a high

[6]Note that we study the ability of the model to reconstruct independent patches, and additional work is required to apply our framework to a full image processing task, where patches usually overlap (Elad and Aharon, 2006).

variability of scenes. We have then split this dataset into a training set $\mathbf{X}^{tr}$, a validation set $\mathbf{X}^{val}$, and a test set $\mathbf{X}^{te}$, respectively of size $50,000$, $25,000$, and $25,000$ patches. All the patches are centered and normalized to have unit $\ell_2$ norm.

The validation and test sets are corrupted by removing a certain percentage of pixels, the task being to reconstruct the missing pixels from the known pixels. We thus introduce for each element $\mathbf{x}$ of the validation/test set, a vector $\tilde{\mathbf{x}}$, equal to $\mathbf{x}$ for the known pixel values and 0 otherwise. In the same way, we define $\tilde{\mathbf{D}}$ as the matrix equal to $\mathbf{D}$, except for the rows corresponding to missing pixel values, which are set to 0. By decomposing $\tilde{\mathbf{x}}$ on $\tilde{\mathbf{D}}$, we obtain a sparse code $\boldsymbol{\alpha}$, and the estimate of the reconstructed patch is defined as $\mathbf{D}\boldsymbol{\alpha}$. Note that this procedure assumes that we know which pixel is missing and which is not for every element $\mathbf{x}$.

The parameters of the experiment are the regularization parameter $\lambda_{tr}$ used during the train step, the regularization parameter $\lambda_{te}$ used during the validation/test step, and the structure of the tree. For every reported result, these parameters have been selected by taking the ones offering the best performance on the *validation* set, before reporting any result from the *test* set. The values for the regularization parameters $\lambda_{tr}, \lambda_{te}$ were tested on a logarithmic scale $\{2^{-10}, 2^{-9}, \ldots, 2^2\}$, and then further refined on a finer logarithmic scale of factor $2^{-1/4}$. For simplicity reasons, we have chosen arbitrarily to use the $\ell_\infty$-norm in the structured norm $\Omega$, with all the weights equal to one. We have tested 21 balanced tree structures of depth 3 and 4, with different *branching factors* $p_1, p_2, \ldots, p_{d-1}$, where $d$ is the depth of the tree and $p_k$, $k \in \{1, \ldots, d-1\}$ is the number of children for the nodes at depth $k$. The branching factors tested for the trees of depth 3 where $p_1 \in \{5, 10, 20, 40, 60, 80, 100\}$, $p_2 \in \{2, 3\}$, and for trees of depth 4, $p_1 \in \{5, 10, 20, 40\}$, $p_2 \in \{2, 3\}$ and $p_3 = 2$, giving 21 possible structures associated with dictionaries with at most 401 elements. For each tree structure, we evaluated the performance obtained with the tree-structured dictionary along with the non-structured dictionary containing the same number of elements. These experiments were carried out four times, each time with a different initialization, and with a different noise realization. Quantitative results are reported on Table 3.1. For every number of

| noise | 50 % | 60 % | 70 % | 80 % | 90 % |
|---|---|---|---|---|---|
| flat | $19.3 \pm 0.1$ | $26.8 \pm 0.1$ | $36.7 \pm 0.1$ | $50.6 \pm 0.0$ | $72.1 \pm 0.0$ |
| tree | $18.6 \pm 0.1$ | $25.7 \pm 0.1$ | $35.0 \pm 0.1$ | $48.0 \pm 0.0$ | $65.9 \pm 0.3$ |

Table 3.1: Quantitative results of the reconstruction task on natural image patches. First row: percentage of missing pixels. Second and third rows: mean square error multiplied by 100, respectively for classical sparse coding, and tree-structured sparse coding.

missing pixels, the tree-structured dictionary outperforms the "unstructured one", and the most significant improvement is obtained in the noisiest setting. Note that having more dictionary elements is worthwhile when using the tree structure. To study the influence of the chosen structure, we have reported in Figure 3.4 the results obtained by the 14 tested structures of depth 3, along with those obtained with the unstructured dictionaries containing the same number of elements, when 90% of the pixels are missing.

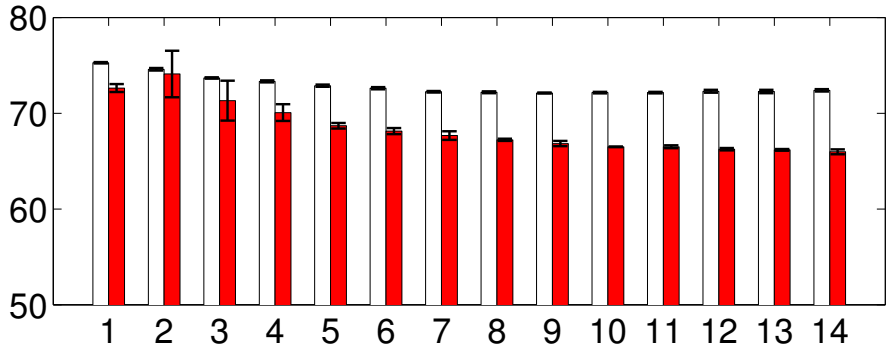



Figure 3.4: Mean square error multiplied by 100 obtained with 14 structures with error bars, sorted by number of dictionary elements. Red plain bars represents the tree-structured dictionaries. White bars correspond to the flat dictionary model containing the same number of dictionary as the tree-structured one. For readability purpose, the $y$-axis of the graph starts at 50.

For every number of dictionary elements, the tree-structured dictionary significantly outperforms the unstructured ones. An example of a learned tree-structured dictionary is presented in Figures 3.5 and 3.6. Dictionary elements naturally organize in groups of patches, with often low frequencies near the root of the tree, and high frequencies near the leaves. Dictionary elements tend to be highly correlated with their parents.

For more experiments on this hierarchical norm, the reader should refer to Jenatton et al. 2010a,b, where experiments on wavelets, speed benchmarks and experiments on text are presented. Interestingly, this type of method can indeed be used for modeling topics in text documents, and a bridge between *dictionary learning for sparse coding* and *hierarchical topic models* (Blei et al., 2010) can be established, which builds upon the interpretation of topic models as multinomial PCA (Buntine, 2002).

### 3.5.2 Speed Comparison of Algorithm 6 with Parametric Max-flow Algorithms

As shown by Hochbaum and Hong (1995), min-cost flow problems, and in particular, the dual problem of (3.3), can be reduced to a specific *parametric max-flow* problem. We thus compare our approach (ProxFlow) with the efficient parametric max-flow algorithm proposed by Gallo et al. (1989) and a simplified version of the latter proposed by Babenko and Goldberg (2006). We refer to these two algorithms as GGT and SIMP respectively. The benchmark is established on the same datasets as those already used in the experimental section of the chapter, namely: (1) three datasets built from over-complete bases of discrete cosine transforms (DCT), with respectively $10^4$, $10^5$ and $10^6$ variables, and (2) images used for the background subtraction task, composed of 57600 pixels. For GGT and SIMP, we use the `paraF` software which is a `C++` paramet-

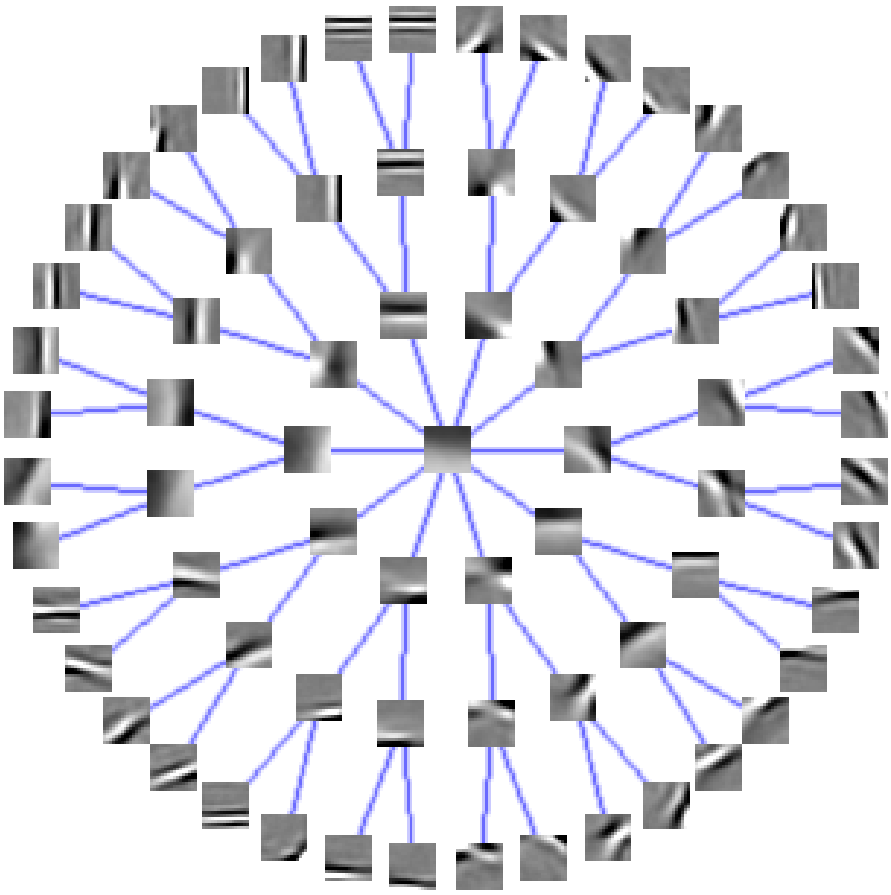


Figure 3.5: Learned dictionary with tree structure of depth 4. The root of the tree is in the middle of the figure. The branching factors are $p_1 = 10$, $p_2 = 2$, $p_3 = 2$. The dictionary is learned on $50,000$ patches of size $16 \times 16$ pixels.

ric max-flow implementation available at `http://www.avglab.com/andrew/soft.html`. Experiments were conducted on a single-core 2.33 Ghz.

We report in the following table the execution time in seconds of each algorithm, as well as the statistics of the corresponding problems:

| Number of variables $p$ | 10 000 | 100 000 | 1 000 000 | 57 600 |
|---|---|---|---|---|
| $\|V\|$ | 20 000 | 200 000 | 2 000 000 | 75 600 |
| $\|E\|$ | 110 000 | 500 000 | 11 000 000 | 579 632 |
| ProxFlow (in sec.) | **0.4** | **3.1** | **113.0** | **1.7** |
| GGT (in sec.) | 2.4 | 26.0 | 525.0 | 16.7 |
| SIMP (in sec.) | 1.2 | 13.1 | 284.0 | 8.31 |

Although we provide the speed comparison for a single value of $\lambda$ (the one used in the

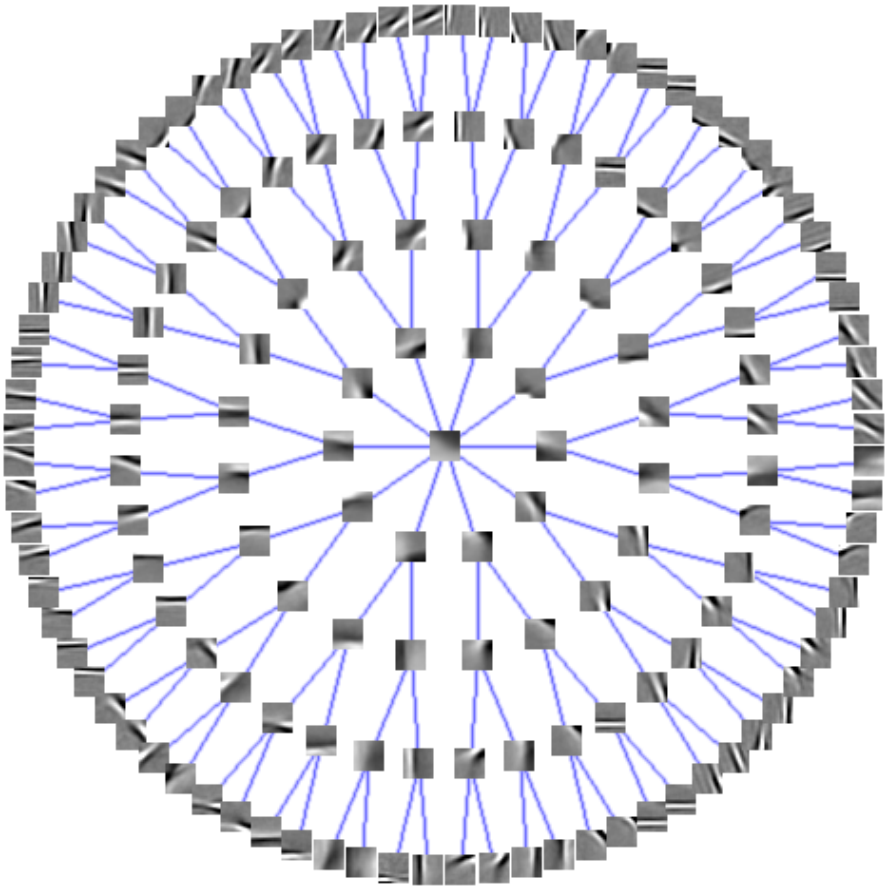


Figure 3.6: Learned dictionary with a tree structure of depth 5. The root of the tree is in the middle of the figure. The branching factors are $p_1 = 10$, $p_2 = 2$, $p_3 = 2$, $p_4 = 2$. The dictionary is learned on $50,000$ patches of size $16 \times 16$ pixels.

corresponding experiments of the chapter), we observed that our approach consistently outperforms GGT and SIMP for values of $\lambda$ corresponding to different regularization regimes.

### 3.5.3 Speed Comparison for Solving Structured Sparse Problems

Our experiments use the algorithm of (Beck and Teboulle, 2009) based on our proximal operator, with weights $\eta_g$ set to 1. We compare our method (ProxFlow) and two generic optimization techniques, namely a subgradient descent (SG) and an interior point method,[7] on a regularized linear regression problem. Both SG and ProxFlow are implemented in `C++`. Experiments are run on a single-core 2.8 GHz CPU. We consider a design matrix (dictionary) $\mathbf{D}$ in $\mathbb{R}^{m \times p}$ built from overcomplete dictio-

[7] In our simulations, we use the commercial software `Mosek`, `http://www.mosek.com/`

naries of discrete cosine transforms (DCT), which are naturally organized on one- or two-dimensional grids and display local correlations. The following families of groups $\mathcal{G}$ using this spatial information are thus considered: (1) every contiguous sequence of length 3 for the one-dimensional case, and (2) every $3\times3$-square in the two-dimensional setting. We generate vectors $\mathbf{y}$ in $\mathbb{R}^m$ according to the linear model $\mathbf{y} = \mathbf{D}\boldsymbol{\alpha}^0 + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, 0.01\|\mathbf{D}\boldsymbol{\alpha}^0\|_2^2)$. The vector $\boldsymbol{\alpha}^0$ has about 20% percent nonzero components, randomly selected, while respecting the structure of $\mathcal{G}$, and uniformly generated between $[-1, 1]$.

In our experiments, the regularization parameter $\lambda$ is chosen to achieve this level of sparsity. For SG, we take the step size to be equal to $a/(k+b)$, where $k$ is the iteration number, and $(a, b)$ are the best parameters selected in $\{10^{-3}, \ldots, 10\} \times \{10^2, 10^3, 10^4\}$. For the interior point methods, since problem (3.1) can be cast either as a quadratic (QP) or as a conic program (CP), we show in Figure 3.7 the results for both formulations. Our approach compares favorably with the other methods, on three problems of different sizes, $(n, p) \in \{(100, 10^3), (1024, 10^4), (1024, 10^5)\}$, see Figure 3.7. In addition, note that QP, CP and SG do not obtain sparse solutions, whereas ProxFlow does. We have also run ProxFlow and SG on a larger dataset with $(n, p) = (100, 10^6)$: after 12 hours, ProxFlow and SG have reached a relative duality gap of 0.0006 and 0.02 respectively.[8]

### 3.5.4 Background Subtraction

Following Cevher et al. (2009); Huang et al. (2009), we consider a background subtraction task. Given a sequence of frames from a fixed camera, we try to segment out foreground objects in a new image. If we denote by $\mathbf{y} \in \mathbf{R}^m$ this image composed of $m$ pixels, we model $\mathbf{y}$ as a sparse linear combination of $p$ other images $\mathbf{D} \in \mathbb{R}^{m\times p}$, plus an error term $\mathbf{e}$ in $\mathbf{R}^m$, i.e., $\mathbf{y} \approx \mathbf{D}\boldsymbol{\alpha} + \mathbf{e}$ for some sparse vector $\boldsymbol{\alpha}$ in $\mathbb{R}^p$. This approach is reminiscent of Wright et al. (2009a) in the context of face recognition, where $\mathbf{e}$ is further made sparse to deal with small occlusions. The term $\mathbf{D}\boldsymbol{\alpha}$ accounts for *background* parts present in both $\mathbf{y}$ and $\mathbf{D}$, while $\mathbf{e}$ contains specific, or *foreground*, objects in $\mathbf{y}$. The resulting optimization problem is $\min_{\boldsymbol{\alpha},\mathbf{e}} \frac{1}{2}\|\mathbf{y}-\mathbf{D}\boldsymbol{\alpha}-\mathbf{e}\|_2^2+\lambda_1\|\boldsymbol{\alpha}\|_1+\lambda_2\|\mathbf{e}\|_1$, with $\lambda_1, \lambda_2 \geq 0$. In this formulation, the $\ell_1$-norm penalty on $\mathbf{e}$ does not take into account the fact that neighboring pixels in $\mathbf{y}$ are likely to share the same label (background or foreground), which may lead to scattered pieces of foreground and background regions (Figure 3.8). We therefore put an additional structured regularization term $\Omega$ on $\mathbf{e}$, where the groups in $\mathbb{G}$ are all the overlapping $3 \times 3$-squares on the image. A dataset with hand-segmented evaluation images is used to illustrate the effect of $\Omega$.[9] For simplicity, we use a single regularization parameter, i.e., $\lambda_1 = \lambda_2$, chosen to maximize the number of pixels matching the ground truth. We consider $p = 200$ images with $m = 57600$ pixels (i.e., a resolution of $120\times 160$, times 3 for the RGB channels). As shown in Figure 3.8, adding $\Omega$ improves the background subtraction results for the two tested images, by removing the scattered

[8] Due to the computational burden, QP and CP could not be run on every problem.

[9] `http://research.microsoft.com/en-us/um/people/jckrumm/wallflower/testimages.htm`

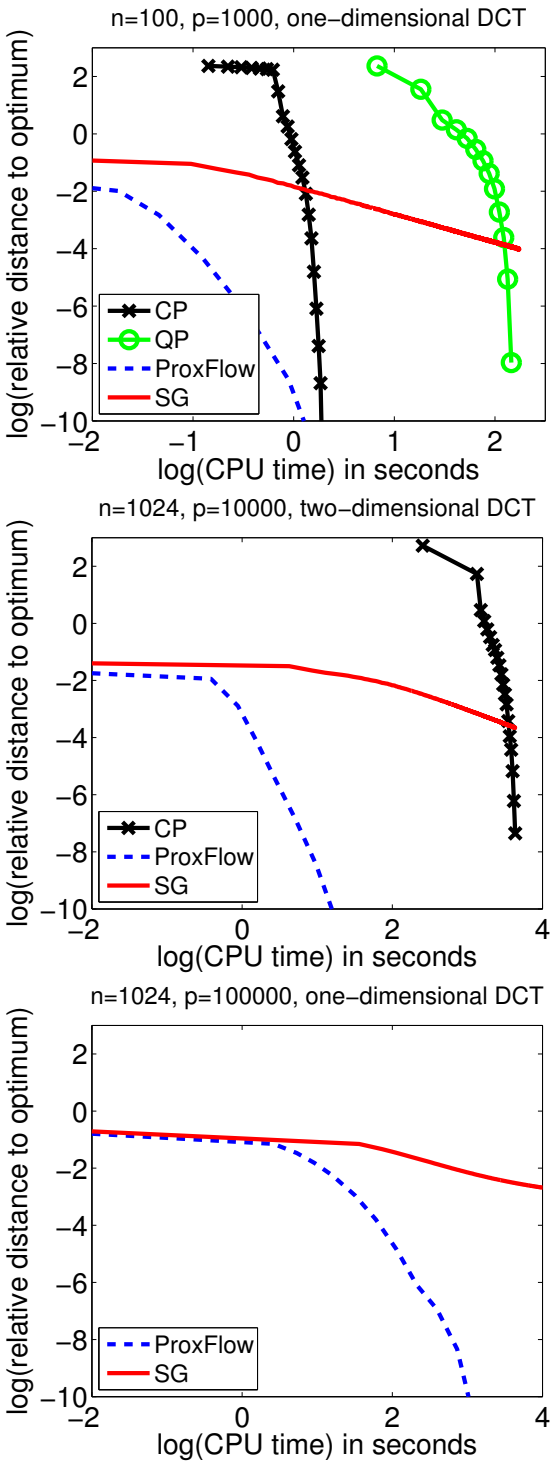



Figure 3.7: Speed comparisons: distance to the optimal primal value versus CPU time (log-log scale). Due to the computational burden, QP and CP could not be run on every problem.

artifacts due to the lack of structural constraints of the $\ell_1$-norm, which encodes neither spatial nor color consistency.

### 3.5.5 Multi-Task Learning of Hierarchical Structures

Jenatton et al. (2010a) have recently proposed to use a hierarchical structured norm to learn dictionaries of natural image patches. Following their work, we seek to represent $n$ signals $\{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ of dimension $m$ as sparse linear combinations of elements from a dictionary $\mathbf{D} = [\mathbf{d}^1, \ldots, \mathbf{d}^p]$ in $\mathbb{R}^{m \times p}$. This can be expressed for all $i$ in $[\![1; n]\!]$ as $\mathbf{x}^i \approx \mathbf{D}\boldsymbol{\alpha}^i$, for some sparse vector $\boldsymbol{\alpha}^i$ in $\mathbb{R}^p$. In (Jenatton et al., 2010a), the dictionary elements are embedded in a *predefined* tree $\mathcal{T}$, via a particular instance of the structured norm $\Omega$, which we refer to it as $\Omega_{\text{tree}}$, and call $\mathcal{G}$ the underlying set of groups. In this case, each signal $\mathbf{x}^i$ admits a sparse decomposition in the form of a subtree of dictionary elements.

Inspired by ideas from multi-task learning (Obozinski et al., 2009), we propose to learn the tree structure $\mathcal{T}$ by pruning irrelevant parts of a larger initial tree $\mathcal{T}_0$. We achieve this by using an additional regularization term $\Omega_{\text{joint}}$ across the different decompositions, so that subtrees of $\mathcal{T}_0$ will *simultaneously* be removed for all signals $\mathbf{y}^i$. In other words, the approach of Jenatton et al. (2010a) is extended by the following formulation:

$$\min_{\mathbf{D} \in \mathcal{D}, \mathbf{A}} \frac{1}{n} \sum_{i=1}^{n} \Big[ \frac{1}{2} \|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda_1 \Omega_{\text{tree}}(\boldsymbol{\alpha}^i) \Big] + \lambda_2 \Omega_{\text{joint}}(\mathbf{A}), \text{ s.t. } \|\mathbf{d}^j\|_2 \leq 1, \text{ for all } j \text{ in } [\![1; p]\!], \tag{3.9}$$

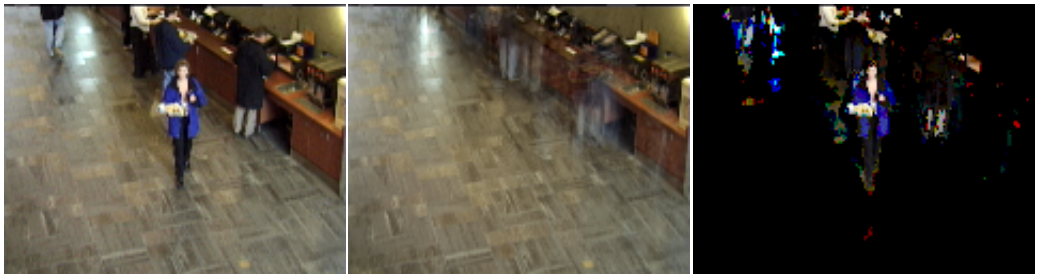
where $\mathbf{A} \triangleq [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n]$ is the matrix of decomposition coefficients in $\mathbb{R}^{p \times n}$. The new regularization term operates on the rows of $\mathbf{A}$ and is defined as $\Omega_{\text{joint}}(\mathbf{A}) \triangleq \sum_{g \in \mathcal{G}} \max_{i \in [\![1;n]\!]} \|\boldsymbol{\alpha}_g^i\|_\infty$.[10] The overall penalty on $\mathbf{A}$, which results from the combination of $\Omega_{\text{tree}}$ and $\Omega_{\text{joint}}$, is itself an instance of $\Omega$ with general overlapping groups, as defined in Eq (3.2).

To address problem (3.9), we use the same optimization scheme as Jenatton et al. (2010a), i.e., alternating between $\mathbf{D}$ and $\mathbf{A}$, fixing one variable while optimizing with respect to the other. The task we consider is the denoising of natural image patches, with the same dataset and protocol as Jenatton et al. (2010a). We study whether learning the hierarchy of the dictionary elements improves the denoising performance, compared to standard sparse coding (i.e., when $\Omega_{\text{tree}}$ is the $\ell_1$-norm and $\lambda_2 = 0$) and the hierarchical dictionary learning of Jenatton et al. (2010a) based on predefined trees (i.e., $\lambda_2 = 0$). The dimensions of the training set — 50 000 patches of size 8×8 for dictionaries with up to $p = 400$ elements — impose to handle extremely large graphs, with $|E| \approx |V| \approx 4.10^7$. Since problem (3.9) is too large to be solved exactly sufficiently many times to select the regularization parameters $(\lambda_1, \lambda_2)$ rigorously, we use the following heuristics: we optimize mostly with the currently pruned tree held fixed (i.e., $\lambda_2 = 0$), and only prune the tree (i.e., $\lambda_2 > 0$) every few steps on a random subset of 10 000 patches. We consider the same

[10] The simplified case where $\Omega_{\text{tree}}$ and $\Omega_{\text{joint}}$ are the $\ell_1$- and mixed $\ell_1/\ell_2$-norms (Yuan and Lin, 2006) corresponds to (Sprechmann et al., 2010).

(a) Input (b) images/background $\Omega + \ell_1$ (c) images/foreground $\ell_1$

(d) images/foreground $\Omega + \ell_1$ (e) images/foreground $\Omega + \ell_1$



(f) Input (g) images/background $\Omega + \ell_1$ (h) images/foreground $\ell_1$

(i) images/foreground $\Omega + \ell_1$ (j) images/foreground $\Omega + \ell_1$

Figure 3.8: Original images $\mathbf{y}$, the background (i.e., $\mathbf{D}\boldsymbol{\alpha}$) reconstructed by our method $\Omega + \ell_1$, and the images/foreground (i.e., the sparsity pattern of $\mathbf{e}$ as a mask on the original image) detected with $\ell_1$ and with $\ell_1 + \Omega$. We also show the results obtained on a different image. with the same values of $\lambda_1, \lambda_2$ as for the previous image. For the top image, the percentage of pixels matching the ground truth is 98.8% with $\Omega$, 87.0% without. As for the bottom image, the result is 93.8% with $\Omega$, 90.4% without.

hierarchies as Jenatton et al. (2010a), involving between 30 and 400 dictionary elements. The regularization parameter $\lambda_1$ is selected on the validation set of 25 000 patches, for both sparse coding (Flat) and hierarchical dictionary learning (Tree). Starting from the tree giving the best performance (in this case the largest one, see Figure 3.9), we solve problem (3.9) following our heuristics, for increasing values of $\lambda_2$. As shown in Figure 3.9, there is a regime where our approach performs significantly better than the two other compared methods. The standard deviation of the noise is 0.2 (the pixels have values in $[0, 1]$); no significant improvements were observed for lower levels of noise. A visual example of a pruned tree is given in Figure 3.10.

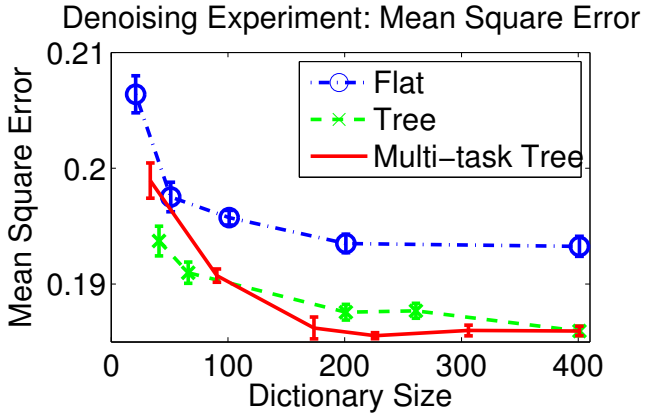



Figure 3.9: Mean square error versus dictionary size. The error bars represent two standard deviations, based on three runs.

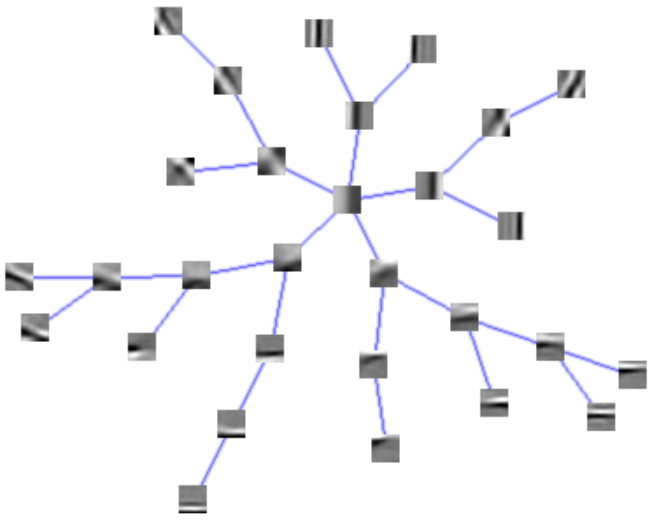


Figure 3.10: Hierarchy obtained by pruning a larger tree of 76 elements.

## 3.6 Conclusions

We have presented a new optimization framework for solving sparse structured problems involving sums of $\ell_\infty$-norms of any (overlapping) groups of variables. Interestingly, this sheds new light on connections between sparse methods and the literature of network flow optimization. In particular, the proximal operator for the formulation we consider can be cast as a quadratic min-cost flow problem, for which we propose an efficient and simple algorithm. This allows the use of accelerated gradient methods. Several experiments demonstrate that our algorithm can be applied to a wide class of learning problems, which have not been addressed before within sparse methods.

# 4

# Non-Local Sparse Models for Image Restoration

**Chapter abstract:** We propose in this work to unify two different approaches to image restoration: On the one hand, learning a basis set (dictionary) adapted to sparse signal descriptions has proven to be very effective in image reconstruction and classification tasks. On the other hand, explicitly exploiting the self-similarities of natural images has led to the successful non-local means approach to image restoration. We propose simultaneous sparse coding as a framework for combining these two approaches in a natural manner. This is achieved by jointly decomposing groups of similar signals on subsets of the learned dictionary. Experimental results in image denoising and demosaicking tasks with synthetic and real noise show that the proposed method outperforms the state of the art, making it possible to effectively restore raw images from digital cameras at a reasonable speed and memory cost.

This chapter is relatively independent from the previous one, and only requires reading Section 1.6 as a prerequisite. The material of this part is essentially based on the following work

J. Mairal, F. Bach, J. Ponce and G. Sapiro. Non-Local Sparse Models for Image Restoration. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 2009

## 4.1 Introduction

This work addresses the problem of reconstructing and enhancing an image given the noisy observations gathered by a digital camera sensor. Today, with advances in sensor design, the signal is relatively clean for digital SLRs at low sensitivities, but it remains noisy for consumer-grade and mobile-phone cameras at high sensitivities (low-light and/or high-speed conditions). The restoration problem is thus still of acute and in fact growing importance (e.g., Buades et al., 2005; Dabov et al., 2007; Elad and Aharon, 2006; Chatterjee and Milanfar, 2009; Lyu et al., 2004; Mairal et al., 2008b,d).

Working with noisy images recorded by digital cameras is difficult since different devices introduce different types of artefacts and spatial correlations in the noise as a result of internal post-processing (demosaicking, white balance, etc.). In this work, we operate directly on the raw sensor output, that suffers from non-homogeneous noise, but is less spatially correlated and not corrupted by post-processing artefacts. In turn, this requires demosaicking the raw signal—that is, reconstructing a full color image from the sensor's RGB (Bayer) pattern—a difficult problem in itself.

Whereas demosaicking is usually tackled using interpolation-based methods (Gunturk et al., 2002, 2005; Paliy et al., 2007; Zhang and Wu, 2005), much of the denoising effort has been aimed at finding a good model for natural images. Early work relied on various smoothness assumptions—such as anisotropic filtering (Perona and Malik, 1990), total variation (Rudin and Osher, 1994), or image decompositions on fixed bases such as wavelets (Mallat, 1999) for example. More recent approaches include non-local means filtering (Buades et al., 2005) and its variants (Kervrann and Boulanger, 2008) that exploit image self-similarities, learned sparse models (Elad and Aharon, 2006; Mairal et al., 2008b,d), Gaussian scale mixtures (Portilla et al., 2003), kernel regression (Takeda et al., 2007; Chatterjee and Milanfar, 2009), fields of experts (Roth and Black, 2005), and block matching with 3D filtering (BM3D) (Dabov et al., 2007).

We view both denoising and demosaicking as image reconstruction problems, and propose a novel image model that combines two now classical techniques into a single framework: The *non-local means* approach to image restoration explicitly exploits self-similarities in natural images (Buades et al., 2005; Efros and Leung, 1999) to average out the noise among similar patches, whereas *sparse coding* encodes natural image statistics by decomposing each image patch into a sparse linear combination of a few elements from a basis set called a *dictionary*.[1] Although fixed dictionaries based on various types of wavelets (Mallat, 1999) have been used in this setting, sparse decompositions based on learned, possibly overcomplete, dictionaries adapted to specific images have been shown to provide better results in practice (Elad and Aharon, 2006; Mairal et al., 2008b). We propose to extend and combine these two approaches by using *simultaneous sparse coding* (Tropp et al., 2006; Tropp, 2006; Turlach et al., 2005; Yuan and Lin, 2006; Obozinski et al., 2009) to impose that similar patches share the same dictionary elements in their sparse decomposition.

To the best of our knowledge, this is the first time that the corresponding models of image self-similarities are explicitly used in a common setting with sparse coding. There are a few recent works that are related to ours: The BM3D procedure of Dabov et al. (2007) exploits both self-similarities and sparsity for the denoising task, but it is based on classical, fixed orthogonal dictionaries. The *locally learned dictionaries* model of Chatterjee and Milanfar (2009) clusters similar patches together, builds orthonormal dictionaries for each cluster using principal component analysis, and use them in a kernel regression framework. Finally, the approach of Yu et al. (2010) simultaneously learns clusters of patches and orthonormal basis for each cluster with an EM algorithm. There

[1] The usage of the word "basis" is slightly abusive here since the elements of the dictionaries are not (a priori) necessarily independent.

is, in all these works and ours, a common principle that consists of jointly processing groups of patches, with richer and more complex data processing tools than just simple averaging, as it was done in the original nl-means method (Buades et al., 2005).

Experiments with images corrupted by synthetic or real noise show that the proposed method outperforms the state of the art in both image denoising and image demosaicking tasks, making it possible to effectively restore raw images from digital cameras at a reasonable speed and memory cost. Furthermore, although it is demonstrated on image denoising and demosaicking tasks in this work, our model is generic, admits straightforward extensions to various image and video restoration tasks such as inpainting (Bertalmio et al., 2000; Criminisi et al., 2004), and can adapt to a large class of data, e.g., multispectral images or MRI data. This model should also prove of interest in deblurring that have become the topic of much recent research (e.g., Puetter et al., 2005) with the emergence of computational photography.

### 4.1.1 Contributions

To summarize, this chapter makes two contributions

- It introduces a formulation to exploit both learned sparse coding and image self-similarities for image restoration.
- It shows that the proposed approach leads to state-of-the-art results for image denoising and image demosaicking, allowing restoring raw images from digital cameras.

This chapter is organized as follows: Section 4.2 presents several works that are related to ours. Section 4.3 is devoted to the approach we propose. Experiments are presented in Section 4.4 and Section 4.5 concludes the chapter.

## 4.2 Related Work

We start with a brief description of well-established approaches to image restoration that are relevant and related to the approach proposed in the next section. Since it is difficult to design a standard model for digital camera noise, these methods assume white Gaussian noise. Even though this generic setting slightly differs from that of real image denoising, it has allowed the development of effective algorithms that are now widely used in digital cameras and commercial software packages. We will use the same assumption in the rest of this work, but will demonstrate empirically that our approach is effective at restoring real images corrupted by non-Gaussian, non-uniform noise.

### 4.2.1 Non-Local Means Filtering

Efros and Leung (1999) showed that the self-similarities inherent to natural images could effectively be used in texture synthesis tasks. Following their insight, Buades et al. (2005) introduced the *non-local means* approach to image denoising, where the prominence of

self-similarities is used as a prior on natural images. This idea has in fact appeared in the literature in various guises and under different equivalent interpretations, e.g., kernel density estimation (Efros and Leung, 1999), Nadaraya-Watson estimators (Buades et al., 2005), mean-shift iterations (Awate and Whitaker, 2006), diffusion processes on graphs (Szlam et al., 2007), long-range random fields (Li and Huttenlocher, 2008), PAC-Bayesian estimation (Salmon and Pennec, 2009). Concretely, let us consider a noisy image written as a column vector $\mathbf{y}$ in $\mathbb{R}^n$, and denote by $\mathbf{y}_i$ the $i$-th pixel and by $\mathbf{y}^i$ the patch of size $m$ centered on this pixel for some appropriate size $m$. This approach exploits the simple but very effective idea that two pixels associated with similar patches $\mathbf{y}^i$ and $\mathbf{y}^j$ should have similar pixel values $\mathbf{y}_i$ and $\mathbf{y}_j$. Using $\mathbf{y}^i$ as an explanatory variable for $\mathbf{y}_i$ leads to the non-local means formulation, where the denoised pixel $\mathbf{x}_i$ is obtained by a weighted average (the corresponding Nadaraya-Watson estimator (Buades et al., 2005):

$$\mathbf{x}_i = \sum_{j=1}^{n} \frac{K_h(\mathbf{y}^i - \mathbf{y}^j)}{\sum_{l=1}^{n} K_h(\mathbf{y}^i - \mathbf{y}^l)} \mathbf{y}_j, \tag{4.1}$$

and $K_h$ is a Gaussian kernel of bandwidth $h$. Variants of this formulation have been proposed, which essentially vary in the strategy for choosing the kernel for a given pixel (Kervrann and Boulanger, 2008; Salmon and Pennec, 2009)

### 4.2.2 Learned Sparse Coding

Although most of this section has already been presented in Section 1.6.2, we recall briefly the concept of image denoising using learned dictionaries for self-containedness reasons, but the reader should refer to Section 1.6.2 for more details.

We suppose that there exists a dictionary $\mathbf{D}$ in $\mathbb{R}^{m \times p}$ such that denoising a patch $\mathbf{y}^i$ in $\mathbb{R}^m$ amounts to solving the sparse decomposition problem

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \|\boldsymbol{\alpha}\|_q \quad \text{s.t.} \quad \|\mathbf{y}^i - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \leq \varepsilon, \tag{4.2}$$

where $\mathbf{D}\boldsymbol{\alpha}$ is an estimate of the clean signal, and $\|\boldsymbol{\alpha}\|_q$ is a sparsity-inducing regularization term. This regularizer is associated with the $\ell_1$ norm when $q = 1$, leading to the well-known Lasso (Tibshirani, 1996) and basis pursuit (Chen et al., 1998) problems, and with the $\ell_0$ *pseudo norm* when $q = 0$. We refer to Section 1.6.2 for the choice of the parameter $\varepsilon$.

To improve the performance of pre-defined dictionaries, Elad and Aharon (2006) have proposed instead to *learn* a dictionary $\mathbf{D}$ adapted to image patches from the image at hand, and demonstrated that it leads to better empirical performance. For an image of size $n$, a dictionary in $\mathbb{R}^{m \times p}$ adapted to the $n$ overlapping patches of size $m$ (typically $m = 8 \times 8 \ll n$) associated with the image pixels, is learned by addressing the following optimization problem

$$\min_{\mathbf{D} \in \mathcal{D}, \mathbf{A}} \sum_{i=1}^{n} \|\boldsymbol{\alpha}^i\|_q \quad \text{s.t.} \quad \|\mathbf{y}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 \leq \varepsilon, \tag{4.3}$$

where $\mathcal{D}$ is the set of matrices in $\mathbb{R}^{m\times p}$ with unit $\ell_2$-norm columns, $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n]$ is a matrix in $\mathbb{R}^{p\times n}$, $\mathbf{y}^i$ is the $i$-th patch of the *noisy* image $\mathbf{y}$, $\boldsymbol{\alpha}^i$ is the corresponding code, and $\mathbf{D}\boldsymbol{\alpha}^i$ is the estimate of the denoised patch. Although dictionary learning is traditionally considered as extremely costly, online procedures such as (Mairal et al., 2010b, 2009a) make it possible to efficiently process millions of patches, allowing the use of large photographs and/or large image databases.

Once the dictionary $\mathbf{D}$ and codes $\boldsymbol{\alpha}^i$ have been learned, every pixel admits $m$ estimates (one per patch containing it), and its value can be computed by averaging these:

$$\mathbf{x} = \frac{1}{m}\sum_{i=1}^{n} \mathbf{R}^i \mathbf{D}\boldsymbol{\alpha}^i, \tag{4.4}$$

where $\mathbf{R}^i$ in $\mathbb{R}^{n\times m}$ is the binary matrix which places patch number $i$ at its proper position in the image.

### 4.2.3 Methods based on Processing Clusters of Patches

Dabov et al. (2007) have proposed a patch-based procedure called BM3D (Block Matching with 3D Filtering) that exploits image self-similarities and gives state-of-the-art results. As in (Elad and Aharon, 2006), they estimate the codes of overlapping patches and average the estimates. However, similar to non-local means filtering (Buades et al., 2005), they reconstruct patches by finding similar ones in the image (*block matching*), stacking them together into a 3D signal block, and denoising the block using hard or soft thresholding (Donoho, 1998) with a 3D orthogonal dictionary (*3D filtering*). In conjunction with a few heuristics, amely, using a combination of weighted averages of overlapping patches, Kaiser windows, and Wiener filtering to further improve results. this simple idea has proven to be very efficient and gives better results than regular non-local means. A key idea of our method is to implement a similar joint decomposition approach in the context of sparse coding with learned dictionaries, as explained in the next section.

A similar principle was used in a different way by Chatterjee and Milanfar (2009), but combined with the kernel regression framework introduced before for image processing (Takeda et al., 2007). Note that their procedure also exploits adapted dictionaries for each cluster, using principal component analysis. This is also related to the work of Yu et al. (2010), who have proposed a joint formulation for clustering the patches and learning a model for every cluster. Their method is based on an EM algorithm, and end up learning both clusters and principal components of each cluster.

## 4.3 Proposed Formulation

We show in this section how image self-similarities can be used to improve learned sparse models with *simultaneous sparse coding*, which encourages similar patches to admit similar sparse decompositions.

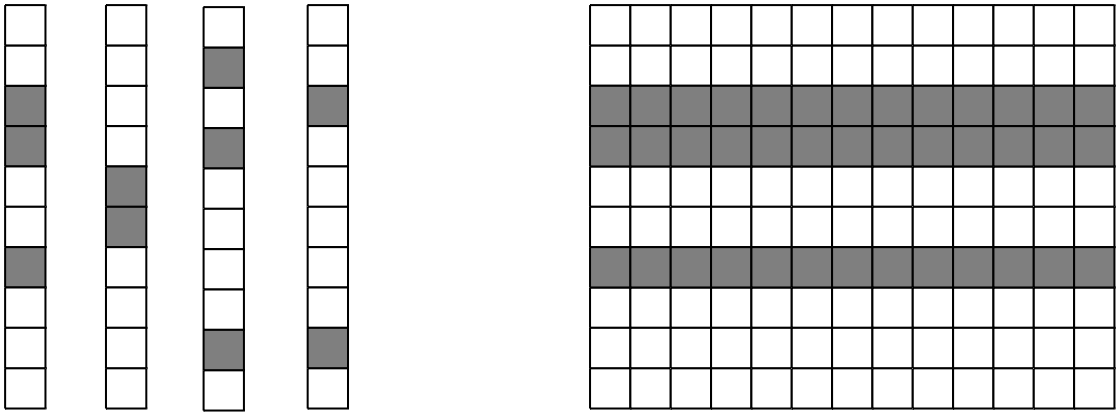


Figure 4.1: Sparsity vs. joint sparsity: Grey squares represents non-zeros values in vectors (left) or matrix (right).

### 4.3.1 Simultaneous Sparse Coding

A joint sparsity pattern—that is, a common set of nonzero coefficients—can be imposed to a set of vectors $\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^l$ through a *grouped-sparsity regularizer* on the matrix $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^l]$ in $\mathbb{R}^{p \times l}$ (Figure 4.1). This amounts to restricting the number of nonzero rows of $\mathbf{A}$, or replacing the $\ell_q$ vector (pseudo) norm in Eq. (4.3) by the $\ell_{q,r}$ (pseudo) matrix norm

$$\|\mathbf{A}\|_{q,r} \triangleq \sum_{i=1}^{p} \|\mathbf{A}_i\|_r^q, \tag{4.5}$$

where $\mathbf{A}_i$ denotes the $i$-th row of $\mathbf{A}$. In practice, one usually chooses for the pair $(q, r)$ the values $(1, 2)$ or $(0, \infty)$, the former leading to a convex norm, while the latter actually counts the number of nonzero rows and is only a pseudo norm (Tropp, 2004).

### 4.3.2 Principle of the Formulation

Non-local means filtering has proven very effective in general, but it fails in some cases. In the extreme, when a patch does not look like any other one in the image, it is impossible to exploit self-similarities to denoise the corresponding pixel value. Sparse image models can handle such situations by exploiting the redundancy between overlapping patches, but they suffer from another drawback: Similar patches sometimes admit very different estimates due to the potential instability of sparse decompositions (the $\ell_0$ pseudo norm is, after all, piecewise constant, and its $\ell_1$ counterpart is only piecewise differentiable), which can result in practice in noticeable reconstruction artefacts. In this chapter, we address this problem by forcing similar patches to admit similar decompositions. Concretely, let us define for each patch $\mathbf{y}^i$ the set $S_i$ of similar patches as

$$S_i \triangleq \{j = 1, \ldots, n \;\text{ s.t. }\; \|\mathbf{y}^i - \mathbf{y}^j\|_2^2 \leq \xi\}, \tag{4.6}$$

where $\xi$ is some threshold. Let us also consider for the moment a fixed dictionary $\mathbf{D}$ in $\mathbb{R}^{m\times p}$. Decomposing the patch $\mathbf{y}^i$ with a grouped-sparsity regularizer on the set $S_i$ amounts to solving

$$\min_{\mathbf{A}^i} \|\mathbf{A}^i\|_{q,r} \quad \text{s.t.} \quad \sum_{j\in S_i} \|\mathbf{y}^j - \mathbf{D}\boldsymbol{\alpha}^{i,j}\|_2^2 \leq \varepsilon_i, \tag{4.7}$$

where $\mathbf{A}^i = [\boldsymbol{\alpha}^{i,j}]_{j\in\mathcal{S}_i} \in \mathbb{R}^{p\times|S_i|}$. We adopt the same strategy as in Section 4.2.2 to choose $\varepsilon_i$ accordingly to the size of $S_i$: $\varepsilon_i = \sigma^2 F^{-1}_{m|S_i|}(\tau)$. In the $\ell_{1,2}$-case, this optimization problem is convex and can be solved efficiently (Friedman et al., 2007). In the $\ell_{0,\infty}$ case, on the other hand, it is intractable, and a greedy approach such as simultaneous orthogonal matching pursuit (Tropp, 2004) must be used to obtain an approximate solution.

In the framework of learned sparse coding, adapting $\mathbf{D}$ to the image(s) of interest naturally leads to the following optimization problem

$$\min_{(\mathbf{A}^i)_{i=1}^n, \mathbf{D}\in\mathcal{D}} \sum_{i=1}^n \frac{\|\mathbf{A}^i\|_{q,r}}{|S_i|^q} \quad \text{s.t.} \quad \forall i \sum_{j\in S_i} \|\mathbf{y}^j - \mathbf{D}\boldsymbol{\alpha}^{i,j}\|_2^2 \leq \varepsilon_i \tag{4.8}$$

where $\mathbf{D}$ is in $\mathbb{R}^{m\times p}$ with unit $\ell_2$-norm columns. The normalization by $|S_i|^q$ is used to ensure equal weights for all groups (as before, we only consider the cases where $(q,r)$ is $(1,2)$ or $(0,\infty)$). As noted in Section 1.6, in classical learned sparse coding, we prefer the $\ell_1$ norm for learning the dictionary and the $\ell_0$ pseudo norm for the final reconstruction. We adopt here a similar choice: We use the convex $\ell_{1,2}$ norm for learning the dictionary, which can be done efficiently (Mairal et al., 2010b), and we use the $\ell_{0,\infty}$ pseudo-norm for the final reconstruction. As in (Elad and Aharon, 2006), this formulation allows all the image patches to be processed as if they were *independent* of each other. To reconstruct the final image, we average the estimates of each pixel,

$$\mathbf{x} = \operatorname{diag}\Big(\sum_{i=1}^n \sum_{j\in S_i} \mathbf{R}^j \mathbf{1}_m\Big)^{-1} \sum_{i=1}^n \sum_{j\in S_i} \mathbf{R}^j \mathbf{D}\boldsymbol{\alpha}^{i,j}, \tag{4.9}$$

where $\mathbf{R}^j$ is defined as in Eq. (4.4) and $\mathbf{1}_m$ is a vector of size $m$ filled with ones. The term on the left is a scaling diagonal matrix, counting the number of estimates for each pixel. Note that when $\mathcal{S}_i = \{i\}$, our formulation is equivalent to regular learned sparse coding.

At first sight, the proposed technique may seem particularly costly, since decomposing a single patch requires solving a large-scale optimization problem (4.7). Similar concerns hold for the original formulations of non-local means (Buades et al., 2005) and BM3D (Dabov et al., 2007). As in these cases, slight changes to our approach are sufficient to make it efficient.

### 4.3.3 Practical Formulation and Implementation

The computational cost of the optimization problem (4.8) is dominated by the computation of the vectors $\boldsymbol{\alpha}^{i,j}$. In the worst case scenario, $n^2$ of these vectors have to be

computed. We show in the rest of this section how to modify our original formulation in order to make this number linear in $n$ and allow efficient optimization.

**Semi-local grouping.** When building $S_i$, one can restrict the search for patches similar to $\mathbf{y}^i$ to a window of size $w \times w$. This semi-local approach is also used by Dabov et al. (2007), and it reduces the worst-case number of vectors $\boldsymbol{\alpha}^{i,j}$ to $nw^2$. In practice, we never use $w$ greater than 64 in this chapter.

**Clustering.** It is also possible to cluster pixels into disjoint groups $C_k$ such that all pixels $i$ in $C_k$ share the same set $S_i$. The optimization problems (4.7) associated with all pixels in the same cluster are identical, further reducing the overall computational cost: In fact, only $n$ vectors $\boldsymbol{\alpha}^{i,j}$ are computed in this case since each pixel belongs to exactly one cluster. This is a key ingredient to the efficiency of our implementation. Other strategies are also possible, allowing a few clusters to overlap for instance.

**Initialization of D.** One important asset of sparse representations is that they can benefit from dictionaries learned offline on a database of natural images, which can be used as a good initial dictionaries for the denoising procedure (Elad and Aharon, 2006). Using the online procedure of Mairal et al. (2010b, 2009a), our initial dictionaries are learned on $2\times10^7$ patches of natural images taken randomly from the 10 000 images of the PASCAL VOC'07 database. As shown in the next section, using this online procedure and such a large training sample has led to a significant performance improvement compared to methods such as (Mairal et al., 2008b) that use batch learning methods such as K-SVD (Aharon et al., 2006) and are unusable with such large-scale data.

**Improved matching.** Following Dabov et al. (2007), we have noticed that better groups of similar patches can be found by using a first round of denoising on the patches (using, for example, the classical sparse coding approach of Eq. (4.3) presented in the previous section) before grouping them. In turn, as shown by our experiments, our simultaneous sparse coding approach greatly improves on this initial denoising step.

**Patch normalization.** To improve the numerical stability of sparse coding, the mean intensity (or RGB color) value of a patch is often subtracted from all its pixel values before decomposing it, then added back to the estimated values (Elad and Aharon, 2006). We have adopted this approach in our implementation, and our experiments have shown that it improves the visual quality of the results.

**Reducing the memory cost.** At first sight, Eq. (4.8) requires storing a large number of codes $\boldsymbol{\alpha}^{i,j}$. Even though these are sparse, and their number can be reduced to the number of pixels using the clustering strategy presented above, this could potentially be a problem for large images. In fact, only a small subset of the vectors $\boldsymbol{\alpha}^{i,j}$ is stored at any given time: The online procedure of Mairal et al. (2010b, 2009a) computes them *on the fly* and does not require storing them to learn the dictionary. In the case of Eq. (4.8), the maximum number of vectors $\boldsymbol{\alpha}^{i,j}$ that have to be stored at any given time is the size of the largest cluster of similar patches.

### 4.3.4 Real Images and Demosaicking

Single-chip digital cameras do not capture a noisy RGB signal at each pixel. Instead, combined with a red (R), green (G), or blue (B) filter, the sensor associated with each

pixel integrates the incoming light flux over the corresponding frequency range and a short period of time. The relation between the pixels and the color information they record is obtained through a specific pattern, the most famous one being the Bayer pattern, G-R-G-R on odd lines and B-G-B-G on even ones. The demosaicking problem consists of reconstructing the whole color image given the sensor measurements. Although most of the approaches found in the literature to solve this problem are based on interpolation (Gunturk et al., 2002, 2005; Paliy et al., 2007; Zhang et al., 2008), the image models investigated in this chapter have also been used for demosaicking: Self-similarities have been exploited by Buades et al. (2009), and learned sparse coding has been used by Mairal et al. (2008b). We adapt here Mairal et al. (2008b) to our simultaneous sparse coding framework. First we learn an initial dictionary $\mathbf{D}^0$ using our online dictionary learning algorithm on a database of natural color images. Our demosaicking procedure can then be decomposed into four simple steps:
(1) Cluster similar patches on the mosaicked image $\mathbf{y}$.
(2) Reconstruct each patch using $\mathbf{D}^0$, addressing for all $i$

$$\min_{\mathbf{A}^i \in \mathbb{R}^{p \times |S_i|}} \|\mathbf{A}^i\|_{0,\infty} \quad \text{s.t.} \quad \forall j \ \mathbf{M}^j(\mathbf{y}^j - \mathbf{D}^0 \boldsymbol{\alpha}^{i,j}) = 0, \tag{4.10}$$

where $\mathbf{M}^j$ is a binary masked corresponding to the Bayer pattern of measured values, and average the reconstructions to obtain an estimate $\mathbf{x}$ of the demosaicked image.
(3) Learn a dictionary $\mathbf{D}^1$ for $\mathbf{x}$ with a strong regularization—that is, replace $\mathbf{y}$ by $\mathbf{x}$ in Eq. (4.8), solving this equation with a large value for $\varepsilon_i$.
(4) Reconstruct each patch using $\mathbf{D}^2 = [\mathbf{D}^0\ \mathbf{D}^1]$ instead of $\mathbf{D}^0$ in Eq. (4.10), and average the estimates using Eq. (4.9) to obtain the final demosaicked image.

As shown in the next section, this procedure outperforms the state of the art from quantitative and qualitative points of view. The raw mosaicked signal of digital cameras in low-light, short-exposure settings is noisy. It should therefore be denoised before demosaicking is attempted. Since our denoising procedure is generic and does not necessary assume the input data to be natural images, *the denoising procedure can be performed on the mosaicked image itself.*

## 4.4 Experimental Validation

We present in this section experiments on image denoising with synthetic noise and image demosaicking, before processing raw images from a digital camera, corrupted by real noise.

### 4.4.1 Denoising – Synthetic Noise

Experiments on denoising with synthetic white Gaussian noise have carried out with 12 standard benchmark images. The parameters used in this experiment are $p = 512$, $m = 9 \times 9$ for $\sigma \leq 25$, $m = 12 \times 12$ for $\sigma = 50$ and $m = 16 \times 16$ for $\sigma = 100$. The value of $\tau$ is chosen a bit more conservatively than in Mairal et al. (2008b) and is set to 0.8, while $\xi$

is chosen according to an empirical rule, $\xi = (32\sigma)^2/m$ for images scaled between 0 and 255, which has shown to be appropriate in all of our denoising experiments for both real and synthetic noise. Peak signal-to-noise ratio (PNSR) is used as performance measure in our quantitative evaluation.[2] Table 4.1 reports the results obtained on each image for different values of the (known) standard deviation of the noise $\sigma$, and Table 4.2 compares the average PSNR on these images obtained by several state-of-the-art image denoising methods—namely GSM (Portilla et al., 2003), FoE (Roth and Fischer, 2008), K-SVD (Elad and Aharon, 2006) and BM3D (Dabov et al., 2007)—with our method in three settings: SC (sparse coding) uses a fixed dictionary learned on a database of natural images without grouping the patches. It is therefore similar to the *global* approach to denoising of (Elad and Aharon, 2006). The only differences are that we have used our online learning procedure to learn the dictionary from $2 \times 10^7$ natural image patches instead of the $10^5$ patches used by Elad and Aharon (2006), and we have used an $\ell_1$ regularizer instead of an $\ell_0$ one to learn the dictionary. In the second setting (LSC, for learned sparse coding), the dictionary is adapted to the test image, again using an $\ell_1$ regularizer, which is similar to the *adaptive* approach of Elad and Aharon (2006) except for our (better) initial dictionary and their $\ell_0$ regularizer. The last setting (LSSC, for learned simultaneous sparse coding) adds a grouping step and uses the full power of our simultaneous sparse coding framework. These PSNR comparisons show that our model leads to better performance than the state-of-the-art techniques in general, and is always at least as good as BM3D, the top performer among those, especially for high values of $\sigma$. Additional qualitative examples are given in Figure 4.2 and 4.3.

Note that the parameters have not been optimized for speed but for quality in these experiments. On a recent Intel Q9450 2.66Ghz CPU, it takes for instance 0.5s to denoise the $256 \times 256$ image peppers with $\sigma = 25$ and the setting SC, 85s with LSC, and 220s with LSSC. With parameters optimized for speed ($k = 256$, fewer iterations in the dictionary learning procedure), the computation times become respectively 0.25s for SC, 10s for LSC, and 21s for LSC, and the final results' quality only drops by 0.05dB, which is visually imperceptible. Our framework is therefore flexible in terms of speed/quality compromise.

### 4.4.2 Demosaicking

We have used the standard Kodak PhotoCD benchmark to evaluate the performance of our demosaicking algorithm. This dataset consists of 24 RGB images of size $512 \times 768$ to which a Bayer mask has been applied. Ground truth is thus available, allowing quantitative comparisons. We have arbitrarily tuned the parameters of our method to optimize its performance on the 5 last images, choosing $p = 256$ (dictionary size), $m = 8 \times 8$ (patch size), and $\xi = 3 \times 10^4$ (for images scaled between 0 and 255). These parameters have been used for all 24 photos.

[2] Denoting by MSE the mean-squared-error for images whose intensities are between 0 and 255, the PSNR is defined as $\text{PSNR} = 10 \log_{10}(255^2/\text{MSE})$ and is measured in dB. A gain of 1dB reduces the MSE by approximately 20%.

| $\sigma$ | 5 | 10 | 15 | 20 | 25 | 50 | 100 |
|---|---|---|---|---|---|---|---|
| house | 39.93 | 36.96 | 35.35 | 34.16 | 33.15 | 30.04 | 25.83 |
| peppers | 38.18 | 34.80 | 32.82 | 31.37 | 30.21 | 26.62 | 23.00 |
| camera. | 38.32 | 34.21 | 32.01 | 30.57 | 29.51 | 26.42 | 23.08 |
| lena | 38.69 | 35.83 | 34.15 | 32.90 | 31.87 | 28.87 | 25.82 |
| barbara | 38.48 | 34.97 | 33.00 | 31.57 | 30.47 | 27.06 | 23.59 |
| boat | 37.35 | 34.02 | 32.20 | 30.89 | 29.87 | 26.74 | 23.84 |
| hill | 37.17 | 33.67 | 31.89 | 30.71 | 29.80 | 27.05 | 24.44 |
| couple | 37.45 | 33.98 | 32.06 | 30.69 | 29.61 | 26.30 | 23.28 |
| man | 37.89 | 34.06 | 32.01 | 30.64 | 29.63 | 26.69 | 24.00 |
| fingerp. | 36.70 | 32.57 | 30.31 | 28.78 | 27.62 | 24.25 | 21.26 |
| bridge | 35.78 | 31.22 | 28.92 | 27.46 | 26.42 | 23.68 | 21.46 |
| flintst. | 36.13 | 32.46 | 30.78 | 29.63 | 28.71 | 25.16 | 21.10 |
| **Av.** | 37.67 | 34.06 | 32.12 | 30.78 | 29.74 | 26.57 | 23.39 |

Table 4.1: Quantitative denoising experiments on 12 standard images. The PSNR values are averaged over 5 experiments with 5 different noise realizations and values of $\sigma$ between 5 and 100. The variance is negligible and not reported due to space limitations.

| $\sigma$ | GSM | FOE | K-SVD | BM3D | SC | LSC | LSSC |
|---|---|---|---|---|---|---|---|
| 5 | 37.05 | 37.03 | 37.42 | 37.62 | 37.46 | 37.66 | **37.67** |
| 10 | 33.34 | 33.11 | 33.62 | 34.00 | 33.76 | 33.98 | **34.06** |
| 15 | 31.31 | 30.99 | 31.58 | 32.05 | 31.72 | 31.99 | **32.12** |
| 20 | 29.91 | 29.62 | 30.18 | 30.73 | 30.29 | 30.60 | **30.78** |
| 25 | 28.84 | 28.36 | 29.10 | 29.72 | 29.18 | 29.52 | **29.74** |
| 50 | 25.66 | 24.36 | 25.61 | 26.38 | 25.83 | 26.18 | **26.57** |
| 100 | 22.80 | 21.36 | 22.10 | 23.25 | 22.46 | 22.62 | **23.39** |

Table 4.2: Quantitative comparative evaluation. We compare our algorithm to GSM (Portilla et al., 2003), FoE (Roth and Fischer, 2008), K-SVD (Elad and Aharon, 2006) and BM3D (Dabov et al., 2007), that were the top performers so far on this benchmark, and whose implementations are available online. The PSNR is chosen as before as performance measure. Best results are in bold.

Figure 4.2: Left: noisy images. Middle: Original images. Right: restored images. From top to bottom: house image ($\sigma = 15$), man image ($\sigma = 50$), hill image ($\sigma = 20$), barbara image ($\sigma = 100$). Note that we reproduce the original brick texture in the house image and the hair texture for the man image both hardly visible in the noisy images.

Figure 4.3: Left: noisy images. Middle: Original images. Right: restored images. From top to bottom: peppers image ($\sigma = 20$), lena image ($\sigma = 10$), couple image ($\sigma = 15$), bridge image ($\sigma = 10$).

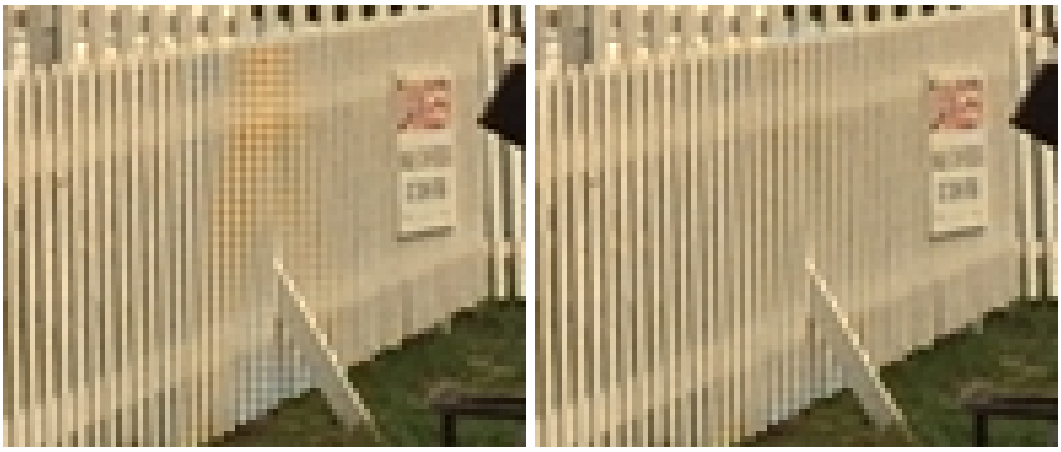


Figure 4.4: Left: Demosaicking with LSC sometimes causes artefacts such as the yellow and blue pixels in the middle of the fence. Right: The reconstruction obtained with the LSSC algorithm does not exhibit such artefacts. (This figure should be viewed in color.)

We evaluate the performance of the three variants SC, LSC, LSSC of our framework defined in the previous subsection, and compare them with the state of the art using the experimental protocol of Paliy et al. (2007) whose LPA method is, to the best of our knowledge, the top performer so far in terms of PSNR (or equivalently mean-squared error) on the Kodak PhotoCD benchmark. Following Paliy et al. (2007), we have excluded a 15-pixel border in fairness to methods that are susceptible to boundary effects. Table 4.3 adds our results to those reported by Paliy et al. (2007) for each one of the 24 photos. The proposed LSSC method outperforms the state-of-the-art algorithms AP (Gunturk et al., 2002), DL (Zhang and Wu, 2005) and LPA (Paliy et al., 2007) by a significant margin of 0.87dB even though our formulation is generic and not tuned to the task of demosaicking, demonstrating the promise of our image model.

When including the image border so as to be able to compare our results with those of Mairal et al. (2008b), it is interesting to note that, in the SC setting, we achieve a mean PSNR of 40.72dB on the 24 images, compared to the 39.56dB of Mairal et al. (2008b). Clearly, it is thus preferable in this case to learn the dictionary from a large dataset of natural images. With LSC, we achieve a mean PSNR of 40.98dB, compared to the 40.32dB of Mairal et al. (2008b), reaching a mean PSNR of 41.24dB with LSSC. Although this quantitative improvement may seem small, it is qualitatively quite significant. Even though SC and LSC perform very well in terms of PSNR, they suffer from classical demosaicking artefacts, as shown by the example of Figure 4.4. On the other hand, our new LSSC model, which exploits self-similarities as well as learned sparse coding, is usually free of most of these artefacts.

| Im. | AP | DL | LPA | SC | LSC | LSSC |
|---|---|---|---|---|---|---|
| 1 | 37.84 | 38.46 | 40.47 | 40.84 | 40.92 | **41.36** |
| 2 | 39.64 | 40.89 | 41.36 | 41.76 | 42.03 | **42.24** |
| 3 | 41.40 | 42.66 | 43.47 | 43.15 | 43.92 | **44.24** |
| 4 | 39.92 | 40.49 | 40.84 | 41.99 | 42.14 | **42.45** |
| 5 | 37.28 | 38.07 | 37.51 | 38.72 | 39.15 | **39.45** |
| 6 | 38.69 | 40.19 | 40.92 | 41.29 | 41.36 | **41.71** |
| 7 | 41.75 | 42.35 | 43.06 | 43.30 | 43.59 | **44.06** |
| 8 | 35.58 | 36.02 | 37.13 | 37.42 | 37.38 | **37.57** |
| 9 | 41.84 | 43.05 | 43.50 | 43.17 | 43.74 | **43.83** |
| 10 | 41.93 | 42.54 | 42.77 | 43.01 | 43.17 | **43.33** |
| 11 | 39.25 | 40.01 | 40.51 | 41.19 | 41.29 | **41.51** |
| 12 | 42.62 | 43.45 | 44.01 | 44.29 | 44.49 | **44.90** |
| 13 | 34.28 | 34.75 | 36.08 | 36.16 | 36.29 | **36.35** |
| 14 | 35.66 | 36.91 | 36.86 | 37.64 | 38.48 | **38.77** |
| 15 | 39.17 | 39.82 | 40.09 | 41.04 | 41.24 | **41.74** |
| 16 | 42.10 | 43.75 | 44.02 | 44.36 | 44.42 | **44.91** |
| 17 | 41.23 | 41.68 | 41.75 | 41.75 | 41.86 | **41.98** |
| 18 | 37.31 | 37.64 | 37.59 | 38.05 | 38.27 | **38.38** |
| 19 | 39.99 | 41.01 | 41.55 | 41.58 | 41.71 | **42.31** |
| 20 | 40.63 | 41.24 | 41.48 | 41.95 | 42.25 | **42.27** |
| 21 | 38.72 | 39.10 | 39.61 | 40.55 | 40.59 | **40.65** |
| 22 | 37.63 | 38.37 | 38.44 | 38.73 | 38.97 | **39.24** |
| 23 | 41.93 | 43.22 | 43.92 | 43.47 | 43.93 | **44.34** |
| 24 | 34.74 | 35.55 | 35.44 | 35.59 | 35.85 | **35.89** |
| **Av.** | 39.21 | 40.05 | 40.52 | 40.88 | 41.13 | **41.39** |

Table 4.3: Comparison of demosaicking performance in terms of PSNR between AP (Gunturk et al., 2002), DL (Zhang and Wu, 2005), LPA (Paliy et al., 2007) and the SC, LSC and LSSC variants of our method. Best results are in bold.

### 4.4.3 Denoising – Real Noise

To evaluate qualitatively our denoising method on real images, we have taken three RAW photographs using a Canon Powershot G9 digital camera at 1600 ISO with a short time exposure. At such a setting, the images are quite noisy. We have extracted the mosaicked data from the RAW image using the open-source *dcraw* software. We have then scaled manually the R,G,B channels so that they visually appear to contain similar amounts of noise. At this point, the noise is, to a first approximation, roughly uniform, and we apply our denoising algorithm to the scaled mosaicked image, before performing demosaicking, white balance, sRGB space conversion, gamma correction, and contrast enhancement to reconstruct the final image. This approach has proven experimentally to lead to better results than denoising each R,G,B channel independently. Of course,

assuming that the noise is uniform is only a rough approximation. Non-spatially uniform noise models are available for specific cameras, and exploited by commercial software packages such at those discussed later in this section. Incorporating these models into our framework is feasible, following Mairal et al. (2008b), but beyond the scope of this chapter. Instead, we demonstrate that, even with a uniform assumption, our algorithm is qualitatively competitive with top-of-the-line commercial denoising software.

The parameters we have used are a patch size of $m = 8 \times 8$ pixels, and $p = 256$ dictionary elements, which is typical for sparse coding methods (Elad and Aharon, 2006; Mairal et al., 2008b). The noise level $\sigma$ is estimated by the user and assumed to be uniform across the image, and $\xi$ is chosen according to the empirical rule presented in Section 4.4.1. Demosaicking is performed using the same parameters as in Section 4.4.2. Figures 4.5, 4.6 and 4.7 compare closeups of the images reconstructed from the RAW file by the camera itself (jpeg output), the image obtained with Adobe Camera Raw 5.0 (no denoising), two state-of-the-art denoising softwares NoiseWare 4.2 and the DxO Optics Pro 5.3 package, and our method. The commercial programs have been run with their default parameters, and these could certainly be further tuned to improve image quality a bit.[3] However, note that, unlike ours, these programs do take advantage of a detailed, non-uniform noise model specific to the camera, yet do not appear to give qualitatively better results. Although a quantitative comparison is not possible, we believe (subjectively) that our method does best on the first and third images, while DxO Optics Pro is slightly better for the second one. As in our previous experiments, LSSC suffers from fewer artefacts than LSC in general. The noise's non-uniformity does not seem to affect our results much, except perhaps for the background of the third image, where part of the noise is reconstructed.

## 4.5 Conclusion

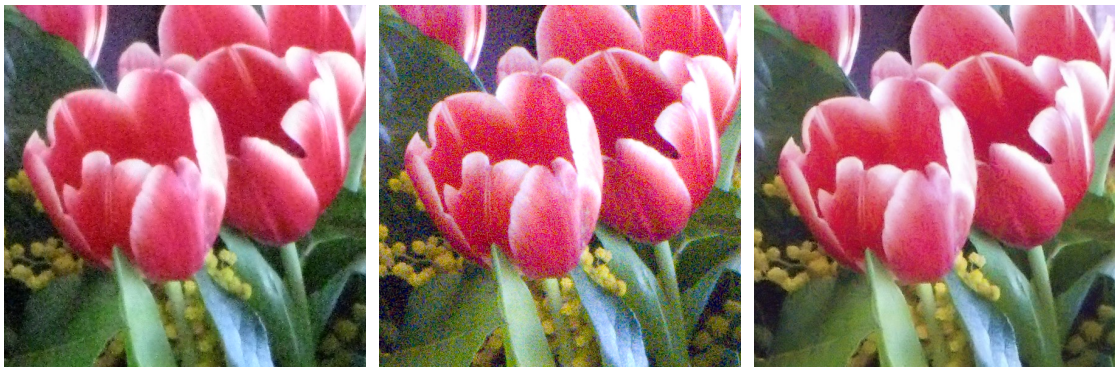
We have proposed in this work a new image model that combines the non-local means and sparse coding approaches to image restoration into a unified framework where similar patches are decomposed using similar sparsity patterns. Quantitative and qualitative experiments with images corrupted with synthetic or real noise have shown that the proposed algorithm outperforms the state of the art in image demosaicking and denoising tasks. Next on our agenda is to include non-uniform noise models in the reconstruction process, then adapt our approach to other challenging image manipulation problems in computational photography, including deblurring, inpainting, and texture synthesis in still images and video sequences.

[3] Note that NoiseWare does not process directly the RAW files, but requires at first to use a demosaicking software. We have chosen to combine NoiseWare and Adobe Camera Raw in our experiments.

(a) Camera jpeg (b) Adobe Camera Raw (c) Noiseware

(d) DxO (e) LSC (f) LSSC

Figure 4.5: Visual Comparison between Camera jpeg output, Adobe Camera Raw, Noise-Ware, DxO Optics Pro, LSC, and proposed LSSC algorithm. (This figure should be viewed in color and by zooming on a computer screen.)



(a) Camera jpeg (b) Adobe Camera Raw (c) Noiseware

(d) DxO (e) LSC (f) LSSC

Figure 4.6: Visual Comparison between Camera jpeg output, Adobe Camera Raw, NoiseWare, DxO Optics Pro, LSC, and proposed LSSC algorithm. (This figure should be viewed in color and by zooming on a computer screen.)

(a) Camera jpeg (b) Adobe Camera Raw (c) Noiseware

(d) DxO (e) LSC (f) LSSC

Figure 4.7: Visual Comparison between Camera jpeg output, Adobe Camera Raw, NoiseWare, DxO Optics Pro, LSC, and proposed LSSC algorithm. (This figure should be viewed in color and by zooming on a computer screen.)

# 5

# Modeling the Local Appearance of Image Patches

**Chapter abstract:** Sparse signal models have been the focus of much recent research, leading to (or improving upon) state-of-the-art results in signal, image, and video restoration. This work extends this line of research into a novel framework for local image discrimination tasks, proposing an energy formulation with both sparse reconstruction and class discrimination components, jointly optimized during dictionary learning. This approach improves over the state of the art in texture segmentation experiments using the Brodatz database, and it paves the way for a novel scene analysis and recognition framework based on simultaneously learning discriminative and reconstructive dictionaries. Preliminary results in this direction using examples from the Pascal VOC'06 and Graz02 datasets are presented as well. We also apply our method to edge detection, category-based edge selection and image classification tasks. Experiments on the Berkeley edge detection benchmark and the PASCAL VOC'05 and VOC'07 datasets demonstrate the computational efficiency of our algorithm and its ability to learn local image descriptions that effectively support demanding computer vision tasks.

The material of the chapter is based on the following publications:

J. Mairal, F. Bach, J. Ponce, G. Sapiro and A. Zisserman. Discriminative Learned Dictionaries for Local Image Analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2008

J. Mairal, M. Leordeanu, F. Bach, M. Hebert and J. Ponce. Discriminative Sparse Image Models for Class-Specific Edge Detection and Image Interpretation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2008

These papers are prior to the task-driven dictionary learning approach which will be presented in the next chapter. They introduce the concept of learning dictionaries in a supervised way, and use different optimization methods, which are more heuristic than the more recent approach of Chapter 6.

## 5.1 Introduction

Sparse representations have recently drawn much interest in signal, image, and video processing. Under the assumption that natural images admit a sparse decomposition in

some redundant basis (or so-called *dictionary*), several such models have been proposed, e.g., curvelets (Candes and Donoho, 2004), wedgelets (Donoho, 1998) bandlets (Mallat and Pennec, 2005b,a) and various sorts of wavelets (Mallat, 1999). Recent works have shown that learning non-parametric dictionaries for image representation instead of using off-the-shelf ones, can improve image restoration (Elad and Aharon, 2006; Ranzato et al., 2007b; Roth and Black, 2005; Weiss and Freeman, 2007). We consider in this chapter the dictionary learning framework as a tool for modelling the local appearance of image patches.

The computer vision community has indeed been interested in extracting sparse information from images for recognition, mainly by designing different types of image descriptors (e.g., SIFT, Lowe, 2004, HOG, Dalal and Triggs, 2005), although the sparsity concept here is different from the sparse decompositions of images we have presented so far in this manuscript. These representations have been shown to be discriminative enough to be used in conjunction with classifiers (e.g., SVMs for SIFT descriptors in "bags of features" models Wallraven et al., 2003; Lazebnik et al., 2006). Introducing learning into the feature extraction task has been part of the motivation for some recent works: e.g., Ranzato et al. (2007a) learns image features using convolutional neural networks; while Lazebnik and Raginsky (2007); Winn et al. (2005) present discriminative strategies for learning visual codebooks. Sparse decompositions have also been used for face recognition (Wright et al., 2009a), signal classification (Grosse et al., 2007; Huang et al., 2009) and texture classification (Lee and Lewicki, 2002; Peyré, 2009; Skretting and Husoy, 2006). Interestingly, while discrimination is the main goal of these papers, the optimization (dictionary design) is purely generative, based on a criteria which does not explicitly include the actual discrimination task, which is one of the key contributions of our work.

The framework introduced in this work addresses the learning of multiple dictionaries which are simultaneously reconstructive and discriminative, and the use of the reconstruction errors of these dictionaries on image patches to derive a pixelwise classification. The novelty of the proposed approach is twofold: First, redundant non-parametric dictionaries are learned, in contrast with the more common use of predefined features and dictionaries (Huang and Aviyente, 2006; Wright et al., 2009a). Second, the sparse local representations are learned with an explicit discriminative goal, making the proposed model different from traditional reconstructive ones (Grosse et al., 2007; Peyré, 2009; Skretting and Husoy, 2006). We illustrate the benefits of this approach in a texture segmentation task on the Brodatz dataset (Randen and Husoy, 1999) for which it improves over the state of the art (Lillo et al., 2007; Mäenpää et al., 2000; Skretting and Husoy, 2006), and also present preliminary results showing that it can be used to learn discriminative key patches from the Pascal VOC'06 (Everingham et al., 2007) database, and perform the weakly supervised form of feature selection advocated by Pantofaru et al. (2006); Tuytelaars and Schmid (2007) on images from the Graz02 dataset (Opelt and Pinz, 2005). We also use this algorithm in a multiscale extension of our discriminative framework and apply it to the problem of edge detection, with raw results very close to the state of the art on the Berkeley segmentation dataset (Martin et al., 2001).

Following Prasad et al. (2006), we also learn a class-specific edge detector and show that using it as a preprocessing stage to a state-of-the-art edge-based classifier of Leordeanu et al. (2007) can significantly improve the performance of the latter.

### 5.1.1 Contributions

This chapter makes three contributions

- It introduces discriminative dictionaries for modelling the local appearance of objects and textures, leading to state-of-the-art results for these tasks.
- It demonstrates that this approach is particularly well suited for modelling edges in natural images and for learning a category-based edge detector.
- It shows that edge-based classifiers such as (Leordeanu et al., 2007) can be significantly improved when coupled with a category-based edge detector.

## 5.2 Learning Discriminative Dictionaries

Assume that we have $N$ sets $S_i$ of training patches, $i = 1 \dots N$, belonging to $N$ different classes. The simplest strategy for using dictionaries for discrimination consists of first learning $N$ dictionaries $\mathbf{D}^i, i = 1 \dots N$, one for each class. Approximating each patch using a constant sparsity $L$ and the $N$ different dictionaries provides $N$ different residual errors, which can then be used as classification features. This is essentially the strategy employed by Peyré (2009); Skretting and Husoy (2006). Thus, the first naive way of estimating the class $i_0$ for some patch $\mathbf{x}$ is to write (as done by Wright et al. (2009a), but with learned dictionaries):

$$\hat{i}_0 = \arg\min_{i=1\dots N} \mathcal{R}^\star(\mathbf{x}, \mathbf{D}^i), \tag{5.1}$$

where

$$\mathcal{R}^\star(\mathbf{x}, \mathbf{D}^i) \triangleq \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \|\mathbf{x} - \mathbf{D}^i \boldsymbol{\alpha}\|_2^2 \;\text{ s.t. }\; \phi(\boldsymbol{\alpha}) \leq \lambda,$$

where $\phi$ denotes either the $\ell_0$ quasi-norm or the $\ell_1$ norm, and $\lambda$ is a regularization parameter.

Instead of this *reconstruction-based* approach, we show that better results can be achieved, in general, by learning *discriminative* sparse representations, while keeping the same robustness against noise or occlusions (see also Huang and Aviyente, 2006), which discriminative methods may be sensitive to (Wright et al., 2009a).[1]

[1] Note also that due to the over-completeness of the dictionaries, and possible correlations between the classes, sparse representations of members of one class with dictionaries from a different class can still be very efficient and produce small values for the residual error $\mathcal{R}^\star$.

### 5.2.1 Learning discriminative dictionaries

We present in this section a formulation for learning discriminative dictionaries, using an energy formulation that contains both sparse reconstruction and class discrimination components, jointly optimized towards the learning of the dictionaries. Given $N$ classes $S_i$ of signals, $i = 1, \ldots, N$, the goal is to learn $N$ discriminative dictionaries $\mathbf{D}^i$, each of them being adapted to reconstructing a specific class better than others. This yields the following optimization problem:

$$\min_{\{\mathbf{D}^j\}_{j=1}^N} \sum_{i=1}^{N} \sum_{l \in S_i} \mathcal{C}_i^\gamma(\{\mathcal{R}^\star(\mathbf{x}^l, \mathbf{D}^j)\}_{j=1}^N) + \gamma\nu\mathcal{R}^\star(\mathbf{x}^l, \mathbf{D}^i).$$

Here, $\mathcal{R}^\star(\mathbf{x}^l, \mathbf{D}^i)$ is the reconstruction error of the signal $\mathbf{x}^l$ using the dictionary $\mathbf{D}^i$ and $\mathcal{C}_i^\gamma$ is a softmax discriminative cost function, which is the multiclass version of the logistic regression function, defined as

$$\mathcal{C}_i^\gamma(y_1, y_2, ..., y_N) \triangleq \log\Big(\sum_{j=1}^{N} e^{-\gamma(y_j - y_i)}\Big),$$

which is close to zero when $y_i$ is the smallest value among the $y_j$ $j = 1, \ldots, N$, and provides an asymptotic linear penalty cost $\gamma(y_i - \min_j y_j)$ otherwise. These are the multiclass versions of the logistic function presented in Figure 5.1, which is differentiable and enjoys properties similar to the hinge loss function from the support vector machine (SVM) literature. Increasing the value of the new parameter $\gamma > 0$ provides a higher relative penalty cost for each misclassified patch, at the same time making the cost function less smooth. Its purpose is to make the dictionary $\mathbf{D}^i$ better at reconstructing

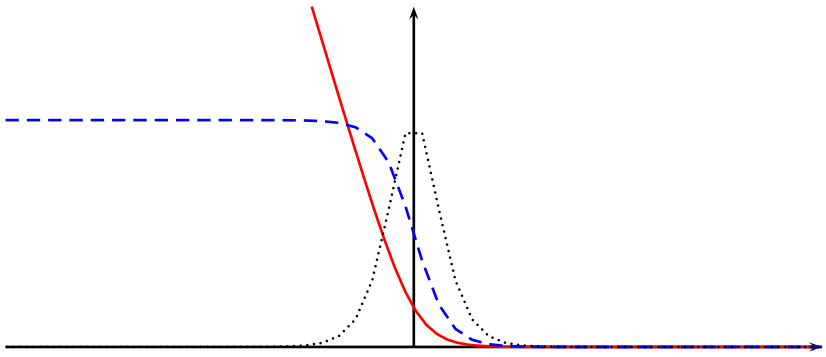


Figure 5.1: The logistic function (red, continuous), its first derivative (blue, dashed), and its second derivative (black, dotted).

the signals from class $S_i$ than the dictionaries $\mathbf{D}^j$ for $j$ different than $i$. In this equation, $\gamma$ is a parameter of the cost function, and $\nu$ controls the trade-off between reconstruction and discrimination. More details on this formulation, on how to optimize it, and on the choices of the parameters $\gamma$ and $\nu$ are given in (Mairal et al., 2008a).

### 5.2.2 A new multiscale feature space

In this subsection, we present a multiscale extension and some improvements to the classification procedure outlined in (Mairal et al., 2008a), which have proven to improve noticeably the performance of our classifier. Although it is presented for illustrative purposes when the signals are image patches, its scope goes beyond vision tasks and similar concepts could be applied in other situations.

An important assumption, commonly and successfully used in image processing, is the existence of multiscale features in images, which we exploit using a multi-layer approach, presented in Figure 5.2. It allows us to work with images at different resolutions, with different sizes of patches and avoids the choice of the hyperparameters $\lambda$ during the testing phase.

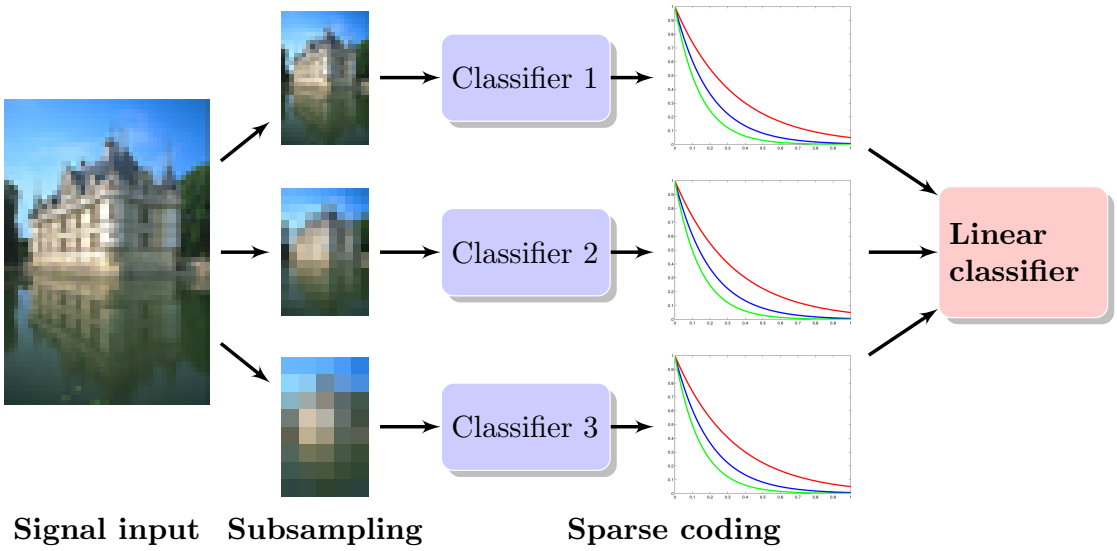



Figure 5.2: Multiscale classifier using discriminative sparse coding. The signal input is subsampled in different signal sizes. Then, each classifier outputs $N$ curves of reconstruction errors as functions of a sparsity constraint, one curve per dictionary. A linear classifier provides a confidence value.

In (Mairal et al., 2008a), the class $i_0$ for some patch $\mathbf{x}$ is taken according to Eq. (5.1). However, $\mathcal{R}^\star$ is a reconstruction error obtained with an arbitrary $\ell_0$ or $\ell_1$ constraint, which does not take into account the different characteristics of the patches. Patches with a high amount of information need indeed a lot of atoms to achieve a correct representation, and should therefore benefit from being classified with a high sparsity factor. On the other hand, some patches admit extremely sparse representations, and should be classified with a small sparsity factor. To cope with this effect, we have chosen when testing a given patch to compute many reconstruction errors with different constraints (different values of $\lambda$). Thanks to the nature of the OMP and LARS-Lasso, this can be done without additional computations since both these algorithms can plot

the reconstruction error as a function of the given constraint value in one pass. The curves produced by each different dictionaries (one dictionary per class) on a patch can then be incorporated into a logistic regression classifier or a linear SVM (Shawe-Taylor and Cristianini, 2004) as feature vectors.

The same idea can be used to combine the output of different classifiers, working at different resolutions and with different sizes of patches. Suppose you train $P$ discriminative classifiers with different sizes of patches and different resolutions. Testing a signal $\mathbf{x}$ consists of sending $\mathbf{x}$ to each classifier independently, cropping and subsampling it so that its size and resolution match the classifier. Each classifier produces $N$ curves representing the reconstruction errors using the $N$ dictionaries and different sparsity constraints. Then, a linear classifier (logistic regression or SVM) permits to combine these outputs into a confidence value. This scheme is presented in Figure 5.2.

## 5.3 Modeling Texture and Local Appearance of Objects

We apply in this section our discriminative dictionary framework for modeling textures and build a local representation of objects.

### 5.3.1 Data Issues

Before presenting experiments, we discuss here some data related issues. Depending on the particular application of the proposed framework, different prefiltering operations can be applied to the input data. We mention first the possibility to apply a Gaussian mask to the input patches (by multiplying element-wise the patches by the mask), in order to give more weight to the center of the patches, since the framework is designed for local discrimination. We mention also the possibility to pre-process the data using a Laplacian filter, which has proven to give more discriminatory power. Since a Laplacian filter can be represented by a difference of Gaussians, this step is consistent with previous works on local descriptors. Both proposed pre-filtering can be simultaneously used depending on the chosen application.

The presented framework is flexible in the sense that it is very easy to take into account different types of vectorial information. For instance, using color patches for the K-SVD has been addressed by Mairal et al. (2008b) by concatenating R,G,B information from a patch into single vectors. This can be directly applied here. One could also opt to only include the mean color of a patch, if we consider that the geometrical structure is more meaningful for discrimination. It is therefore possible to work with vectors representing grayscale patches and just 3 average R,G,B values. This permits to take into account the color information without multiplying by 3 the dimensionality of the patches. Depending on the data, other types of information could be added this way.

### 5.3.2 Texture Segmentation of the Brodatz Dataset

Texture segmentation and classification is a natural application of our framework, since it can be formulated as a local feature extraction and patch classification process. We

| ♯ | C1 | C2 | C3 | C4 | R1 | R2 | D1 | D2 |
|---|---|---|---|---|---|---|---|---|
| 1 | 7.2 | 6.7 | 5.5 | 3.37 | 2.22 | 1.69 | 1.89 | **1.61** |
| 2 | 18.9 | 14.3 | **7.3** | 16.05 | 24.66 | 36.5 | 16.38 | 16.42 |
| 3 | 20.6 | 10.2 | 13.2 | 13.03 | 10.20 | 5.49 | 9.11 | **4.15** |
| 4 | 16.8 | 9.1 | 5.6 | 6.62 | 6.66 | 4.60 | 3.79 | **3.67** |
| 5 | 17.2 | 8.0 | 10.5 | 8.15 | 5.26 | **4.32** | 5.10 | 4.58 |
| 6 | 34.7 | 15.3 | 17.1 | 18.66 | 16.88 | 15.50 | 12.91 | **9.04** |
| 7 | 41.7 | 20.7 | 17.2 | 21.67 | 19.32 | 21.89 | 11.44 | **8.80** |
| 8 | 32.3 | 18.1 | 18.9 | 21.96 | 13.27 | 11.80 | 14.77 | **2.24** |
| 9 | 27.8 | 21.4 | 21.4 | 9.61 | 18.85 | 21.88 | 10.12 | **2.04** |
| 10 | 0.7 | 0.4 | NA | 0.36 | 0.35 | **0.17** | 0.20 | **0.17** |
| 11 | **0.2** | 0.8 | NA | 1.33 | 0.58 | 0.73 | 0.41 | 0.60 |
| 12 | 2.5 | 5.3 | NA | 1.14 | 1.36 | **0.37** | 1.97 | 0.78 |
| **Av.** | 18.4 | 10.9 | NA | 10.16 | 9.97 | 10.41 | 7.34 | **4.50** |

Table 5.1: Error rates for the segmentation/classification task for the Brodatz dataset. The proposed framework is compared with a number of reported state-of-the-art results (Lillo et al., 2007; Mäenpää et al., 2000; Skretting and Husoy, 2006) and the best results reported by Randen and Husoy (1999). Randen and Husoy (1999) is denoted by C1, Mäenpää et al. (2000) by C2, Skretting and Husoy (2006) by C3 and Lillo et al. (2007) by C4. R1 and R2 denote the reconstructive approach, while D1 and D2 stand for the discriminative one. A Gaussian regularization has been used for R1 and D1, a graph-cut-based one for R2 and D2. The best results for each image are in bold.

have chosen to evaluate our method on the Brodatz dataset, introduced by Randen and Husoy (1999), which provides a set of "patchwork" images composed of textures from different classes, and a training sample for each class. The suite of the 12 images is presented by Mäenpää et al. (2000).

In our experiments, following the same methodology as Skretting and Husoy (2006), each of the patches of the training images were used as a training set. We use patches of size $m = 144$ ($12 \times 12$), dictionaries of size $p = 128$ and a sparsity factor $\lambda = 4$ ($\ell_0$ constraint). A Gaussian mask of standard deviation 4 (element-wise multiplication) and a Laplacian filter were applied on each patch as a prefiltering. Then 30 iterations of our discriminative framework are performed. After this training stage, each one of the $12 \times 12$ patches from the test images are classified using the learned dictionaries (by comparing the corresponding representation error $\mathcal{R}^\star$ for each dictionary). Smoothing follows to obtain the segmentation. We present two alternatives, either using a simple Gaussian filtering with a standard deviation of 12, or applying a graph-cut alpha-expansion algorithm, (Boykov et al., 2001; Kolmogorov and Zabih, 2004), based on a classical Potts model with an 8-neighborhood system. The cost associated with a patch $\mathbf{x}$ and a class $S_i$ is 0 if $\mathbf{x}$ has been classified as part of $S_i$, and 1 otherwise. A constant regularization cost between two adjacent patches of 1.75 has proven to be appropriate. Table 5.1 reports the results.

From these experiments, we observe that our method significantly outperforms those reported by Lillo et al. (2007); Mäenpää et al. (2000); Randen and Husoy (1999); Skretting and Husoy (2006), regardless of the selected regularization method, and performs best for all but two of the images. Moreover, while the purely *reconstructive* framework already provides good results, we observe that the *discriminative* one noticeably improves the classification rate except for examples 5 and 12. In these two particular cases, it has proven to be an artefact from the image smoothing. The classification rate before and after smoothing are indeed not fully correlated. Smoothing will better remove isolated misclassified patches and sometimes the misclassified patches from the reconstructive approach are more isolated than with the discriminative one. Note that our model under-performs at image 2, where the size of the patches we had chosen has proven to be particularly not adapted, which is a motivation for developing a multiscale framework. Some qualitative results are presented in Figure 5.3.

### 5.3.3 Learning Discriminant Images Patches of Objects

To assess the promise of our local appearance model, we verify its ability to learn discriminative patches for object categories from a very general database with a high variability. To that effect, we have chosen some classes from the Pascal VOC'06 database (Everingham et al., 2007) and conducted the following qualitative experiment: Given one object class $A$ (e.g., bicycle, sheep, car, cow), we build two sets of patches $S_1$ and $S_2$. The first one is composed of 200 000 $12 \times 12$ patches, extracted from bounding boxes that contain one object of class $A$. The second one was composed of 200 000 $12 \times 12$ background patches from an image that contains one object of class $A$. This way, classification at the patch level is difficult since the overlap between $S_1$ and $S_2$ is important: (i) many small patches from an object look like some patches from the background and vice-versa; (ii) some patches from an object's bounding boxes do not necessarily fully overlap with the object.

Analyzing globally the patches as a whole, or using a multiscale framework like in (Agarwal and Triggs, 2006; Tuytelaars and Schmid, 2007), to capture global appearance of objects, are among the possibilities that could make this problem more tractable, but these are beyond the scope of this paper. Instead, we want to show here that our purely local model can deal with this overlap between classes and learn the local parts of objects from class $A$ that are discriminative, when observed at the patch level. The experiment we did consists of learning two discriminative dictionaries, $\mathbf{D}_1$ for $S_1$ and $\mathbf{D}_2$ for $S_2$, with $p = 128$, $\lambda = 4$ and 15 iterations of our algorithm, and then to pursue the discriminative learning during 15 additional iterations, but at each new iteration, pruning the set $S_1$ by keeping the 90% "best classified patches." This way, one hopes to remove the overlap between $S_1$ and $S_2$ and to enforce the learning on the key-patches of the objects. Examples with various classes of the Pascal dataset are presented in Figures 5.4 and 5.5. All the images we used in our test procedure are from the official validation set and are not used during the training. The test images are rescaled so that the maximum between the height and the width of the image is less than 256 pixels. The same prefiltering as for the texture segmentation is applied and the average color of each

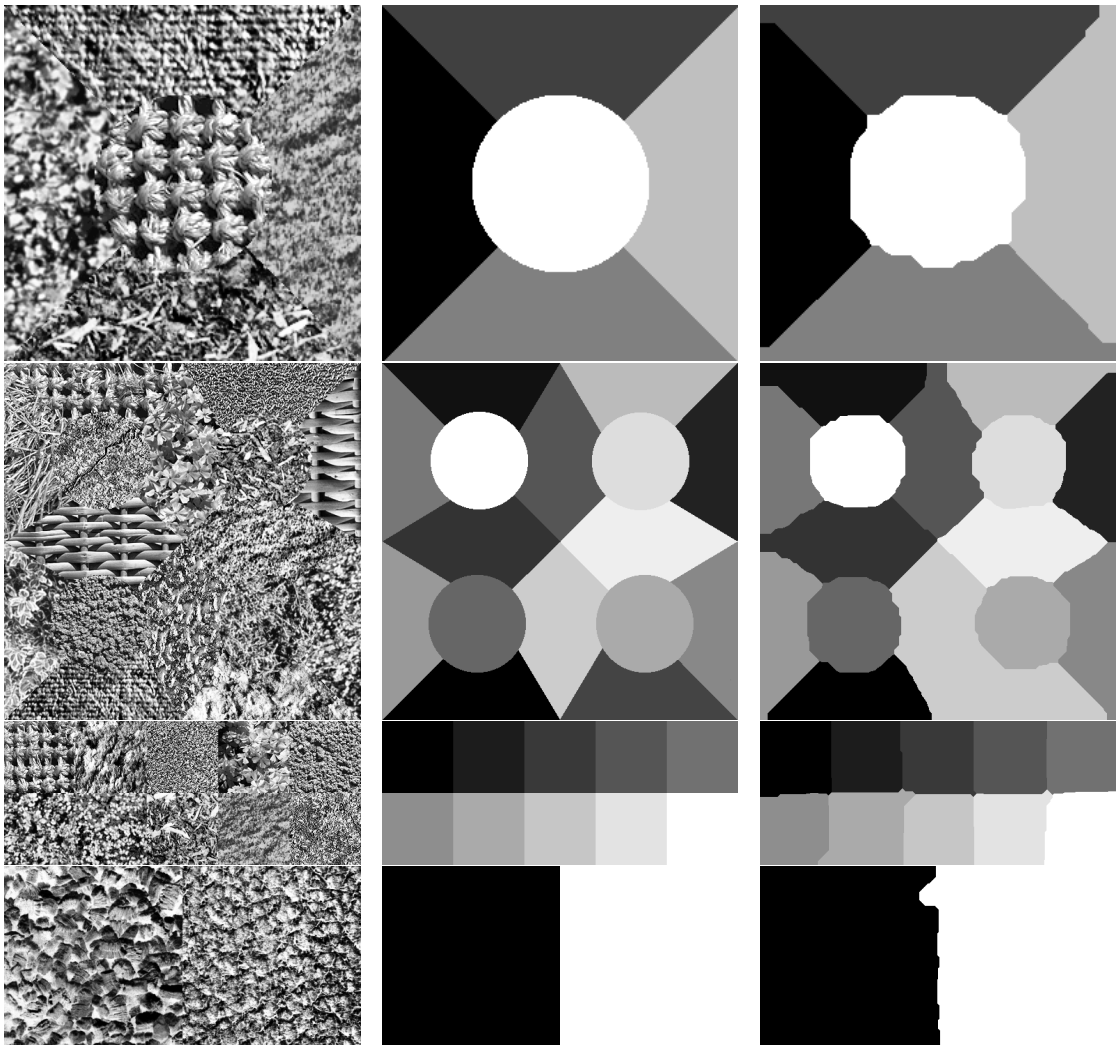


Figure 5.3: Subset of the Brodatz dataset with various number of classes: From top to bottom, images 4, 7, 9 and 12. The ground-truth segmentation is displayed in the middle and the resulting segmentation on the right side with a graph-cut regularization. Note that the segmentation is in general very precise but fails at separating two classes on image 7.

patch is taken into account. The learned key-patches focus on parts of the object that stand as locally discriminative compared to the background. These eventually could be used as inputs to other algorithms of the “bags-of-words" type.

Figure 5.6 shows examples of the learned dictionaries obtained with the discriminative and the reconstructive approaches.

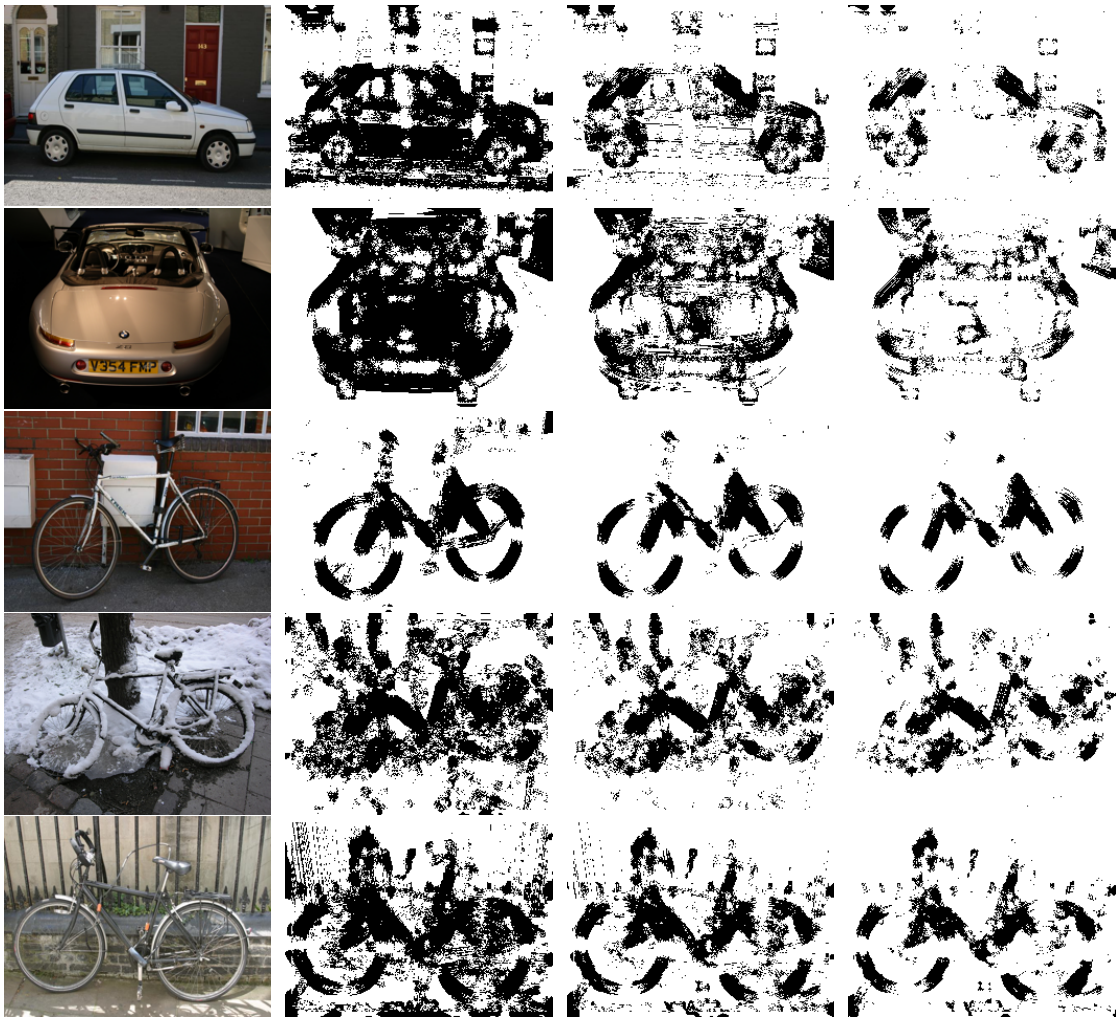


Figure 5.4: Learning of key-patches from the Pascal VOC'06 dataset. Column 1 presents the test image. Columns 2,3,4 present the raw pixelwise classification results obtained respectively at iterations 20,25 and 30 of the procedure, during the pruning of the dataset. Interestingly, the vertical and horizontal edges of the bicycles are not considered as locally discriminative in an urban environment.

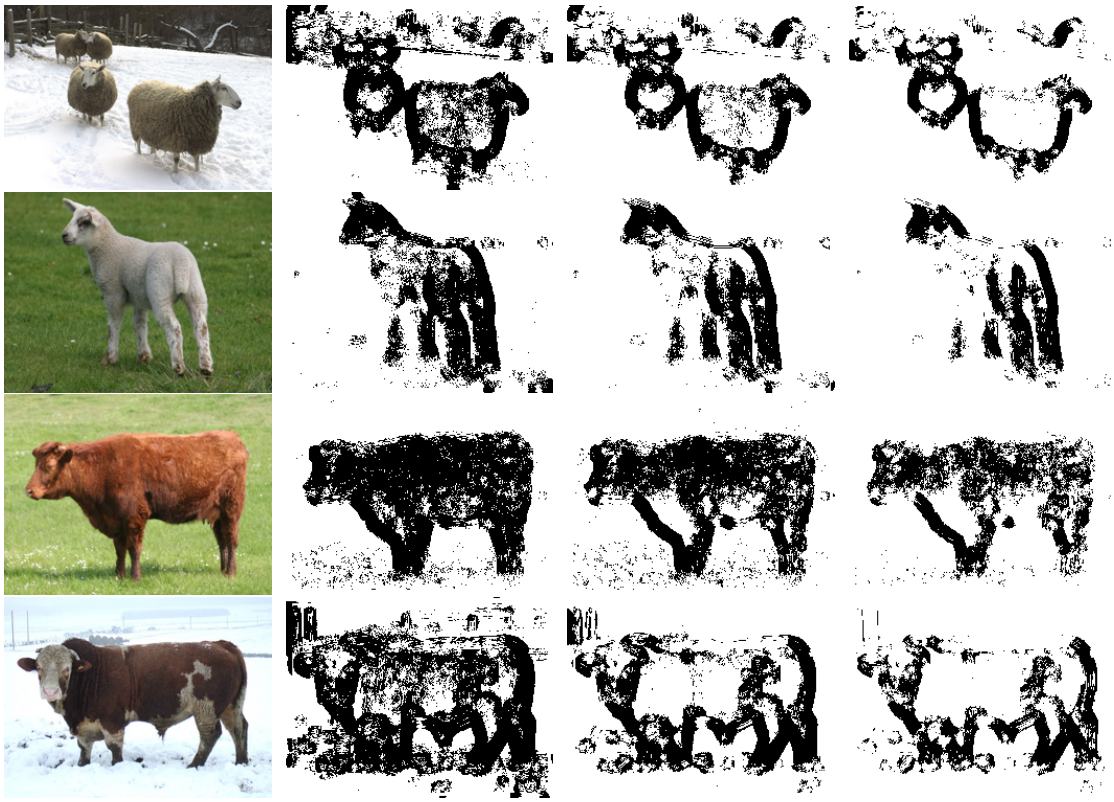
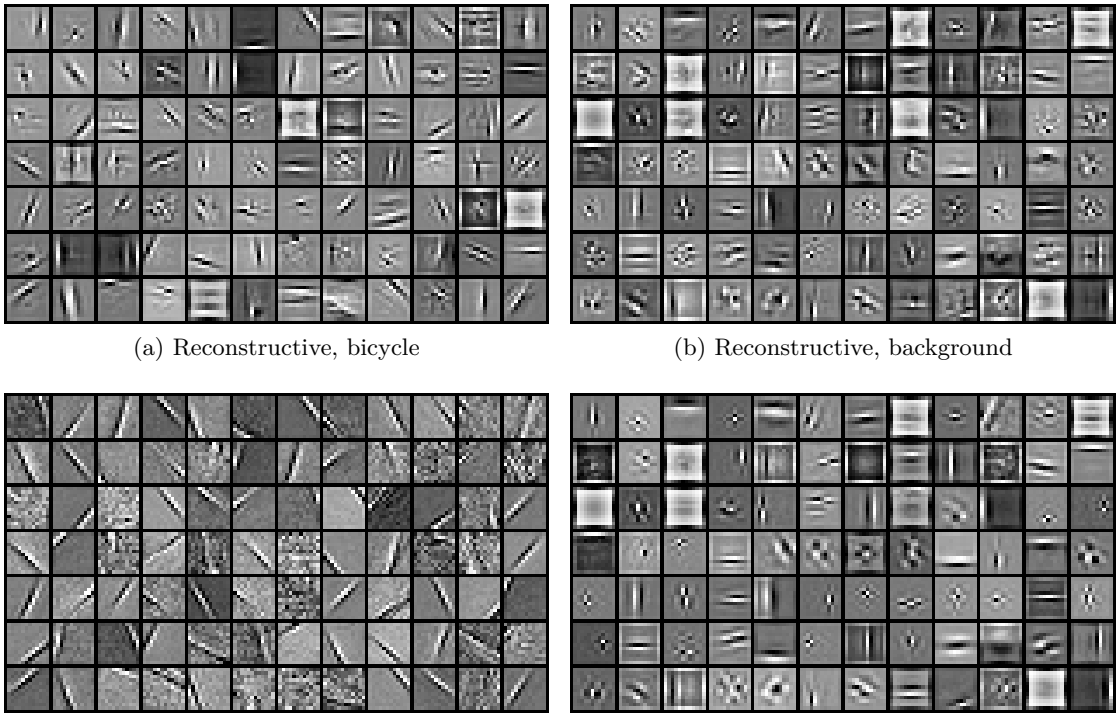


Figure 5.5: Learning of key-patches from the Pascal VOC'06 dataset. Column 1 presents the test image. Columns 2,3,4 present the raw pixelwise classification results obtained respectively at iterations 20,25 and 30 of the procedure, during the pruning of the dataset. Interestingly, the vertical and horizontal edges of the bicycles are not considered as locally discriminative in an urban environment.

### 5.3.4 Weakly-Supervised Feature Selection

Using the same methodology as Pantofaru et al. (2006) and Tuytelaars and Schmid (2007), we evaluate quantitatively our pixelwise classification on the "bike" category from the Graz02 dataset (Opelt and Pinz, 2005), in a weakly supervised fashion (*without using any ground truth information or bounding box during the training*), with the same parameters, pre-processing and algorithm as in the previous subsection (which prunes iteratively the training set after iteration 15), except that we use a patch size of $m = 15 \times 15$ and process the images at half resolution to capture more context around each pixel. We use the first 300 images of the classes "bike" and "background" and use odd images for training, keeping the even images for testing. To produce a confidence value per pixel we have chosen to measure the reconstruction errors of the tested patches with different sparsity factors $\lambda = 1, \ldots, 15$ and use these values as feature vectors in a logistic linear classifier. A Gaussian regularization similar to that used in our texture



(a) Reconstructive, bicycle (b) Reconstructive, background

(c) Discriminative, bicycle (d) Discriminative, background

Figure 5.6: Parts of the dictionaries, learned on the class 'bicycle' from the Pascal VOC06 dataset. The left part has been learned on bounding boxes containing a bicycle, the right part on background regions. The resulting dictionaries from the two approaches, reconstructive and discriminative, are presented. Visually, the dictionaries produced by the discriminative approach are less similar to each other than with the reconstructive one.

segmentation experiments is applied and has proven to improve noticeably the classification performance. Corresponding precision-recall curves are presented in Figure 5.7 and compared with (Pantofaru et al., 2006; Tuytelaars and Schmid, 2007). As one can see, our algorithm produces the best results. Nevertheless, a more exhaustive study with different classes and datasets with a more precise ground truth would be needed to draw general conclusions about the relative performance of these three methods.

## 5.4 Combining Geometry and Local Appearance of Edges

We show in this section how to use our dictionary learning framework to model the local appearance of edges. We first evaluate the performance of the framework for an edge detection task, and then move to category-based edge detection and object recognition.

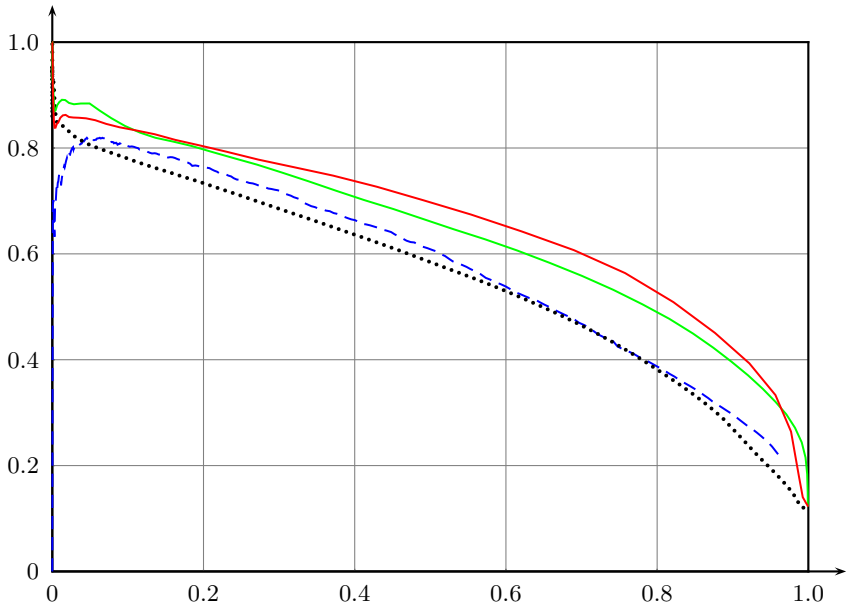



Figure 5.7: Precision-recall curve obtained by our framework for the bikes, without pruning of the training dataset (green, continuous), and after 5 pruning iterations (red, continuous), compared with the one from Pantofaru et al. (2006) (blue, dashed) and Tuytelaars and Schmid (2007) (black, dotted).

### 5.4.1 Edge Detection

We have chosen to train and evaluate our multiscale discriminative framework on the Berkeley segmentation dataset (Martin et al., 2001). To accelerate the procedure and obtain thin edges, we first process all the images with a Canny edge detector (Canny, 1986) without thresholding. Then, we use the manually segmented images from the training set to classify the pixels from these Canny edges into two classes : $S_1$ for the ones that are close to a human edges, and $S_2$ for the others (bad Canny edges). RGB patches are concatenated into vectors. The size $p$ of all of our dictionaries are 256. 14 local classifiers using 7 different sizes of patches' edges $e = 5, 7, 9, 11, 15, 19, 23$, and 2 resolutions (full and half) are trained independently and $J = 25$ iterations with a sparsity constraint of $L = 6$, on a sample of 150 000 random patches from $S_1$ and 150 000 patches from $S_2$. This maximum size of patches associated with the half-resolution version of the images allows us to capture sufficient neighborhood context around each pixel, which has proven to be crucial (Dollar et al., 2006). Then, a new sample of the training set is encoded using each trained dictionary and the curves of the reconstruction error as function of the sparsity constraint ($\lambda = 1, 2, \ldots, 15$ for the $\ell_0$ case), ($\lambda = 0.1, 0.2, \ldots, 2.0$ for the $\ell_1$ case) are computed. All the curves corresponding to a given patch are concatenated into a single feature vector. A linear logistic classifier is trained. During the test phase, we have chosen to compute independently a confidence value per

pixel on the Canny edges without doing any post-processing or spatial regularization, which by itself is also a difficult problem. We have therefore chosen to evaluate our raw and noisy results but we believe these can be further improved by applying post-processing methods like Ren et al. (2005). Precision-recall curves obtained using the Berkeley segmentation benchmark are presented in Figure 5.8 and are compared with Pb (Martin et al., 2004), BEL (Dollar et al., 2006) and UCM (Arbelaez, 2006). Note that without any post-processing, our generic method achieves similar performance as (Dollar et al., 2006) just behind (Arbelaez, 2006) in terms of F-measure (see Martin et al., 2001, 2004, for its definition), although it was not specifically designed for this task. Note that our method performs the best for high recalls, but is slightly behind for lower recalls, where our edges map contain many small nonmeaningful edges (noise).

Interestingly, our formulation with the $\ell_0$ constraint (F=0.66) gives slightly better results than when using the $\ell_1$ one (F = 0.64) in our multiscale framework. Nevertheless, in the single scale case, we have observed an opposite phenomenon.

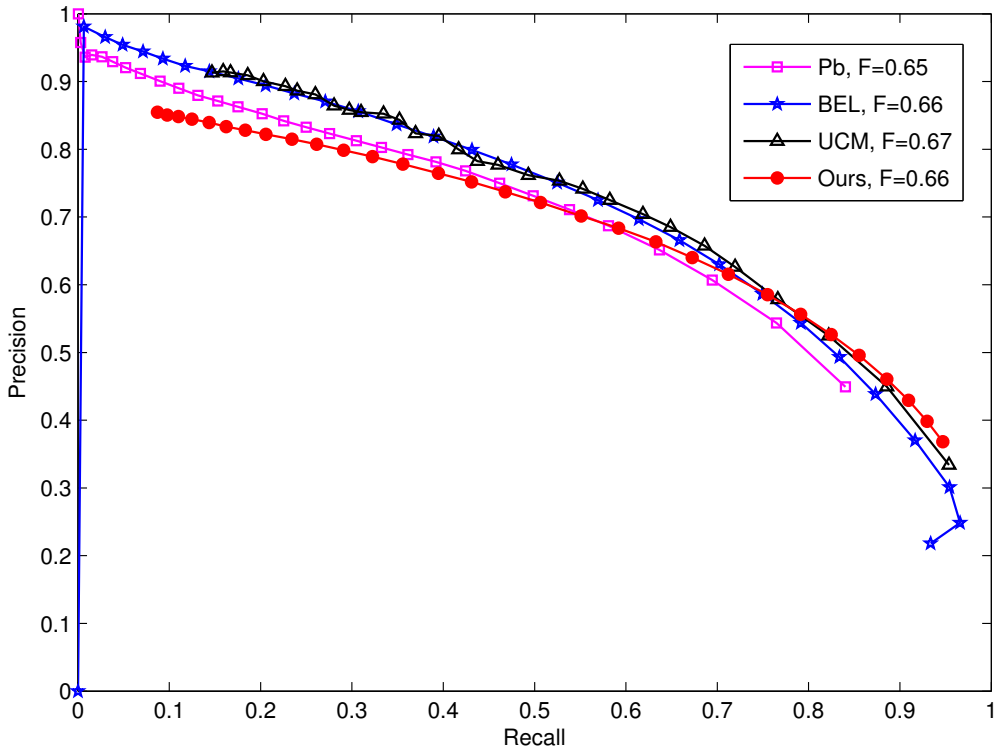



Figure 5.8: Precision-recall curve for our edge detection task.

### 5.4.2 Combining Shape with Local Appearance for Object Recognition

We now show how to use our edge detector for object recognition by combining it with the shape-based method for category recognition from Leordeanu et al. (2007). Their algorithm learns the shape of a specific category in a discriminative fashion by selecting from training images the pieces of contours that are most relevant for a specific category. The method exploits the pairwise geometric relationships between simple features that include relative angle and distance information. Thus, features are selected based on how discriminative they are together, as a group, and not on an individual basis (for more details on learning these models, see Leordeanu et al., 2007). After the models are learned, they are matched against contours extracted from novel images by formulating the task as a graph matching problem, where the focus is on pairwise relationships between features and not their local appearance. While Leordeanu et al. (2007) make the point that shape is stronger than local appearance for category recognition we want to demonstrate that there is a natural way of combining shape and local appearance that produces a significant increase in the recognition performance.

While shape is indeed important for category recognition, the toughest challenge for such shape-based algorithms on difficult databases is the change in view point, which makes the use of $2D$ shape less powerful. Therefore, it is important to be able to help the shape recognizer by local appearance methods that are geometry independent and thus less sensitive to such changes in viewpoint. Our proposed approach of combining local appearance with shape is to first learn a class specific edge detector on pieces of contours. Next this class specific edge detector is used to filter out the irrelevant edges while learning the shape-based classifier based on (Leordeanu et al., 2007). Similarly, at testing time, the shape-based algorithm is applied to the contours that survive after filtering them with our class dependent edge detector. The outputs of both the shape-based classifier and the real values given by our detector are later combined for the final recognition score. This framework provides a natural way of combining the lower-level, appearance based edge detection and contour filtering with the more higher level, shape-based approach. The algorithm can be described in more detail as follows:

- **Contour Training**: learn a category based edge classifier using our proposed method. For each image, we apply our general edge detector, then obtain pieces of contours obtained as Leordeanu et al. (2007). Next, we train class specific detectors on such contours belonging to the positive class vs. all other classes.

- **Shape Training**: the output of the class specific edge detector on each training image (contours with average scores less than 0.5 are removed) is given to the shape-based algorithm from Leordeanu et al. (2007). Thus the shape classifier is trained on images that were first preprocessed with our class dependent contour classification.

- **Testing**: each testing image is first preprocessed at the individual contours level in the same way as it is done at training time. The edge classifier it is used to

filter out contours that had an average score less than 0.5 (over all pixels belonging to that contour). The contours that survived are then used by the shape-based classifier, to obtain the final recognition score.

We now use our edge detector on all the images from Pascal VOC'05 and VOC'07 (Everingham et al., 2007) and postprocess them to remove nonmeaningful edges using the same grouping method as Leordeanu et al. (2007). Then, we train our class-specific edge detector on the training set of each dataset, using the same training set as Leordeanu et al. (2007) for VOC'05 and the training set of VOC'07. For each class (4 in VOC'05 and 20 in VOC'07) a one-vs-all classifier is trained using the exact same parameters as for the edge detection, which allows us to give a confidence value for each edge as being part of a specific object type. Some examples of filtered edges maps are presented in Figure 5.9.

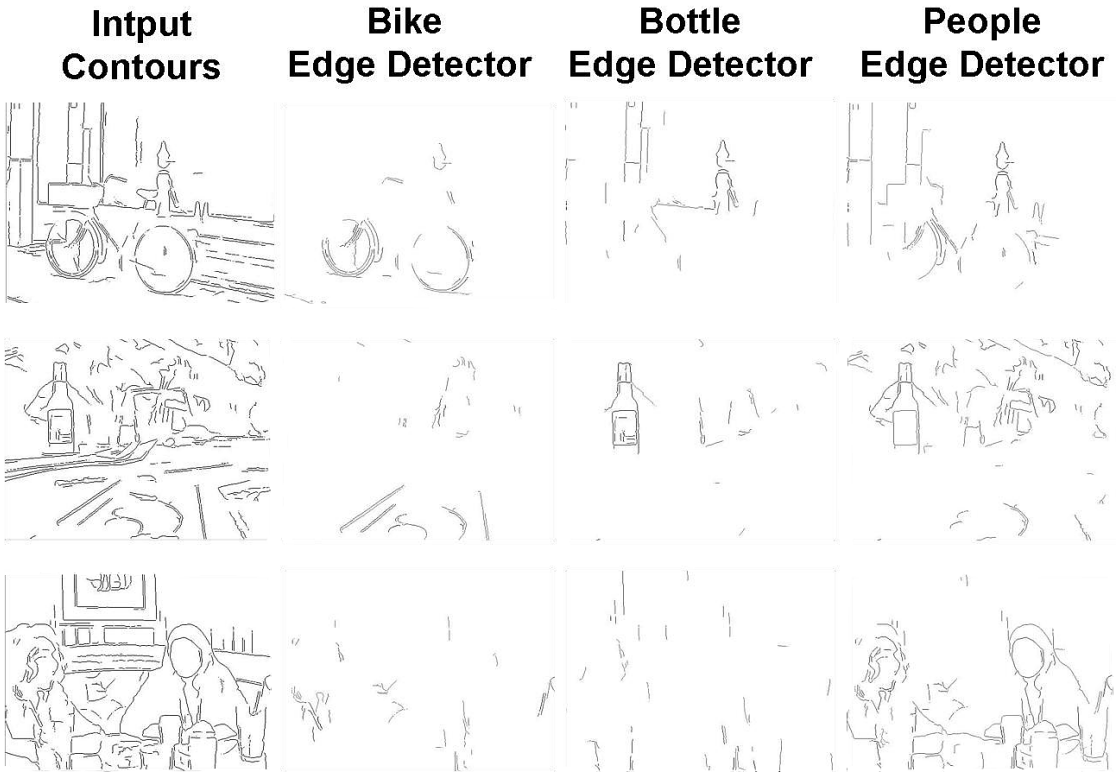



Figure 5.9: Examples of filtered edges. Each column shows the edges that survive (score $\geq 0.5$) after applying different class specific edge detectors.

In our first set of recognition experiments we want to quantify the benefit of using our class-specific edge detector for object category recognition (shown in Table 5.2). Thus, we perform the same sets of experiments as Leordeanu et al. (2007) and Winn et al. (2005), on the same training and testing image sets from Pascal 2005 dataset, on

a multiclass classification task. Following the works we compare against, we also use bounding boxes (for more details on the the experiments set up see Leordeanu et al., 2007 and Winn et al., 2005). Notice (Tables 5.2, 5.3 and 5.4) that by using our algorithm we are able to reduce the error rate more than 3-fold as compared with the shape alone classifier (3.2% vs. 10.6%) and more then 7-fold when compared to the appearance based method of Winn et al. (2005). We believe that these results are very encouraging and that they demonstrate the promise of our edge classifier for higher level tasks such as object recognition.

Table 5.2: Average multiclass recognition rates on the Pascal 2005 Dataset

| Algorithm | Ours + Leordeanu | Leordeanu | Winn |
|---|---|---|---|
| Pascal 05 Dataset | 96.8% | 89.4% | 76.9% |

In the second sets of experiments we apply the same appearance-shape combination algorithm to the two-class classification task of recognizing object categories from the Pascal 2007 dataset (is the category present or not?). We believe that for this very challenging dataset, class-based contour classification can help the shape recognizer greatly, since the objects are undergoing significant changes in viewpoint and scale making their $2D$ shape representation less powerful. We have experimented with 8 object classes (Tables 5.3 and 5.4). For each class we use the training set with bounding boxes (as provided in the Pascal 2007 challenge) for learning both the class specific edge detector and the shape models (in the same fashion as we did for the Pascal 2005 experiment). The test set consisted of the full images (no bounding boxes were used) given as validation set in the Pascal 07 challenge (only the images containing one of the eight classes were kept for both testing and training). Given the difficulty of this dataset we believe that our results are very promising and demonstrate the benefit of combining lower level appearance with higher level, shape based information for object category recognition. More experiments and a full comparison are of course needed.

| Category | Bikes | Cars | Motorbikes | People |
|---|---|---|---|---|
| Bikes | 93.0% | 3.5% | 1.7% | 1.8% |
| Cars | 1.2% | 97.7% | 0.0% | 1.1% |
| Motorbikes | 1.9% | 1.8% | 96.3% | 0% |
| People | 0% | 0% | 0% | 100% |

Table 5.3: Confusion matrix for Pascal 2005 Dataset (using the bounding boxes). Compare this with the confusion matrix obtained when shape alone is used by Leordeanu et al. (2007), on the exact same set of experiments.

| Category | Ours+Leordeanu et al. (2007) | Leordeanu et al. (2007) |
|---|---|---|
| Aeroplane | 71.9% | 61.9% |
| Boat | 67.1% | 56.4% |
| Cat | 82.6% | 53.4% |
| Cow | 68.7% | 59.22% |
| Horse | 76.0% | 67% |
| Motorbike | 80.6% | 73.6% |
| Sheep | 72.9% | 58.4% |
| Tvmonitor | 87.7% | 83.8% |

Table 5.4: Classification results (recognition performance) at equal error rate for 8 classes from the Pascal 07 dataset. Our filtering method reduces the average error rate by 33%.

## 5.5 Conclusion

We have introduced a novel framework for using learned sparse image representations in local classification tasks. Using a local sparsity prior on images, our algorithm learns the local appearance of object categories in a discriminative framework. This is achieved via an efficient optimization of an energy function, leading to the learning of over-complete and non-parametric dictionaries that are explicitly optimized to be both representative and discriminative. We have shown that the proposed approach leads to state-of-the-art segmentation results on the Brodatz dataset, with significant improvements over previously published methods for most examples. Applied to more general image datasets, mainly of natural images, it permits to learn some key-patches of objects and to perform local discrimination/segmentation.

We present a multiscale discriminative framework based on these learned sparse representations, and apply it to the problem of edge detection, and class-specific edge detection, which proves to greatly improve the results obtained with a contour-based classifier (Leordeanu et al., 2007). Our current efforts are devoted to find a way to use our local appearance model as a preprocessing step for a global recognition framework using bags of words, as we have done for a contour-based classifier. Clustering methods of locally selected patches to define interest regions is an option we are considering, which could eventually allow us to get rid of expensive sliding windows analysis for object detection (Dalal and Triggs, 2005). Another direction we are pursuing is to use directly the coefficients $\boldsymbol{\alpha}$ of the sparse decompositions as features. Developing a discriminative optimization framework which can use the robust $\ell_1$ regularization instead of the $\ell_0$ one was a first step, which should make this possible.

# 6

# Task-Driven Dictionary Learning

**Chapter abstract:** Modeling data with linear combinations of a few elements from a *learned* dictionary has been the focus of much recent research in machine learning, neuroscience and signal processing. For signals such as natural images that admit such sparse representations, it is now well established that these models are well suited to *restoration* tasks. In this context, learning the dictionary amounts to solving a large-scale matrix factorization problem, which can be done efficiently with classical optimization tools. The same approach has also been used for learning features from data for other purposes, e.g., image classification, but tuning the dictionary in a supervised way for these tasks has proven to be more difficult. In this chapter, we present a general formulation for supervised dictionary learning adapted to a wide variety of tasks, and present an efficient algorithm for solving the corresponding optimization problem. Experiments on handwritten digit classification, digital art identification, nonlinear inverse image problems, and compressed sensing demonstrate that our approach is effective in large-scale settings, and is well suited to supervised and semi-supervised classification, as well as regression tasks for data that admit sparse representations.

The concept of learning dictionaries in a supervised way was presented in

J. Mairal, F. Bach, J. Ponce, G. Sapiro and A. Zisserman. Supervised Dictionary Learning. In *Advances Neural Information Processing Systems (NIPS)*. 2008

However, the material of the chapter is essentially based on a more recent work:

J. Mairal, F. Bach and J. Ponce. Task-Driven Dictionary Learning. *submitted*. arXiv:1009.5358v1, 2010

## 6.1 Introduction

The classical dictionary learning for reconstructing signals is well adapted to restoration tasks, such as restoring a noisy signal. These dictionaries, which are good at reconstructing clean signals, but bad at reconstructing noise, have indeed led to state-of-the-art denoising algorithms (Elad and Aharon, 2006; Mairal et al., 2008b, 2009c). This "data-driven" unsupervised dictionary learning has also been used for other purposes than

pure signal reconstruction, such as classification, (Grosse et al., 2007; Raina et al., 2007; Kavukcuoglu et al., 2009; Yang et al., 2009; Wright et al., 2009a), but recent works have shown that better results can be obtained when the dictionary is tuned to the specific task (and not just data) it is intended for. Duarte-Carvajalino and Sapiro (2009) have for instance proposed to learn dictionaries for compressed sensing, and Mairal et al. (2008a, 2009b) and Bradley and Bagnell (2009) have learned dictionaries for signal classification. In this chapter, we will refer to this general approach as *task-driven* dictionary learning.

Whereas purely data-driven dictionary learning has been shown to be equivalent to a large-scale matrix factorization problem that can be effectively addressed with several methods (Olshausen and Field, 1997; Engan et al., 1999; Lewicki and Sejnowski, 2000; Aharon et al., 2006; Lee et al., 2007; Mairal et al., 2010b), its task-driven counterpart has proven to be much more difficult to optimize. Presenting a general efficient framework for various task-driven dictionary learning problems is the main topic of this chapter. Even though it is different from existing machine learning approaches, it shares similarities with many of them.

For instance, Blei et al. (2003) have proposed to learn a latent topic model intended for document classification, and in fact Bradley and Bagnell (2009); Jenatton et al. (2010a) have shown that classical dictionary learning techniques could also be used to model text documents. In a different context, Argyriou et al. (2008) have introduced a convex formulation for multi-task classification problems where an orthogonal linear transform of input features is jointly learned with a classifier. Learning compact features has also been addressed in the literature of neural networks, with restricted Boltzmann machines (RBM's) and convolutional neural networks for example (see Lee et al., 2009; Ranzato et al., 2007b,a; LeCun et al., 2006; Larochelle and Bengio, 2008, and references therein). Interestingly, the question of learning the data representation in an unsupervised or supervised way has also been investigated for these approaches. For instance, Blei and McAuliffe (2008) have proposed a supervised topic model and tuning latent data representations for minimizing a cost function is often achieved with *backpropagation* in neural networks (LeCun et al., 1998b).

### 6.1.1 Contributions

This work makes three main contributions:

- It introduces a supervised formulation for learning dictionaries adapted to various tasks instead of dictionaries only adapted to data reconstruction.
- It shows that the resulting optimization problem is smooth under mild assumptions, and empirically that stochastic gradient descent addresses it efficiently.
- It shows that the proposed formulation is well adapted to semi-supervised learning and can exploit unlabeled data when they admit sparse representations, and leads to state-of-the-art results for various machine learning and signal processing problems.

The rest of this chapter is organized as follows: Section 6.2 briefly recalls the classical data-driven dictionary learning framework. Section 6.3 is devoted to our new task-driven framework, and Section 6.4 to efficient algorithms addressing the corresponding optimization problems. Section 6.5 presents several dictionary learning experiments for signal classification, signal regression, and compressed sensing.

## 6.2 Related Work: Data-Driven Dictionary Learning

The classical dictionary learning framework having already been introduced in details in the previous chapters, we just briefly recalls the formulation here to fix the notations. Classical dictionary learning techniques for sparse coding (Olshausen and Field, 1997, 1996; Engan et al., 1999; Lewicki and Sejnowski, 2000; Aharon et al., 2006; Lee et al., 2007) consider a finite training set of signals $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ in $\mathbb{R}^{m \times n}$ and minimize the empirical cost function

$$f_n(\mathbf{D}) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell_u(\mathbf{x}^i, \mathbf{D}),$$

with respect to a dictionary $\mathbf{D}$ in $\mathbb{R}^{m \times p}$, each column representing a dictionary element. Here $\ell_u$ is a loss function such that $\ell_u(\mathbf{x}, \mathbf{D})$ should be small if $\mathbf{D}$ is "good" at representing the signal $\mathbf{x}$ in a sparse fashion. As emphasized by the index $u$ of $\ell_u$, this optimization problem is *unsupervised*. Note that, in this setting, *overcomplete* dictionaries with $p > m$ are allowed. As others (see Lee et al., 2007, for example), we define $\ell_u(\mathbf{x}, \mathbf{D})$ as the optimal value of a sparse coding problem. In this work, we choose the elastic-net formulation of Zou and Hastie (2005):

$$\ell_u(\mathbf{x}, \mathbf{D}) \triangleq \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda_1 \|\boldsymbol{\alpha}\|_1 + \frac{\lambda_2}{2} \|\boldsymbol{\alpha}\|_2^2, \tag{6.1}$$

where $\lambda_1$ and $\lambda_2$ are regularization parameters. When $\lambda_2 = 0$, this leads to the $\ell_1$ sparse decomposition problem, also known as *basis pursuit* (Chen et al., 1998), or *Lasso* (Tibshirani, 1996). Here, our choice of the elastic-net formulation over the Lasso is mainly for stability reasons. Using a parameter $\lambda_2 > 0$ makes the problem of Eq. (6.1) strongly convex and, as shown later in this chapter, ensures its solution to be unique and Lipschitz with respect to $\mathbf{x}$ and $\mathbf{D}$ with a constant depending on $\lambda_2$. Whereas the stability of this solution is not necessarily an issue when learning a dictionary for a reconstruction task, it can be important for other tasks, as shown in the experimental section.

To prevent the $\ell_2$-norm of $\mathbf{D}$ from being arbitrarily large, which would lead to arbitrarily small values of $\boldsymbol{\alpha}$, it is common to constrain its columns $\mathbf{d}^1, \ldots, \mathbf{d}^p$ to have $\ell_2$ norms less than or equal to one. We will call $\mathcal{D}$ the convex set of matrices satisfying this constraint:

$$\mathcal{D} \triangleq \{\mathbf{D} \in \mathbb{R}^{m \times p} \quad \text{s.t.} \quad \forall j \in \{1, \ldots, p\}, \quad \|\mathbf{d}^j\|_2 \leq 1\}. \tag{6.2}$$

As pointed out by Bottou and Bousquet (2008), one is usually not interested in a perfect minimization of the *empirical* cost $f_n(\mathbf{D})$, but instead in the minimization with respect

to $\mathbf{D}$ of the *expected* cost

$$f(\mathbf{D}) \triangleq \mathbb{E}_{\mathbf{x}}[\ell_u(\mathbf{x}, \mathbf{D})] = \lim_{n \to \infty} f_n(\mathbf{D}) \quad \text{a.s.}, \tag{6.3}$$

where the expectation is taken relative to the (unknown) probability distribution $p(\mathbf{x})$ of the data, and is supposed to be finite (see more details in Chapter 2).[1] In practice, dictionary learning problems often involve a large amount of data. For instance when the vectors $\mathbf{x}$ represent image patches, $n$ can be up to several millions in a single image. In this context, online learning techniques have shown to be very efficient for obtaining a stationary point of this optimization problem (Mairal et al., 2010b). Our method exploits this stochastic setting as well, by proposing to minimize an expected cost corresponding to a supervised dictionary learning formulation, which we now present.

## 6.3 Proposed Formulation

We introduce in this section a general framework for learning dictionaries adapted to specific supervised tasks, e.g., classification, as opposed to the unsupervised formulation of the previous section, and present different extensions along with possible applications. We call this approach task-driven dictionary learning.

### 6.3.1 Basic Formulation

Obtaining a good performance in classification tasks is often related to the problem of finding a good data representation (for instance finding a feature space where classes of signals are linearly separable). Sparse decompositions obtained with data-driven learned dictionaries have been used for that purpose by Grosse et al. (2007) and Raina et al. (2007), showing promising results for audio data and natural images. We present in this section a formulation for learning a dictionary in a *supervised* way for prediction tasks (regression or classification).

Concretely, given a learned dictionary $\mathbf{D}$, a vector $\mathbf{x}$ in $\mathcal{X} \subseteq \mathbb{R}^m$ can be represented as $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$, where

$$\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}) \triangleq \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda_1 \|\boldsymbol{\alpha}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}\|_2^2, \tag{6.4}$$

is the elastic-net solution (Zou and Hastie, 2005).

Now suppose that we are interested in predicting a variable $\mathbf{y}$ in $\mathcal{Y}$ from an observation $\mathbf{x}$, where $\mathcal{Y}$ is *either a finite set of labels* for classification tasks, or a *subset of* $\mathbb{R}^q$ *for some integer* $q$ for regression tasks. After learning $\mathbf{D}$ with the unsupervised formulation of Eq. (6.3), and using $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$ as features for predicting the variable $\mathbf{y}$, we can learn model parameters $\mathbf{W}$ by solving the following optimization problem

$$\min_{\mathbf{W} \in \mathcal{W}} f(\mathbf{W}) + \frac{\nu}{2}\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{6.5}$$

[1] We use "a.s." (almost sure) to denote convergence with probability one.

where $\mathcal{W}$ is a convex set, $\nu$ is a regularization parameter, and $f$ is defined as

$$f(\mathbf{W}) \triangleq \mathbb{E}_{\mathbf{y},\mathbf{x}}[\ell_s(\mathbf{y}, \mathbf{W}, \boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}))], \tag{6.6}$$

where $\ell_s$ is a convex loss function, the index $s$ of $\ell_s$ indicating that the loss is adapted to a *supervised* learning problem. The expectation is taken with respect to the unknown probability distribution $p(\mathbf{y}, \mathbf{x})$ of the data. Depending on the setting (classification or regression), several loss functions can be used such as square, logistic, or hinge loss from support vector machines (see Shawe-Taylor and Cristianini, 2004, and references therein), possibly with a particular kernel as done by Raina et al. (2007). Note that we do not specifiy yet the size of the matrix $\mathbf{W}$ since it will depend on the chosen application, as illustrated later in this chapter.

The formulation of Eq. (6.5) exploits features $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$, where $\mathbf{D}$ is learned with the unsupervised formulation from Section 6.2. However, Mairal et al. (2009b) and Bradley and Bagnell (2009) have shown that better results can be achieved when the dictionary is obtained in a fully supervised setting, tuned for the prediction task. We now introduce the task-driven dictionary learning formulation, that consists of *jointly* learning $\mathbf{W}$ and $\mathbf{D}$ by solving

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{W}\in\mathcal{W}} f(\mathbf{D}, \mathbf{W}) + \frac{\nu}{2}\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{6.7}$$

where $\mathcal{D}$ is a set of constraints defined in Eq. (6.2), and $f$ has the form

$$f(\mathbf{D}, \mathbf{W}) \triangleq \mathbb{E}_{\mathbf{y},\mathbf{x}}[\ell_s(\mathbf{y}, \mathbf{W}, \boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}))]. \tag{6.8}$$

The main difficulty of this optimization problem comes from the non-differentiability of $\boldsymbol{\alpha}^\star$, which is the solution of a nonsmooth optimization problem (6.4). Bradley and Bagnell (2009) have tackled this difficulty by introducing a smooth approximation of the sparse regularization which leads to smooth solutions, allowing the use of implicit differentiation to compute the gradient of the cost function they have considered. This approximation encourages some coefficients in $\boldsymbol{\alpha}^\star$ to be small, and does not produce true zeros. It can be used when "true" sparsity is not required. In a different formulation, Mairal et al. (2009b) have used nonsmooth sparse regularization, but used heuristics to tackle the optimization problem. We show in Section 6.4 that better optimization tools than these heuristics can be used, while keeping a nonsmooth regularization for computing $\boldsymbol{\alpha}^\star$.

Small variants of this formulation can also be considered: Non-negativity constraints may be added on $\boldsymbol{\alpha}^\star$ and $\mathbf{D}$, leading to a supervised version of nonnegative matrix factorization (Lee and Seung, 2001), regularized with a sparsity-inducing penalty (Hoyer, 2004). $\ell_s$ could also be a function of $\mathbf{D}$ and $\mathbf{x}$ instead of just $\mathbf{y}, \mathbf{W}, \boldsymbol{\alpha}^\star$. For simplicity, we have omitted these possibilities, but the formulations and algorithms we present in this work can be easily extended to these cases.

Before presenting extensions and applications of the formulation we have introduced, let us first discuss the assumptions under which our analysis will hold.

### Assumptions

From now on, we suppose that:

**(A)** The data $(\mathbf{y}, \mathbf{x})$ admits a probability density $p$ with a compact support $K_{\mathcal{Y}} \times K_{\mathcal{X}} \subseteq \mathcal{Y} \times \mathcal{X}$. This is a reasonable assumption in audio, image, and video processing applications, where it is imposed by the data acquisition process, where values returned by sensors are bounded. To simplify the notations later in the chapter, we suppose from now on that $\mathcal{X}$ and $\mathcal{Y}$ are compact.[2]

**(B)** When $\mathcal{Y}$ is a subset of a finite-dimensional real vector space, $p$ is continuous and $\ell_s$ is twice continuously differentiable.

**(C)** When $\mathcal{Y}$ is a finite set of labels, for all $\mathbf{y}$ in $\mathcal{Y}$, $p(\mathbf{y}, .)$ is continuous and $\ell_s(\mathbf{y}, .)$ is twice continuously differentiable.[3]

Assumptions **(B)** and **(C)** allow us to use several loss functions such as the square, logistic, or softmax losses.

## 6.3.2 Extensions

We now present two extensions of the previous formulations. The first one includes a linear transform of the input data, and the second one exploits unlabeled data in a semi-supervised setting.

### Learning a Linear Transform of the Input Data

In this section, we add to our basic formulation a linear transform of the input features, represented by a matrix $\mathbf{Z}$. Our motivation for this is twofold: It can be appealing to reduce the dimension of the feature space via such a linear transform, and/or it can make the model richer by increasing the numbers of free parameters. The resulting formulation is the following:

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{W}\in\mathcal{W},\mathbf{Z}\in\mathcal{Z}} f(\mathbf{D},\mathbf{W},\mathbf{Z}) + \frac{\nu_1}{2}\|\mathbf{W}\|_{\mathrm{F}}^2 + \frac{\nu_2}{2}\|\mathbf{Z}\|_{\mathrm{F}}^2, \tag{6.9}$$

where $\nu_1$ and $\nu_2$ are two regularization parameters, $\mathcal{Z}$ is a convex set and

$$f(\mathbf{D},\mathbf{W},\mathbf{Z}) \triangleq \mathbb{E}_{\mathbf{y},\mathbf{x}}[\ell_s(\mathbf{y},\mathbf{W},\boldsymbol{\alpha}^\star(\mathbf{Z}\mathbf{x},\mathbf{D}))]. \tag{6.10}$$

It is worth noticing that the formulations of Eq. (6.7) and Eq. (6.9) can also be extended to the case of a cost function depending on several dictionaries involving several sparse coding problems, such as the one used by Mairal et al. (2008a) for signal classification. Such a formulation is not developed here for simplicity reasons, but algorithms to address it can be easily derived from this chapter.

[2] Even though images are acquired in practice after a quantization process, it is a common assumption in image processing to consider pixel values in a continuous space.

[3] For a given value of $\mathbf{y}$ and function $g$, $g(\mathbf{y}, .)$ denotes the function which associates to a vector $\mathbf{x}$ the value $g(\mathbf{y}, \mathbf{x})$.

### Semi-supervised Learning

As shown by Raina et al. (2007), sparse coding techniques can be effective for learning good features from unlabeled data. The extension of our task-driven formulation to the semi-supervised learning setting is quite natural and takes the form

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{W}\in\mathcal{W}} (1-\mu)\mathbb{E}_{\mathbf{y},\mathbf{x}}[\ell_s(\mathbf{y},\mathbf{W},\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D}))] + \mu\mathbb{E}_{\mathbf{x}}[\ell_u(\mathbf{x},\mathbf{D})] + \frac{\nu}{2}\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{6.11}$$

where the second expectation is taken with respect to the marginal distribution of $\mathbf{x}$. The function $\ell_u$ is the loss function defined in Eq. (6.1), and $\mu$ in $[0,1]$ is a new parameter for controlling the trade-off between the unsupervised learning cost function and the supervised learning one. As shown in the experimental section, this parameter is important for classification experiments when only a few labeled training samples are available. Note that this formulation is also compatible with the one of Eq. (6.9).

## 6.3.3 Applications

For illustration purposes, we present four possible applications of our task-driven dictionary learning formulations. Our approach is of course not limited to these examples.

### Regression

In this setting, $\mathcal{Y}$ is a subset of a $q$-dimensional real vector space, and the task is to predict variables $\mathbf{y}$ in $\mathcal{Y}$ from the observation of vectors $\mathbf{x}$ in $\mathcal{X}$. A typical application is for instance the restoration of clean signals $\mathbf{y}$ from observed corrupted signals $\mathbf{x}$. Classical signal restoration techniques often focus on removing additive noise or solving inverse linear problems (Daubechies et al., 2004). When the corruption results from an unknown nonlinear transformation, we formulate the restoration task as a general regression problem. This is the case for example in the experiment presented in Section 6.5.3.

We define the task-driven dictionary learning formulation for regression as follows:

$$\min_{\mathbf{W}\in\mathcal{W},\mathbf{D}\in\mathcal{D}} \mathbb{E}_{\mathbf{y},\mathbf{x}}\Big[\frac{1}{2}\|\mathbf{y}-\mathbf{W}\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})\|_2^2\Big] + \frac{\nu}{2}\|\mathbf{W}\|_{\mathrm{F}}^2. \tag{6.12}$$

At test time, when a new signal $\mathbf{x}$ is observed, the estimate of the corresponding variable $\mathbf{y}$ provided by this model is $\mathbf{W}\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})$ (plus possibly an intercept which we have omitted here for simplicity reasons). Note that we here propose to use the square loss for estimating the difference between $\mathbf{y}$ and its estimate $\mathbf{W}\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})$, but any other twice differentiable loss can be used.

### Binary Classification

In this section and in the next one, we propose to learn dictionaries adapted to classification tasks. Our approach follows the formulation presented by Mairal et al. (2009b), but is slightly different and falls into our task-driven dictionary learning framework. In this setting, the set $\mathcal{Y}$ is equal to $\{-1;+1\}$. Given a vector $\mathbf{x}$, we want to learn the

parameters $\mathbf{w}$ in $\mathbb{R}^p$ of a linear model to predict $y$ in $\mathcal{Y}$, using the sparse representation $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$ as features, and jointly optimize $\mathbf{D}$ and $\mathbf{w}$. For instance, using the logistic regression loss, our formulation becomes

$$\min_{\mathbf{w}\in\mathbb{R}^p, \mathbf{D}\in\mathcal{D}} \mathbb{E}_{y,\mathbf{x}}\Big[\log\left(1+e^{-y\mathbf{w}^\top\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})}\right)\Big] + \frac{\nu}{2}\|\mathbf{w}\|_2^2, \tag{6.13}$$

Once $\mathbf{D}$ and $\mathbf{w}$ have been learned, a new signal $\mathbf{x}$ is classified according to the sign of $\mathbf{w}^\top\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$. For simplicity reasons, we have omitted the intercept in the linear model, but it can easily be included in the formulation. Note that instead of the logistic regression loss, any other twice differentiable loss can be used as well.

As suggested by Mairal et al. (2009b), it is possible to extend this approach with a bilinear model by learning a matrix $\mathbf{W}$ so that a new vector $\mathbf{x}$ is classified according to the sign of $\mathbf{x}^\top\mathbf{W}\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$. In this setting, our formulation becomes

$$\min_{\mathbf{W}\in\mathbb{R}^{m\times p}, \mathbf{D}\in\mathcal{D}} \mathbb{E}_{y,\mathbf{x}}\Big[\log\left(1+e^{-y\mathbf{x}^\top\mathbf{W}\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})}\right)\Big] + \frac{\nu}{2}\|\mathbf{W}\|_{\mathrm{F}}^2. \tag{6.14}$$

This bilinear model requires learning $pm$ parameters as opposed to the $p$ parameters of the linear one. It is therefore richer and can sometimes offer a better classification performance when the linear model is not rich enough to explain the data, but it might be more subject to overfitting.

Note that we have naturally presented the binary classification task using the logistic regression loss, but as we have experimentally observed, the square loss is also an appropriate choice in many situations.

### Multi-class Classification

When $\mathcal{Y}$ is a finite set of labels in $\{1, \ldots, q\}$ with $q > 2$, extending the previous formulation to the multi-class setting can be done in several ways, which we briefly describe here. The simplest possibility is to use a set of binary classifiers presented in Eq. (6.13) in a "one-vs-all" or "one-vs-one" scheme. Another possibility is to use a multi-class cost function such as the soft-max function, to find linear predictors $\mathbf{w}_k$, $k$ in $\{1, \ldots, q\}$ such that for a vector $\mathbf{x}$ in $\mathcal{X}$, the quantities $\mathbf{w}_y^\top\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$ are encouraged to be greater than $\mathbf{w}_k^\top\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$ for all $k \neq y$.

Another possibility is to turn the multi-class classification problem into a regression one and consider that $\mathcal{Y}$ is a set of $q$ binary vectors of dimension $q$ such that the $k-$th vector has 1 on its $k$-th coordinate, and 0 elsewhere. This allows using the regression formulation of Eq. (6.12) to solve the classification problem.

### Learning Compressed sensing

Let us consider a signal $\mathbf{x}$ in $\mathbb{R}^m$, the theory of compressed sensing (Candes, 2006; Donoho, 2006) tells us that under certain assumptions, the vector $\mathbf{x}$ can be recovered exactly from a few measurements $\mathbf{Z}\mathbf{x}$, where $\mathbf{Z}$ in $\mathbb{R}^{r\times m}$ is called a "sensing" matrix with $r \ll m$. Unlike classical signal processing methods, such a linear transformation is

sometimes physically included in the data acquisition process itself (Duarte et al., 2008), meaning that a sensor can provide measurements $\mathbf{Zx}$ without directly measuring $\mathbf{x}$ at any moment.

In a nutshell, the recovery of $\mathbf{x}$ has been proven to be possible when $\mathbf{x}$ admits a sparse representation on a dictionary $\mathbf{D}$, and the sensing matrix $\mathbf{Z}$ is incoherent with $\mathbf{D}$, meaning that the rows of $\mathbf{Z}$ are sufficiently uncorrelated with the columns of $\mathbf{D}$ (see Candes, 2006; Donoho, 2006 for more details).[4] To ensure this condition to be satisfied, $\mathbf{Z}$ is often chosen to be a random matrix, which is incoherent with any dictionary with high probability.

The choice of a random matrix is appealing for many reasons. In addition to the fact that it provides theoretical guarantees of incoherence, it is well suited to the case where it impossible to store a deterministic matrix $\mathbf{Z}$ into memory (when $m$ is large for instance), whereas it is sufficient to store the seed of a random process to generate a random matrix. On the other hand, large signals can often be cut into smaller parts that still admit sparse decompositions, e.g., image patches, which can be treated independently with a deterministic smaller matrix $\mathbf{Z}$. When this is the case or when $m$ has a reasonable size, the question of whether to use a deterministic matrix $\mathbf{Z}$ or a random one arises, and it has been empirically observed that learned matrices $\mathbf{Z}$ can outperform classical random projections: For example, Weiss et al. (2007) have shown that dimensionality reduction techniques such as principal component analysis (PCA) or independent component analysis (ICA) could do better than random projections in noisy settings, and Duarte-Carvajalino and Sapiro (2009) have shown that jointly learning sensing matrices and dictionaries can do even better in certain cases. Seeger (2008) has also proposed a bayesian framework for learning sensing matrices in compressed sensing applications.

Following the latter authors, we study the case where $\mathbf{Z}$ is not random but learned at the same time as the dictionary, and introduce a formulation which falls into out task-driven dictionary learning framework:

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{W}\in\mathbb{R}^{m\times p},\mathbf{Z}\in\mathbb{R}^{r\times m}} \mathbb{E}_{\mathbf{y},\mathbf{x}}\Big[\frac{1}{2}\|\mathbf{y}-\mathbf{W}\boldsymbol{\alpha}^\star(\mathbf{Zx},\mathbf{D})\|_2^2\Big] + \frac{\nu_1}{2}\|\mathbf{W}\|_{\mathrm{F}}^2 + \frac{\nu_2}{2}\|\mathbf{Z}\|_{\mathrm{F}}^2, \tag{6.15}$$

where we learn $\mathbf{D}$, $\mathbf{W}$ and $\mathbf{Z}$ so that the variable $\mathbf{y}$ should be well reconstructed when encoding the "sensed" signal $\mathbf{Zx}$ with a dictionary $\mathbf{D}$. In a noiseless setting, $\mathbf{y}$ is naturally set to the same value as $\mathbf{x}$. In a noisy setting, it can be a corrupted version of $\mathbf{x}$.

After having presented our general task-driven dictionary learning formulation, we present next a strategy to address the corresponding nonconvex optimization problem.

[4] The assumption of "incoherence" between $\mathbf{D}$ and $\mathbf{Z}$ can be replaced with a different but related hypothesis called *restricted isometry property*. Again the reader should refer to (Candes, 2006; Donoho, 2006) for more details.

## 6.4 Optimization

We first show that the cost function $f$ of our basic formulation (6.7) is differentiable and compute its gradient. Then, we refine the analysis for the different variations presented in the previous section, and describe an efficient online learning algorithm to address them.

### 6.4.1 Differentiability of $f$

We analyse the differentiability of $f$ as defined in Eq. (6.7) with respect to its two arguments $\mathbf{D}$ and $\mathbf{W}$. We consider here the case where $\mathcal{Y}$ is a compact subset of a finite dimensional real vector space, but all formulas and proofs are similar when $\mathcal{Y}$ is a finite set of labels. The purpose of this section is to show that even though the sparse coefficients $\boldsymbol{\alpha}^\star$ are obtained by solving a non-differentiable optimization problem, $f$ is differentiable on $\mathcal{W} \times \mathcal{D}$, and one can compute its gradient.

The main argument in the proof of Propositions 12 and 13 below is that, although the function $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$ is not differentiable, it is uniform Lipschitz continuous, and differentiable almost everywhere. The only points where $\boldsymbol{\alpha}^\star$ is not differentiable correspond to points where the set of nonzero coefficients of $\boldsymbol{\alpha}^\star$, which we always denote by $\Lambda$ in this chapter, change. Considering optimality conditions of the elastic-net formulation of Eq. (6.1), these points are easy to characterize:

**Lemma 12 (Optimality conditions of the elastic net)**
*$\boldsymbol{\alpha}^\star$ is a solution of Eq. (6.4) if and only if for all $j$ in $\{1, \ldots, p\}$,*

$$\begin{aligned} \mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star) - \lambda_2 \boldsymbol{\alpha}_j &= \lambda_1 \operatorname{sign}(\boldsymbol{\alpha}_j^\star) \quad \textit{if} \quad \boldsymbol{\alpha}_j^\star \neq 0, \\ |\mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star) - \lambda_2 \boldsymbol{\alpha}_j^\star| &\leq \lambda_1 \quad \textit{otherwise.} \end{aligned} \tag{6.16}$$

*Denoting by $\Lambda \triangleq \{j \in \{1, \ldots, p\} \quad s.t. \quad \boldsymbol{\alpha}_j^\star \neq 0\}$ the active set, we also have*

$$\boldsymbol{\alpha}_\Lambda^\star = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I})^{-1}(\mathbf{D}_\Lambda^\top \mathbf{x} - \lambda_1 \mathbf{s}_\Lambda), \tag{6.17}$$

*where $\mathbf{s}_\Lambda$ in $\{-1; +1\}^{|\Lambda|}$ carries the signs of $\boldsymbol{\alpha}_\Lambda^\star$.*

The proof of this lemmas directly comes from the optimality conditions we have presented in Section 1.4.1. The next proposition exploits these optimality conditions to characterize the regularity of $\boldsymbol{\alpha}^\star$.

**Proposition 11 (Regularity of the elastic net solution)**
*Assume $\lambda_2 > 0$ and **(A)**. Then,*

1. *The function $\boldsymbol{\alpha}^\star$ is uniformly Lipschitz on $\mathcal{X} \times \mathcal{D}$.*
2. *Let $\mathbf{D}$ be in $\mathcal{D}$, $\varepsilon$ be a positive scalar and $\mathbf{s}$ be a vector in $\{-1, 0, +1\}^p$, and define $K_\mathbf{s}(\mathbf{D}, \varepsilon) \subseteq \mathcal{X}$ as: $\mathbf{x}$ is in $K_\mathbf{s}(\mathbf{D}, \varepsilon)$ if and only if*

$$\forall j \in \{1, \ldots, p\}, \begin{cases} |\mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star) - \lambda_2 \boldsymbol{\alpha}_j^\star| \leq \lambda_1 - \varepsilon & \textit{if} \quad \mathbf{s}_j = 0, \\ \mathbf{s}_j \boldsymbol{\alpha}_j^\star \geq \varepsilon & \textit{if} \quad \mathbf{s}_j \neq 0. \end{cases} \tag{6.18}$$

*where* $\boldsymbol{\alpha}^\star$ *is shorthand for* $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$.

*Then, there exists* $\kappa > 0$ *independent of* $\mathbf{s}$, $\mathbf{D}$ *and* $\varepsilon$ *so that for all* $\mathbf{x}$ *in* $K_{\mathbf{s}}(\mathbf{D}, \varepsilon)$, *the function* $\boldsymbol{\alpha}^\star$ *is twice continuously differentiable on* $B_{\kappa\varepsilon}(\mathbf{x}) \times B_{\kappa\varepsilon}(\mathbf{D})$, *where* $B_{\kappa\varepsilon}(\mathbf{x})$ *and* $B_{\kappa\varepsilon}(\mathbf{D})$ *denote the open balls of radius* $\kappa\varepsilon$ *respectively centered on* $\mathbf{x}$ *and* $\mathbf{D}$.

With these results in hand, we then show that $f$ admits a first-order Taylor expansion meaning that it is differentiable. Indeed, the sets where $\boldsymbol{\alpha}^\star$ is not differentiable being negligible in the expectation from the definition of $f$ in Eq. (6.8). We can now state our main result:

**Proposition 12** (**Differentiability and gradients of** $f$)
*Assume* $\lambda_2 > 0$, ***(A)***, ***(B)*** *and* ***(C)***. *Then, the function* $f$ *defined in Eq. (6.7) is differentiable, and*

$$\begin{cases} \nabla_{\mathbf{W}} f(\mathbf{D}, \mathbf{W}) = \mathbb{E}_{\mathbf{y},\mathbf{x}}[\nabla_{\mathbf{W}} \ell_s(\mathbf{y}, \mathbf{W}, \boldsymbol{\alpha}^\star)], \\ \nabla_{\mathbf{D}} f(\mathbf{D}, \mathbf{W}) = \mathbb{E}_{\mathbf{y},\mathbf{x}}[-\mathbf{D}\boldsymbol{\beta}^\star \boldsymbol{\alpha}^{\star\top} + (\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)\boldsymbol{\beta}^{\star\top}], \end{cases} \tag{6.19}$$

*where* $\boldsymbol{\alpha}^\star$ *is short for* $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$, *and* $\boldsymbol{\beta}^\star$ *is a vector in* $\mathbb{R}^p$ *that depends on* $\mathbf{y}, \mathbf{x}, \mathbf{W}, \mathbf{D}$ *with*

$$\boldsymbol{\beta}^\star_{\Lambda^C} = 0 \qquad \text{and} \qquad \boldsymbol{\beta}^\star_\Lambda = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I})^{-1} \nabla_{\boldsymbol{\alpha}_\Lambda} \ell_s(\mathbf{y}, \mathbf{W}, \boldsymbol{\alpha}^\star), \tag{6.20}$$

*where* $\Lambda$ *denotes the indices of the nonzero coefficients of* $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$.

The proof of this proposition is given in Appendix B. We have shown that the function defined in Eq. (6.7) is smooth, and computed its gradients. The same can be done for the more general formulation defined by Eq. (6.10):

**Proposition 13** (**Differentiability and gradients for the extended formulation**)
*Assume* $\lambda_2 > 0$, ***(A)***, ***(B)*** *and* ***(C)***. *Then, the function* $f$ *defined in Eq. (6.10) is differentiable. The gradients of* $f$ *are*

$$\begin{cases} \nabla_{\mathbf{W}} f(\mathbf{D}, \mathbf{W}, \mathbf{Z}) = \mathbb{E}_{\mathbf{y},\mathbf{x}}[\nabla_{\mathbf{W}} \ell_s(\mathbf{y}, \mathbf{W}, \boldsymbol{\alpha}^\star)], \\ \nabla_{\mathbf{D}} f(\mathbf{D}, \mathbf{W}, \mathbf{Z}) = \mathbb{E}_{\mathbf{y},\mathbf{x}}[-\mathbf{D}\boldsymbol{\beta}^\star \boldsymbol{\alpha}^{\star\top} + (\mathbf{Z}\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)\boldsymbol{\beta}^{\star\top}], \\ \nabla_{\mathbf{Z}} f(\mathbf{D}, \mathbf{W}, \mathbf{Z}) = \mathbb{E}_{\mathbf{y},\mathbf{x}}[\mathbf{D}\boldsymbol{\beta}^\star \mathbf{x}^\top], \end{cases} \tag{6.21}$$

*where* $\boldsymbol{\alpha}^\star$ *is short for* $\boldsymbol{\alpha}^\star(\mathbf{Z}\mathbf{x}, \mathbf{D})$, *and* $\boldsymbol{\beta}^\star$ *is defined in Eq. (6.20).*

The proof is similar to the one of Proposition 12 in Appendix B, and uses similar arguments.

### 6.4.2 Algorithm

Stochastic gradient descent algorithms are typically designed to minimize functions whose gradients have the form of an expectation as in Eq. (6.19). They have been shown to converge to stationary points of (possibly nonconvex) optimization problems under a few assumptions that are a bit stricter than the ones satisfied in this chapter (see

Bottou and Bousquet, 2008, and references therein).[5] As we have noted in Chapter 2, these algorithms are generally well suited to unsupervised dictionary learning when their learning rate is well tuned.

The method we propose here is a projected first-order stochastic gradient algorithm (see Kushner and Yin, 2003), and it is given in Algorithm 8. It sequentially draws i.i.d samples $(\mathbf{y}^t, \mathbf{x}^t)$ from the probability distribution $p(\mathbf{y}, \mathbf{x})$. Obtaining such i.i.d. samples may be difficult since the density $p(\mathbf{y}, \mathbf{x})$ is unknown. At first approximation, the vectors $(\mathbf{y}^t, \mathbf{x}^t)$ are obtained in practice by cycling over a randomly permuted training set, which is often done in similar machine learning settings (Bottou, 1998).

**Algorithm 8** Stochastic gradient descent algorithm for task-driven dictionary learning.

**Require:** $p(\mathbf{y}, \mathbf{x})$ (a way to draw i.i.d samples of $p$), $\lambda_1, \lambda_2, \nu \in \mathbb{R}$ (regularization parameters), $\mathbf{D} \in \mathcal{D}$ (initial dictionary), $\mathbf{W} \in \mathcal{W}$ (initial parameters), $T$ (number of iterations), $t_0, \rho$ (learning rate parameters).

1: **for** $t = 1$ to $T$ **do**

2: Draw $(\mathbf{y}^t, \mathbf{x}^t)$ from $p(\mathbf{y}, \mathbf{x})$.

3: Sparse coding: compute $\boldsymbol{\alpha}^\star$ using a modified LARS (Efron et al., 2004).

$$\boldsymbol{\alpha}^\star \leftarrow \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{x}^t - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda_1 \|\boldsymbol{\alpha}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}\|_2^2.$$

4: Compute the active set: $\Lambda \leftarrow \{j \in \{1, \ldots, p\} : \boldsymbol{\alpha}_j^\star \neq 0\}$.

5: Compute $\boldsymbol{\beta}^\star$:

$$\boldsymbol{\beta}^\star_{\Lambda^C} = 0 \qquad \text{and} \qquad \boldsymbol{\beta}^\star_{\Lambda} = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I})^{-1} \nabla_{\boldsymbol{\alpha}_\Lambda} \ell_s(\mathbf{y}_t, \mathbf{W}, \boldsymbol{\alpha}^\star).$$

6: Choose the learning rate $\rho_t \leftarrow \min\left(\rho, \rho \frac{t_0}{t}\right)$.

7: Update the parameters by a projected gradient step

$$\mathbf{W} \leftarrow \Pi_{\mathcal{W}}\Big[\mathbf{W} - \rho_t(\nabla_{\mathbf{W}} \ell_s(\mathbf{y}^t, \mathbf{W}, \boldsymbol{\alpha}^\star) + \nu \mathbf{W})\Big],$$
$$\mathbf{D} \leftarrow \Pi_{\mathcal{D}}\Big[\mathbf{D} - \rho_t(-\mathbf{D}\boldsymbol{\beta}^\star \boldsymbol{\alpha}^{\star\top} + (\mathbf{x}^t - \mathbf{D}\boldsymbol{\alpha}^\star)\boldsymbol{\beta}^{\star\top})\Big],$$

where $\Pi_{\mathcal{W}}$ and $\Pi_{\mathcal{D}}$ are respectively orthogonal projections on the sets $\mathcal{W}$ and $\mathcal{D}$.

8: **end for**

9: **return** $\mathbf{D}$ (learned dictionary).

At each iteration, the sparse code $\boldsymbol{\alpha}^\star(\mathbf{x}^t, \mathbf{D})$ is computed by solving the elastic-net formulation of Zou and Hastie (2005). We have chosen to use the LARS algorithm,

[5] As often done in machine learning, we use stochastic gradient descent in a setting where it is not guaranteed to converge in theory, but is has proven to behave well in practice, as shown in our experiments. The convergence proof of Bottou and Bousquet (2008) for non-convex problems indeed assumes three times differentiable cost functions.

a homotopy method (Osborne et al., 2000b; Efron et al., 2004), which was originally developed to solve the Lasso formulation—that is, $\lambda_2 = 0$, but which can be modified to solve the elastic-net problem. Interestingly, it admits an efficient implementation that provides a Cholesky decomposition of the matrix $(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I})^{-1}$ (see Efron et al., 2004; Zou and Hastie, 2005) as well as the solution $\boldsymbol{\alpha}^\star$. In this setting, $\boldsymbol{\beta}^\star$ can be obtained without having to solve from scratch a new linear system.

The learning rate $\rho_t$ is chosen according to a heuristic rule. Several strategies have been presented in the literature (see LeCun et al., 1998b; Murata, 1999, and references therein). A classical setting uses a learning rate of the form $\rho/t$, where $\rho$ is a constant.[6] However, such a learning rate is known to decrease too quickly in many practical cases, and one sometimes prefers a learning rate of the form $\rho/(t+t_0)$, which requires tuning two parameters. In this chapter, we have chosen a learning rate of the form $\min(\rho, \rho t_0/t)$—that is, a constant learning rate $\rho$ during $t_0$ iterations, and a $1/t$ annealing strategy when $t > t_0$, a strategy used by Murata (1999) for instance. Finding good parameters $\rho$ and $t_0$ also requires in practice a good heuristic. The one we have used successfully in all our experiments is $t_0 = T/10$, where $T$ is the total number of iterations. Then, we try several values of $\rho$ during a few hundreds of iterations and keep the one that gives the lowest error on a small validation set.

In practice, one can also improve the convergence speed of our algorithm with a mini-batch strategy—that is, by drawing $\eta > 1$ samples at each iteration instead of a single one. This is a classical heuristic in stochastic gradient descent algorithms and, in our case, this is further motivated by the fact that solving $\eta$ elastic-net problems with the same dictionary $\mathbf{D}$ can be accelerated by the precomputation of the matrix $\mathbf{D}^\top \mathbf{D}$ when $\eta$ is large enough. Such a strategy is also used by Mairal et al. (2010b) for the classical data-driven dictionary learning approach. In practice, we use the value $\eta = 200$ in all our experiments (a value found to be good for the unsupervised setting as well).

As many algorithms tackling non-convex optimization problems, our method for learning supervised dictionaries can lead to poor results if is not well initialized. The classical unsupervised approach of dictionary learning presented in Eq. (6.3) has been found empirically to be better behaved than the supervised one, and easy to initialize (Mairal et al., 2010b). We therefore have chosen to initialize our dictionary $\mathbf{D}$ by addressing the unsupervised formulation of Eq. (6.3) using the SPAMS toolbox (Mairal et al., 2010b).[7] With this initial dictionary $\mathbf{D}$ in hand, we optimize with respect to $\mathbf{W}$ the cost function of Eq (6.5), which is convex. This procedure gives us a pair $(\mathbf{D}, \mathbf{W})$ of parameters which are used to initialize Algorithm 8.

### 6.4.3 Extensions

We here present the slight modifications to Algorithm 8 necessary to address the two extensions discussed in Section 6.3.2.

[6] A $O(1/t)$ asymptotic learning rate is usually used for proving the convergence of stochastic gradient descent algorithms (Bottou and Bousquet, 2008).

[7] http://www.di.ens.fr/willow/SPAMS/

The last step of Algorithm 8 updates the parameters $\mathbf{D}$ and $\mathbf{W}$ according to the gradients presented in Eq. (6.21). Modifying the algorithm to address the formulation of Eq. (6.13) also requires updating the parameters $\mathbf{Z}$ according to the gradient from Proposition 13:

$$\mathbf{Z} \leftarrow \Pi_{\mathcal{Z}}\Big[\mathbf{Z} - \rho_t(\mathbf{D}\boldsymbol{\beta}^\star\mathbf{x}^\top + \nu_2\mathbf{Z})\Big],$$

where $\Pi_{\mathcal{Z}}$ denotes the orthogonal projection on the set $\mathcal{Z}$.

The extension to the semi-supervised formulation of Eq. (6.11) assumes that one can draw samples from the marginal distribution $p(\mathbf{x})$. This is done in practice by cycling over a randomly permuted set of unlabeled vectors. Extending Algorithm 8 to this setting requires the following modifications: At every iteration, we draw one pair $(\mathbf{y}^t, \mathbf{x}^t)$ from $p(\mathbf{y}, \mathbf{x})$ and one sample $\mathbf{x}'^t$ from $p(\mathbf{x})$. We proceed exactly as in Algorithm 8, except that we also compute $\boldsymbol{\alpha}^{\star\prime} \triangleq \boldsymbol{\alpha}^\star(\mathbf{x}'^t, \mathbf{D})$, and replace the update of the dictionary $\mathbf{D}$ by

$$\mathbf{D} \leftarrow \Pi_{\mathcal{D}}\Big[\mathbf{D} - \rho_t\Big((1-\mu)\big(-\mathbf{D}\boldsymbol{\beta}^\star\boldsymbol{\alpha}^{\star\top} + (\mathbf{x}^t - \mathbf{D}\boldsymbol{\alpha}^\star)\boldsymbol{\beta}^{\star\top}\big) + \mu\big(-(\mathbf{x}'^t - \mathbf{D}\boldsymbol{\alpha}^{\star\prime})\boldsymbol{\alpha}^{\star\prime\top}\big)\Big)\Big],$$

where the term $-(\mathbf{x}'^t - \mathbf{D}\boldsymbol{\alpha}^{\star\prime})\boldsymbol{\alpha}^{\star\prime\top}$ is in fact the gradient $\nabla_{\mathbf{D}}\ell_u(\mathbf{x}'^t, \mathbf{D})$ as shown in (Mairal et al., 2010b, Proposition 1).

## 6.5 Experimental Validation

Before presenting our experiments, we briefly discuss the important question of choosing the parameters of our approach.

### 6.5.1 Choosing the Parameters

Performing cross-validation on the parameters $\lambda_1$, $\lambda_2$ (elastic-net parameters), $\nu$ (regularization parameter) and $p$ (size of the dictionary) would of course be cumbersome, and we use a few simple heuristics to either reduce the search space for these parameters or fix arbitrarily some of them. We have proceeded in the following way:

- Since we want to exploit sparsity, we often set $\lambda_2$ to 0, even though $\lambda_2 > 0$ is necessary in our analysis for proving the differentiability of our cost function. This has proven to give satisfactory results in most experiments, except for the experiment of Section 6.5.5, where choosing a small positive value for $\lambda_2$ was necessary for our algorithm to converge.

- We have empirically observed that natural image patches (that are preprocessed to have zero-mean and unit $\ell_2$-norm) are usually well reconstructed with values of $\lambda_1$ around 0.15 (a value used by Mairal et al. (2009b) for instance), and that one only needs to test a few different values, for instance $\lambda_1 = 0.15 + 0.025k$, with $k \in \{-3, \ldots, 3\}$.

- When there is a lot of training data, which is often the case for natural image patches, the regularization with $\nu$ becomes unnecessary and this parameter can

arbitrarily set to a small value, e.g., $\nu = 10^{-9}$ for normalized input data. When there are not many training points, this parameter is set up by cross-validation.

- We have also observed that a larger dictionary usually means a better performance, but a higher computational cost. Setting the size of the dictionary is therefore often a trade-off between results quality and efficiency. In our experiments, we often try the values $p$ in $\{50, 100, 200, 400\}$.

We show in this section several applications of our method to real problems, starting with handwritten digits classification, then moving to the restoration of images that are damaged by an unknown nonlinear transformation, digital art authentification, and compressed sensing.

### 6.5.2 Handwritten Digits Classification

We choose in this section a discriminative task that consists of classifying digits from the MNIST (LeCun et al., 1998a) and USPS (LeCun et al., 1990) handwritten datasets. MNIST is made of 70 000 $28 \times 28$ images, 60 000 for training, 10 000 for testing, each of them containing one handwritten digit. USPS is composed of 7291 training images and 2007 test images of size $16 \times 16$.

We choose to address this multiclass classification problem with a one-vs-all strategy, learning independently 10 dictionaries and classifiers per class, using the formulation of Eq. (6.13). This approach has proven in our case to be more scalable than learning a single large dictionary with a multi-class loss, while providing very good results. We use the Lasso formulation for encoding the digits—that is, we set $\lambda_2 = 0$. All the digits are preprocessed to have zero-mean and are normalized to have unit $\ell_2$-norm. We try the parameters $\lambda_1 = 0.15 + 0.025k$, with $k \in \{-3, \ldots, 3\}$, for the reasons mentioned in the previous section, and $\nu$ in $\{10^{-1}, \ldots, 10^{-6}\}$. For choosing the parameters, we use for MNIST the last 10 000 digits of the training set for validation, and train on the first 50 000 ones. For USPS, we proceed in a similar way, keeping 10% of the training set for validation. Note that a full cross-validation scheme may give better results, but it would be computationally more expensive.

Most effective digit recognition techniques in the literature use features with shift invariance properties (Haasdonk and Keysers, 2002; Ranzato et al., 2007a). Since our formulation is less sophisticated than for instance the convolutional network architecture of Ranzato et al. (2007a) and does not enjoy such properties, we have chosen to artificially augment the size of our training set by considering versions of the digits that are shifted by one pixel in every direction. This is of course not an optimal way of introducing shift invariance in our framework, but it is fairly simple.

After choosing the parameters using the validation set (and of course none of the testing set), we retrain our model on the full training set. Each experiment is performed with 40 000 iterations of our algorithm with a mini-batch of size 200. To study the influence of the dictionary size, we report on Table 6.1 the performance on the test set achieved for different dictionary sizes, with $p$ in $\{50, 100, 200, 300\}$ for the two datasets.

| **D** | unsupervised | | | | supervised | | | |
|---|---|---|---|---|---|---|---|---|
| $p$ | 50 | 100 | 200 | 300 | 50 | 100 | 200 | 300 |
| MNIST | 5.27 | 3.92 | 2.95 | 2.36 | .96 | .73 | .57 | .54 |
| USPS | 8.02 | 6.03 | 5.13 | 4.58 | 3.64 | 3.09 | 2.88 | 2.84 |

Table 6.1: Test error in percent of our method for the digit recognition task for different dictionary sizes $p$.

We observe that learning $\mathbf{D}$ in a supervised way significantly improves the performance of the classification. Moreover our method achieves state-of-the-art results on MNIST with a 0.54% error rate, which is similar to the 0.60% error rate of Ranzato et al. (2007a).[8] Our 2.84% error rate on USPS is slightly behind the 2.4% error rate of Haasdonk and Keysers (2002).

Our second experiment follows Ranzato et al. (2007a), where only a small percentage of the training samples are actually labelled. We use the semi-supervised formulation of Eq. (6.11) which exploits unlabeled data. Unlike the first experiment where the parameters are chosen using a validation set, and following Ranzato et al. (2007a), we make a few arbitrary choices. Indeed, we use $p = 300$, $\lambda_1 = 0.075$, $\lambda_2 = 0$ and $\nu = 10^-5$, which were the parameters chosen by the validation procedure in the previous experiment. The dictionaries associated with each digit class are initialized using the unsupervised formulation of Section 6.2. To measure the performance of our algorithm for different values of $\mu$, we use a continuation strategy: Starting with $\mu = 1.0$, we sequentially decrease its value by 0.1 until we have $\mu = 0$, learning with 10 000 iterations for each new value of $\mu$. We report the corresponding error rates in Figure 6.1, showing that our approach offers a competitive performance similar to Ranzato et al. (2007a). Indeed, the best error rates of our method for $n = 300, 1000, 5000$ unlabeled data are respectively $5.81, 3.55$ and $1.81\%$, which is similar to Ranzato et al. (2007a) who has reported $7.18, 3.21$ and $1.52\%$ with the same sets of labeled data.

### 6.5.3 Learning a Nonlinear Image Mapping

To illustrate our method in the context of regression, we consider a classical image processing task called "inverse halftoning". With the development of several *binary* display technologies in the 70s (including, for example, printers and PC screens), the problem of converting a grayscale continuous-tone image into a binary one that looks perceptually similar to the original one ("halftoning") was posed to the image processing community. Examples of halftoned images obtained with the classical algorithm of Floyd and Steinberg (1976) are presented in the second column of Figure 6.2, with original images in the first column. Restoring these binary images to continuous-tone ones ("inverse halftoning") has become a classical problem (see Dabov et al., 2006, and references therein).

[8] It is also shown by Ranzato et al. (2007a) that better results can be achieved by considering deformations of the training set.

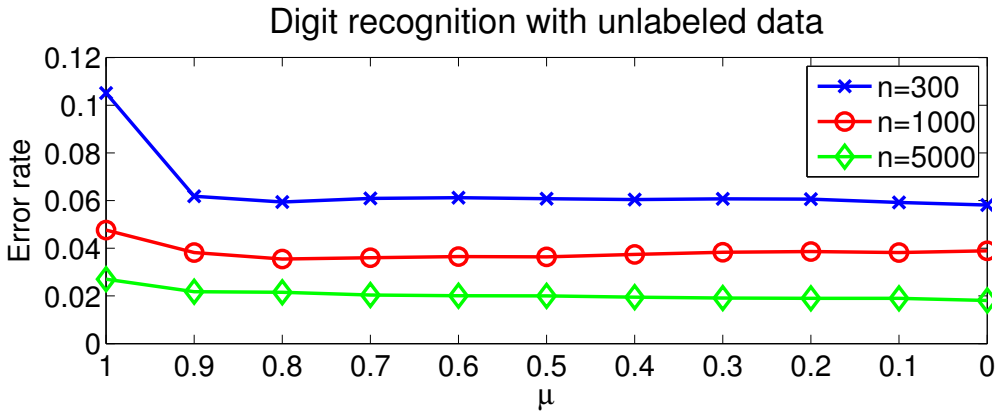



Figure 6.1: Error rates on MNIST when using $n$ labeled data, for various values of $\mu$.

Unlike most image processing approaches that address this second problem by explicitly modeling the halftoning process, we formulate it as a signal regression problem, *without exploiting any prior on the task*. We use a database of 36 images; 24 are high-quality images from the Kodak PhotoCD dataset[9] and are used for training, and 12 are classical images often used for evaluating image processing algorithms, which can be found in Chapter 1; the first four (house, peppers, cameraman, lena) are used for validation and the remaining eight for testing.

We apply the Floyd-Steinberg algorithm implemented in the LASIP Matlab toolbox[10] to the grayscale continuous-tone images in order to build our training / validation / testing set. We extract all pairs of patches from the original/halftoned images in the training set, which provides us with a database of approximately 9 millions of patches. We then use the "signal regression" formulation of Eq. (6.12) to learn a dictionary $\mathbf{D}$ and model parameters $\mathbf{W}$, by performing two passes of our algorithm over the 9 million training pairs.

At this point, we have learned how to restore a small patch from an image, but not yet how to restore a full image. Following other patch-based approaches to image restoration (Elad and Aharon, 2006), we extract from a test image all patches including overlaps, and restore each patch *independently* so that we get different estimates for each pixel (one estimate for each patch the pixel belongs to). The estimates are then averaged to reconstruct the full image. Estimating each patch independently and then averaging the estimates is of course not optimal, but it gives very good results in many image restoration tasks (see Elad and Aharon, 2006; Mairal et al., 2009c, and references therein). The final image is then post-processed using the denoising algorithm of Mairal et al. (2009c) to remove possible artefacts.

To evaluate our method qualitatively, we measure how well it reconstructs the conti-

[9] http://r0k.us/graphics/kodak/
[10] http://www.cs.tut.fi/~lasip/

nuous-tone images of the test set from the halftoned ones. To reduce a bit the number of hyperparameters of our model, we have made a few arbitrary choices: In this experiment, the Lasso of Tibshirani (1996) is also preferred to the elastic-net formulation of Zou and Hastie (2005), and $\lambda_2$ is set to 0. Even though $\lambda_2 > 0$ is necessary in our analysis, $\lambda_2 = 0$ has proven to give satisfactory results in this experiment (this is, however, *not true* in general: For example, the experiments of Section 6.5.5 require $\lambda_2 > 0$ for our algorithm to converge). Thus, with millions of training samples, our model is unlikely to overfit and the regularization parameter $\nu$ is set to 0 as well. The remaining free parameters of our approach are the size $m$ of the patches, the size $p$ of the dictionary and the regularization parameter $\lambda_1$. We have optimized these parameters by measuring the mean-squared error reconstruction on the validation set. We have tried patches of size $m = l \times l$, with $l \in \{6, 8, 10, 12, 14, 16\}$, dictionaries of sizes $p = 100$, 250 and 500 , and determined $\lambda_1$ by first trying values on the logarithmic scale $10^i$, $i = -3, \ldots, 2$, then refining this parameter on the scale $0.1, 0.2, 0.3, \ldots, 1.0$. The best parameters found are $m = 10 \times 10$, $p = 500$ and $\lambda_1 = 0.6$. Since the test procedure is slightly different from the training one (the test includes an averaging step to restore a full image whereas the training one does not), we have refined the value of $\lambda_1$, trying different values one an additive scale $\{0.4, 0.45, \ldots, 0.75, 0.8\}$, and selected the value $\lambda_1 = 0.55$, which has given the best result on the validation set.

Note that the largest dictionary has been chosen, and better results could potentially be obtained using an even larger dictionary, but this would imply a higher computational cost. Examples of results are presented in Figure 6.2. Halftoned images are binary but look perceptually similar to the original image. Reconstructed images have very few artefacts and most details are well preserved. We report in Table 6.2 a quantitative comparison between our approach and various ones from the literature, including the state-of-the-art algorithm of (Dabov et al., 2006), which had until now the best results on this dataset. Even though our method does not explicitly model the transformation, it leads to better results in terms of PSNR.[11] We also present in Figure 6.3 the results obtained by applying our algorithm to various binary images found on the web, from which we do not know the ground truth, and which have not necessarily been obtained with the Floyd-Steinberg algorithm. The results are qualitatively rather good.

From this experiment, we conclude that our method is well suited to (at least, some) nonlinear regression problems on natural images, paving the way to new applications of sparse image coding.

### 6.5.4 Digital Art Identification

Recognizing authentic paintings from imitations using statistical techniques has been the topic of a few recent works (Lyu et al., 2004; Johnson et al., 2008; Hugues et al., 2009). Classical methods compare, for example, the kurtosis of wavelet coefficients between a

[11] Denoting by MSE the mean-squared-error for images whose intensities are between 0 and 255, the PSNR is defined as PSNR $= 10 \log_{10}(255^2/\text{MSE})$ and is measured in dB. A gain of 1dB reduces the MSE by approximately 20%.

Figure 6.2: From left to right: Original images, halftoned images, reconstructed images. The top image is lena, middle image is boat and bottom image is hill. Even though the halftoned images (center column) perceptually look relatively close to the original images (left column), they are binary. Reconstructed images (right column) are obtained by restoring the halftoned binary images. Best viewed by zooming on a computer screen.


Figure 6.3: Results obtained by our method on various binary images publicly available on the Internet. No ground truth is available for these images from old computer games, and the algorithm that has generated these images is unknown. Input images are on the left. Restored images are on the right. Best viewed by zooming on a computer screen.

| | Image | FIHT2 | WInHD | LPA-ICI | SA-DCT | Ours |
|---|---|---|---|---|---|---|
| Val. | house | 30.8 | 31.2 | 31.4 | 32.4 | **33.0** |
| | peppers | 25.3 | 26.9 | 27.7 | 28.6 | **29.6** |
| | cameraman | 25.8 | 26.8 | 26.5 | 27.8 | **28.1** |
| | lena | 31.4 | 31.9 | 32.5 | **33.0** | **33.0** |
| Test | barbara | 24.5 | 25.7 | 25.6 | **27.0** | 26.6 |
| | boat | 28.6 | 29.2 | 29.7 | 30.1 | **30.2** |
| | hill | 29.5 | 29.4 | 30.0 | 30.2 | **30.5** |
| | couple | 28.2 | 28.7 | 29.2 | 29.8 | **29.9** |
| | man | 29.3 | 29.4 | 30.1 | 30.3 | **30.4** |
| | fingerprint | 26.0 | 28.1 | 28.3 | 28.5 | **29.0** |
| | bridge | 25.2 | 25.6 | 26.0 | **26.2** | **26.2** |
| | flintstones | 24.7 | 26.4 | 27.2 | 27.6 | **28.0** |

Table 6.2: Quantitative experiments for the inverse halftoning experiments. Results are reported in PSNR (higher is better). SA-DCT refers to (Dabov et al., 2006), LPA-ICI to (Foi et al., 2004), FIHT2 to (Kite et al., 2000) and WInHD to (Neelamani et al., 2009). Best results for each image are in bold.

set of authentic paintings and imitations (Lyu et al., 2004), or involve more sophisticated features (Johnson et al., 2008). Recently, Hugues et al. (2009) have considered a dataset of 8 authentic paintings from Pieter Bruegel the Elder, and 5 imitations.[12] They have proposed to learn dictionaries for sparse coding, and use the kurtosis of the sparse coefficients as discriminative features. We use their dataset, which they kindly provided to us.

The supervised dictionary learning approach we have presented is designed for classifying relatively small signals, and should not be directly applicable to the classification of large images, for which classical computer vision approaches based on bags of words may be better adapted (see Yang et al., 2009; Boureau et al., 2010, for such approaches). However, we show that, for this particular dataset, a simple voting scheme based on the classification of small image patches with our method leads to good results.

The experiment we carry out consists of finding which painting is authentic, and which one is fake, in a pair known to containt one of each.[13] We proceed in a leave-one-out fashion, where we remove for testing one authentic and one imitation paintings from the dataset, and learn on the remaining ones. Since the dataset is small and does not have an official training/test set, we measure a cross-validation score, testing all possible pairs. We consider $12 \times 12$ color image patches, without any pre-processing, and classify each patch from the test images independently. Then, we use a simple majority vote among the test patches to decide which image is the authentic one in the pair test, and

[12] The origin of these paintings is assessed by art historians.

[13] This task is of course considerably easier than classifying each painting as authentic or fake. We do not claim to propose a method that readily applies to forgery detection.

which one is the imitation.[14]

For each pair of authentic/imitation paintings, we build a dataset containing 200 000 patches from the authentic images and 200 000 from the imitations. We use the formulation from Eq. (6.13) for binary classification, and arbitrarily choose dictionaries containing $p = 100$ dictionary elements. Since the training set is large, we set the parameter $\nu$ to 0, also choose the Lasso formulation for decomposing the patches by setting $\lambda_2 = 0$, and cross-validate on the parameter $\lambda_1$, trying values on a grid $\{10^{-4}, 10^{-3}, \ldots, 10^{0}\}$, and then refining the result on a grid with a logarithmic scale of 2. We also compare Eq. (6.13) with the logistic regression loss and the basic formulation of Eq. (6.5) where $\mathbf{D}$ is learned unsupervised.

For classifying individual patches, the cross-validation score of the supervised formulation is a classification rate of $54.04 \pm 2.26\%$, which slightly improves upon the "unsupervised" one that achieves $51.94 \pm 1.92\%$. The task of classifying independently small image patches is difficult since there is significant overlap between the two classes. On the other hand, finding the imitation in a pair of (authentic,imitation) paintings with the voting scheme is easier and the "unsupervised formulation" only fails for one pair, whereas the supervised one has always given the right answer in our experiments.

### 6.5.5 Compressed Sensing

In this experiment, we apply our method to the problem of learning dictionaries and projection matrices for compressed sensing. As explained in Section 6.3.3, our formulation and the conclusions of this section hold for relatively small signals, where the sensing matrix can be stored into memory and learned. Thus, we consider here small image patches of natural images of size $m = 10 \times 10$ pixels. To build our training/validation/test set, we have chosen the Pascal VOC 2006 database of natural images (Everingham et al., 2006): Images 1 to 3000 are used for training; images 3001 to 4000 are used for validation, and the remaining 1304 images are kept for testing. Images are downsampled by a factor 2 so that the JPEG compression artefacts present in this dataset become visually imperceptible, thereby improving its quality for our experiment.

We compare different settings where the task is to reconstruct the patches $\mathbf{y}$ of size $m = 10 \times 10$, from an observation $\mathbf{Zx}$ of size $r \ll m$ (for instance $r = 20$ linear measurements), where $\mathbf{Z}$ in $\mathbb{R}^{r \times m}$ is a sensing matrix. For simplicity reasons, we only consider here the noiseless case, where $\mathbf{y} = \mathbf{x}$. At test time, as explained in Section (6.3.3), our estimate of $\mathbf{y}$ is $\mathbf{W}\boldsymbol{\alpha}^\star(\mathbf{Zx}, \mathbf{D})$, where $\mathbf{D}$ in $\mathbb{R}^{r \times p}$ is a dictionary for representing $\mathbf{Zx}$, and $\mathbf{W}$ in $\mathbb{R}^{m \times p}$ can be interpreted here as a dictionary for representing $\mathbf{y}$. We evaluate several strategies for obtaining $(\mathbf{Z}, \mathbf{D}, \mathbf{W})$:

- We consider the case, which we call RANDOM, where the entries of $\mathbf{Z}$ are i.i.d. samples of the Gaussian distribution $\mathcal{N}(0, 1/\sqrt{m})$. Since the purpose of $\mathbf{Z}$ is to reduce the dimensionality of the input data, it is also natural to consider the case where $\mathbf{Z}$

[14] Note that this experimental setting is different from Hugues et al. (2009), who used only authentic paintings for training (and not imitations). We therefore do not make quantitive comparison with this work.

is obtained by principal component analysis on natural image patches (PCA). Finally, we also learn $\mathbf{Z}$ with the supervised learning formulation of Eq. (6.15), (SL), but consider the case where it is initialized randomly (SL1) or by PCA (SL2).

- The matrix $\mathbf{D}$ can either be fixed or learned. A typical setting would be to set $\mathbf{D} = \mathbf{Z}\mathbf{D}'$, where $\mathbf{D}'$ is discrete-cosine-transform matrix (DCT), often used in signal processing applications (Elad and Aharon, 2006). It can also be learned with an unsupervised learning formulation (UL), or a supervised one (SL).

- $\mathbf{W}$ is always learned in a supervised way.

Since our training set is very large (several millions of patches), we arbitrarily set the regularization parameters $\nu_1$ and $\nu_2$ to 0. We measure reconstruction errors with dictionaries of various levels of overcompleteness by choosing a size $p$ in $\{100, 200, 400\}$. The remaining free parameters are the regularization parameters $\lambda_1$ and $\lambda_2$ for obtaining the coefficients $\boldsymbol{\alpha}^\star$. We try the values $\lambda_1 = 10^i$, with $i$ in $\{-5, \ldots, 0\}$. Unlike what we have done in the experiments of Section 6.5.3, it is absolutely necessary in this setting to use $\lambda_2 > 0$ (according to the theory), since using a zero value for this parameter has led to instabilities and prevented our algorithm from converging. We have tried the values $\lambda_2 = 10^i \lambda_1$, with $i$ in $\{-2, -1, 0\}$. Each learning procedure is performed by our algorithm in one pass on 10 millions of patches randomly extracted from our training images. The pair of parameters $(\lambda_1, \lambda_2)$ that gives the lowest reconstruction error on the validation set is selected, and we report in Table 6.3 the result obtained on a test set of 500 000 patches randomly extracted from the 1304 test images. The conclusions of this compressed sensing experiment on natural image patches are the following:

- *When $\mathbf{Z}$ is initialized as a Gaussian random matrix (case RANDOM), learning $\mathbf{D}$ and $\mathbf{Z}$ significantly improves the reconstruction error (case SL1).* A similar observation was made by Duarte-Carvajalino and Sapiro (2009) with a different formulation.

- *Results obtained with PCA are in general much better than those obtained with random projections*, which is consistent with the conclusions of Weiss et al. (2007).

- *However, PCA does better than SL1. When PCA is used for initializing our supervised formulation, results can be slightly improved (case SL2).* This illustrates either the limits of the non-convex optimization procedure, or that PCA seem to be particularly well adapted to this problem.

- *Learned dictionaries (cases UL and SL) outperform classical DCT dictionaries.*

## 6.6 Conclusion

We have presented in this chapter a general formulation for learning sparse data representations tuned to specific tasks. Unlike classical approaches that learn a dictionary

adapted to the reconstruction of the input data, our method learns features in a supervised way. We have shown that this approach is effective for solving classification and regression tasks in a large-scale setting, allowing the use of millions of training samples, and is able of exploiting successfully unlabeled data, when only a few labeled samples are available. Experiments on handwritten digits classification, non-linear inverse image mapping, digital art authentification, and compressed sensing have shown that our method leads to state-of-the-art results for several real problems. Future work will include adapting our method to various image processing problems such as image deblurring and image super-resolution, and other inverse problems.

| $\mathbf{Z}$ | RANDOM | | | SL1 |
|---|---|---|---|---|
| $\mathbf{D}$ | DCT | UL | SL | SL |
| $r = 5$ | $77.3 \pm 4.0$ | $76.9 \pm 4.0$ | $76.7 \pm 4.0$ | $54.1 \pm 1.3$ |
| $r = 10$ | $57.8 \pm 1.5$ | $56.5 \pm 1.5$ | $55.7 \pm 1.4$ | $36.5 \pm 0.7$ |
| $r = 20$ | $37.1 \pm 1.2$ | $35.4 \pm 1.0$ | $34.5 \pm 0.9$ | $21.4 \pm 0.1$ |
| $r = 40$ | $19.3 \pm 0.8$ | $18.5 \pm 0.7$ | $18.0 \pm 0.6$ | $10.0 \pm 0.3$ |
| $\mathbf{Z}$ | PCA | | | SL2 |
| $\mathbf{D}$ | DCT | UL | SL | SL |
| $r = 5$ | $49.9 \pm 0.0$ | $47.6 \pm 0.0$ | $47.5 \pm 0.1$ | $47.3 \pm 0.3$ |
| $r = 10$ | $33.7 \pm 0.0$ | $32.3 \pm 0.0$ | $32.3 \pm 0.1$ | $31.9 \pm 0.2$ |
| $r = 20$ | $20.4 \pm 0.0$ | $19.7 \pm 0.0$ | $19.6 \pm 0.1$ | $19.4 \pm 0.2$ |
| $r = 40$ | $9.2 \pm 0.0$ | $9.1 \pm 0.0$ | $9.0 \pm 0.0$ | $9.0 \pm 0.0$ |

Table 6.3: Compressed sensing experiment on small natural image patches. The mean squared error (MSE) measured on a test set is reported for each scenario, as well as standard deviations, which are obtained by reproducing 5 times each experiment, randomizing the algorithm initializations and the sampling of the training images. Each patch is normalized to have unit $\ell_2$ norm, and the mean squared reconstruction error is multiplied by 100 for readability purposes (such that a value of 100 corresponds to no reconstruction of the signal). Here, $r$ is the number of rows of the matrix $\mathbf{Z}$ and represents the dimension of the observations $\mathbf{Zx}$. In the setting RANDOM, $\mathbf{Z}$ is a Gaussian random matrix; in PCA, $\mathbf{Z}$ is obtained by principal component analysis on natural image patches; in SL1, $\mathbf{Z}$ is learned with our supervised formulation but initialized with a Gaussian random matrix; in SL2, $\mathbf{Z}$ is learned with the supervised formulation but initialized with PCA. The settings DCT, UL, SL for $\mathbf{D}$ respectively correspond to a discrete-cosine-transform, a dictionary learned in an unsupervised way, and a dictionary learned with our supervised formulation.

# 7

# Conclusion

We have addressed in this thesis several questions related to sparse problems and dictionary learning, borrowing tools from several communities, including machine learning, convex, stochastic and network flow optimization, signal and image processing, computer vision and computer science.

First, we have introduced a new online algorithm for learning dictionaries and solving various matrix factorization problems such as non negative matrix factorization, and sparse principal component analysis. We have proven its convergence and demonstrated experimentally that it is significantly faster than batch alternatives.

In our second contribution, we introduce optimization tools for solving sparse structured problems involving any (overlapping) groups of variables, which shed new light on connections between sparse methods and the literature of network flow optimization. We experimentally demonstrate that this method can be applied to a wide class of learning problems, which have not been addressed efficiently before.

We then present image processing applications combining dictionary learning and the idea that images admit self-similarities. By imposing that similar image patches should admit similar sparsity patterns, we stabilize the decomposition of the patches and improve the quality of recosntruction, leading to state-of-the-art results for image denoising and image demosaicking.

Finally, unlike classical approaches learning dictionaries adapted to the reconstruction of input data, we present a method to learn dictionaries in a supervised way for different prediction tasks. We apply this approach to classification and regression problems, such as digit recognition or nonlinear image inverse problems. We also apply this idea to computer vision tasks, by learning local appearance models of objects, textures and edges in images. We show that the performance of contour-based classifiers can significantly be improved using the category-based edge detector which we have developed based on this principle.

There are also other questions, which we are currently investigating and which have not been presented in this thesis. The first one consists of adapting the task-driven dictionary learning framework of Chapter 6, for image deblurring and super-resolution from a single image. Preliminary results show that it significantly improves upon the generative approach of Yang et al. (2010) and competes favorably for non-blind image deblurring with state-of-the-art methods such as Foi et al. (2006); Dabov et al. (2008);

Guerrero-Colon et al. (2008). Another interesting question is how to extend the formulation for solving blind deblurring problems. This work is primarily done by Florent Couzinié-Devy, with the collaboration of Jean Ponce, Francis Bach and myself. Another line of research is primarily undertaken by Louise Benoit, with the collaboration of the same people listed above, and consists of studying a class of structured dictionaries that are generated based on the principle of the *epitome* of Jojic et al. (2003), extending the work on *image-signature* dictionaries of Aharon and Elad (2008). Open questions include incorporating spatial consistency in the patches decompositions of images, better formalize this framework and find fast algorithms that are adapted to it.

There are machine learning applications and extensions of the methods we have introduced that might be interesting to consider. In Chapter 3, we show preliminary but promising results when using structured sparsity for a background subtraction application. More precisely, this is achieved by representing new frames from a video stream containing foreground objects as a linear combination of background images plus an error term. To the best of our knowledge, this idea was introduced by Wright et al. (2009a) in the context of robust face recognition, where a partly occluded test image was represented in such a way, with the hope that the error term will be able to capture the occluded part. To achieve robustness, the latter authors have proposed to regularize the error term with the $\ell_1$-norm, whereas we have proposed to use a structured sparse regularization norm to encode spatial consistency among neighboring pixels. The idea of penalizing an error term by the $\ell_1$-norm has then been used by Candès et al. (2010) in a robust PCA formulation. Using the structured sparsity-inducing norm instead, and the algorithms we have proposed to solve structured sparse problems, should allow a natural extension of this framework to a robust structured PCA formulation. More generally, the same idea could be used in the context of matrix factorization.

Moreover, the use of structured sparsity to enforce spatial consistency in the background subtraction task suggests that such regularization norms could be useful in other computer vision applications such as image segmentation. Interestingly, these tasks are often addressed using min-cut/max-flow (or graph-cut) algorithms (Boykov et al., 2001), which makes an interesting connection with the optimization method we have introduced in Chapter 3.

During this thesis, we have tried to make bridges between several fields, and we believe that there are other connections to make with sparse problems. For instance, to the best of our knowledge, sparse methods are not yet able to correctly handle binary or quantized data. To address this issue, we are interested in discrete optimization and coding theory, where we hope to find tools to propose more general formulations.

Another important direction we consider is to apply our tools to experimental sciences, where we wish our methods eventually to be useful. This would go through collaborations with scientists from bio-informatics, neuroscience, which are two fields where matrix factorization techniques and sparse methods have already been applied, but also possibly physicists, who could provide us real-life problems.

# A

# Theorems and Useful Lemmas

We provide in this section a few theorems and lemmas from the optimization and probability literature, which are used in this thesis.

**Theorem 2** (**Corollary of Theorem 4.1 from Bonnans and Shapiro (1998), due to Danskin (1967).**)
*Let $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$. Suppose that for all $\mathbf{x} \in \mathbb{R}^p$ the function $f(\mathbf{x}, .)$ is differentiable, and that $f$ and $\nabla_{\mathbf{u}} f(\mathbf{x}, \mathbf{u})$ the derivative of $f(\mathbf{x}, .)$ are continuous on $\mathbb{R}^p \times \mathbb{R}^q$. Let $v(\mathbf{u})$ be the optimal value function $v(\mathbf{u}) = \min_{\mathbf{x} \in C} f(\mathbf{x}, \mathbf{u})$, where $C$ is a compact subset of $\mathbb{R}^p$. Then $v(\mathbf{u})$ is directionally differentiable. Furthermore, if for $\mathbf{u}^0 \in \mathbb{R}^q$, $f(., \mathbf{u}^0)$ has a unique minimizer $\mathbf{x}^0$ then $v(\mathbf{u})$ is differentiable in $\mathbf{u}^0$ and $\nabla_{\mathbf{u}} v(\mathbf{u}^0) = \nabla_{\mathbf{u}} f(\mathbf{x}^0, \mathbf{u}^0)$.*

**Theorem 3** (**Sufficient condition of convergence for a stochastic process, see Bottou (1998) and references therein (Métivier, 1983; Fisk, 1965).**)
*Let $(\Omega, \mathcal{F}, P)$ be a measurable probability space, $u_t$, for $t \geq 0$, be the realization of a stochastic process and $\mathcal{F}_t$ be the filtration determined by the past information at time $t$. Let*

$$\delta_t = \begin{cases} 1 & \text{if } \mathbb{E}[u_{t+1} - u_t | \mathcal{F}_t] > 0, \\ 0 & \text{otherwise.} \end{cases}$$

*If for all $t$, $u_t \geq 0$ and $\sum_{t=1}^{\infty} \mathbb{E}[\delta_t (u_{t+1} - u_t)] < \infty$, then $u_t$ is a quasi-martingale and converges almost surely. Moreover,*

$$\sum_{t=1}^{\infty} |\mathbb{E}[u_{t+1} - u_t | \mathcal{F}_t]| < +\infty \quad a.s.$$

**Lemma 13** (**A corollary of Donsker theorem (see chap. 19.2, lemma 19.36 and example 19.7, Van der Vaart, 1998)**
*Let $F = \{f_\theta : \chi \to \mathbb{R}, \theta \in \Theta\}$ be a set of measurable functions indexed by a bounded subset $\Theta$ of $\mathbb{R}^d$. Suppose that there exists a constant $K$ such that*

$$|f_{\theta_1}(x) - f_{\theta_2}(x)| \leq K \|\theta_1 - \theta_2\|_2,$$

*for every $\theta_1$ and $\theta_2$ in $\Theta$ and $x$ in $\chi$. Then, $F$ is P-Donsker (see Van der Vaart, 1998, chap. 19.2). For any $f$ in $F$, Let us define $\mathbb{P}_n f$, $\mathbb{P} f$ and $\mathbb{G}_n f$ as*

$$\mathbb{P}_n f = \frac{1}{n} \sum_{i=1}^{n} f(X_i), \qquad \mathbb{P} f = \mathbb{E}_X[f(X)], \qquad \mathbb{G}_n f = \sqrt{n}(\mathbb{P}_n f - \mathbb{P} f).$$

*Let us also suppose that for all* $f$, $\mathbb{P}f^2 < \delta^2$ *and* $\|f\|_\infty < M$ *and that the random elements* $X_1, X_2, \ldots$ *are Borel-measurable. Then, we have*

$$\mathbb{E}_P \|\mathbb{G}_n\|_F = O(1),$$

*where* $\|\mathbb{G}_n\|_F = \sup_{f \in F} |\mathbb{G}_n f|$. *For a more general variant of this lemma and additional explanations and examples, see Van der Vaart (1998).*

**Lemma 14** (**A simple lemma on positive converging sums.**)
*Let* $a_n$, $b_n$ *be two real sequences such that for all* $n$, $a_n \geq 0, b_n \geq 0$, $\sum_{n=1}^{\infty} a_n = \infty$, $\sum_{n=1}^{\infty} a_n b_n < \infty$, $\exists K > 0$ *s.t.* $|b_{n+1} - b_n| < K a_n$. *Then,* $\lim_{n \to +\infty} b_n = 0$.

*Proof.* The proof is similar to Bertsekas (1999, prop 1.2.4). □

# B

# Proofs

## B.1 Proofs of Lemmas

### B.1.1 Proof of Lemma 2

*Proof.* This proof is inspired by Prop 4.32 of Bonnans and Shapiro (2000) on the Lipschitz regularity of solutions of optimization problems. Using assumption **(B)**, for all $t$, the surrogate $\hat{f}_t$ is strictly convex with a Hessian lower-bounded by $\kappa_1$. Then, a short calculation shows that it verifies the *second-order growth condition*

$$\hat{f}_t(\mathbf{D}^{t+1}) - \hat{f}_t(\mathbf{D}^t) \geq \kappa_1 \|\mathbf{D}^{t+1} - \mathbf{D}^t\|_{\mathrm{F}}^2. \tag{B.1}$$

Moreover,

$$\begin{aligned}\hat{f}_t(\mathbf{D}^{t+1}) - \hat{f}_t(\mathbf{D}^t) &= \hat{f}_t(\mathbf{D}^{t+1}) - \hat{f}_{t+1}(\mathbf{D}^{t+1}) + \hat{f}_{t+1}(\mathbf{D}^{t+1}) - \hat{f}_{t+1}(\mathbf{D}^t) + \hat{f}_{t+1}(\mathbf{D}^t) - \hat{f}_t(\mathbf{D}^t) \\ &\leq \hat{f}_t(\mathbf{D}^{t+1}) - \hat{f}_{t+1}(\mathbf{D}^{t+1}) + \hat{f}_{t+1}(\mathbf{D}^t) - \hat{f}_t(\mathbf{D}^t),\end{aligned}$$

where we have used that $\hat{f}_{t+1}(\mathbf{D}^{t+1}) - \hat{f}_{t+1}(\mathbf{D}^t) \leq 0$ because $\mathbf{D}^{t+1}$ minimizes $\hat{f}_{t+1}$ on $\mathcal{D}$. Since $\hat{f}_t(\mathbf{D}) = \frac{1}{t}(\frac{1}{2}\operatorname{Tr}(\mathbf{D}^\top \mathbf{D}\mathbf{B}^t) - \operatorname{Tr}(\mathbf{D}^\top \mathbf{C}^t))$, and $\|\mathbf{D}\|_{\mathrm{F}} \leq \sqrt{p}$, it is possible to show that $\hat{f}_t - \hat{f}_{t+1}$ is Lipschitz with constant $c_t = \frac{1}{t}(\|\mathbf{C}^{t+1} - \mathbf{C}^t\|_{\mathrm{F}} + \sqrt{p}\|\mathbf{B}^{t+1} - \mathbf{B}^t\|_{\mathrm{F}})$, which gives

$$\hat{f}_t(\mathbf{D}^{t+1}) - \hat{f}_t(\mathbf{D}^t) \leq c_t \|\mathbf{D}^{t+1} - \mathbf{D}^t\|_{\mathrm{F}}. \tag{B.2}$$

From Eq. (B.1) and (B.2), we obtain

$$\|\mathbf{D}^{t+1} - \mathbf{D}^t\|_{\mathrm{F}} \leq \frac{c_t}{\kappa_1}.$$

Assumptions **(A)**, **(C)** and Eq. (2.9) ensure that the vectors $\boldsymbol{\alpha}^i$ and $\mathbf{x}^i$ are bounded with probability one and therefore $c_t = O(1/t)$ a.s. □

### B.1.2 Proof of Lemma 3

*Proof.* The proof relies on tools from conic duality (Boyd and Vandenberghe, 2004). We can equivalently rewrite problem (3.3) as

$$\min_{\mathbf{v}\in\mathbb{R}^p, \mathbf{z}\in\mathbb{R}^{|\mathcal{G}|}} \frac{1}{2}\|\mathbf{u} - \mathbf{v}\|_2^2 + \lambda \sum_{g\in\mathcal{G}} \eta_g z_g, \text{ such that } \|\mathbf{v}_{|g}\| \leq z_g, \ \forall g \in \mathcal{G},$$

by introducing the primal variables $\mathbf{z} = (z_g)_{g\in\mathcal{G}} \in \mathbb{R}^{|\mathcal{G}|}$, with the additional $|\mathcal{G}|$ conic constraints $\|\mathbf{v}_{|g}\| \leq z_g,\ g \in \mathcal{G}$.

This primal problem is convex and satisfies Slater's conditions for generalized conic inequalities (i.e., existence of a feasible point in the interior of the domain), which implies that strong duality holds (Boyd and Vandenberghe, 2004). We now consider the Lagrangian $\mathcal{L}$ defined as

$$\mathcal{L}(\mathbf{v}, \mathbf{z}, \boldsymbol{\tau}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{u} - \mathbf{v}\|_2^2 + \lambda \sum_{g\in\mathcal{G}} \eta_g z_g - \sum_{g\in\mathcal{G}} \begin{pmatrix} z_g \\ \mathbf{v}_{|g} \end{pmatrix}^\top \begin{pmatrix} \tau_g \\ \boldsymbol{\xi}^g \end{pmatrix},$$

with the dual variables $\boldsymbol{\tau} = (\tau_g)_{g\in\mathcal{G}}$ in $\mathbb{R}^{|\mathcal{G}|}$, and $\boldsymbol{\xi} = (\boldsymbol{\xi}^g)_{g\in\mathcal{G}}$ in $\mathcal{R}^{p\times|\mathcal{G}|}$, such that for all $g \in \mathcal{G}$, $\boldsymbol{\xi}^g_j = 0$ if $j \notin g$ and $\|\boldsymbol{\xi}^g\|_* \leq \tau_g$.

The dual function is obtained by taking the derivatives of $\mathcal{L}$ with respect to the primal variables $\mathbf{v}$ and $\mathbf{z}$ and equating them to zero, which leads to

$$\begin{aligned} & \mathbf{v} - \mathbf{u} - \textstyle\sum_{g\in\mathcal{G}} \boldsymbol{\xi}^g && = 0, \\ \forall g \in \mathcal{G}, \qquad & \lambda\eta_g - \tau_g && = 0. \end{aligned}$$

After simplifying the Lagrangian and flipping the sign of $\boldsymbol{\xi}$, we obtain the dual problem in Eq. (3.6). As far as the optimality conditions are concerned, they are derived from the Karush–Kuhn–Tucker conditions for generalized conic inequalities (Boyd and Vandenberghe, 2004). We have that $\{\mathbf{v}, \mathbf{z}, \boldsymbol{\tau}, \boldsymbol{\xi}\}$ are optimal if and only if

$$\begin{aligned} & \mathbf{v} - \mathbf{u} + \textstyle\sum_{g\in\mathcal{G}} \boldsymbol{\xi}^g && = 0, && \\ \forall g \in \mathcal{G}, \qquad & \lambda\eta_g - \tau_g && = 0, && \\ \forall g \in \mathcal{G}, \qquad & z_g\tau_g - \mathbf{v}_{|g}^\top \boldsymbol{\xi}^g && = 0, && \text{(Complementary slackness)} \\ \forall g \in \mathcal{G}, \qquad & \|\mathbf{v}_{|g}\| \leq z_g, && && \\ \forall g \in \mathcal{G}, \qquad & \|\boldsymbol{\xi}^g\|_* \leq \tau_g. && && \end{aligned}$$

Combining the complementary slackness with the definition of the dual norm, we have

$$\forall g \in \mathcal{G},\ z_g\tau_g = \mathbf{v}_{|g}^\top \boldsymbol{\xi}^g \leq \|\mathbf{v}_{|g}\|\|\boldsymbol{\xi}^g\|_*.$$

Furthermore, using the fact that $\forall g \in \mathcal{G},\ \|\mathbf{v}_{|g}\| \leq z_g$ and $\|\boldsymbol{\xi}^g\|_* \leq \tau_g = \lambda\eta_g$, we obtain the following chain of inequalities

$$\forall g \in \mathcal{G},\ \lambda z_g\eta_g = \mathbf{v}_{|g}^\top \boldsymbol{\xi}^g \leq \|\mathbf{v}_{|g}\|\|\boldsymbol{\xi}^g\|_* \leq z_g\|\boldsymbol{\xi}^g\|_* \leq \lambda z_g\eta_g,$$

for which equality must hold. We notably have

$$\begin{cases} \mathbf{v}_{|g}^\top \boldsymbol{\xi}^g = \|\mathbf{v}_{|g}\|\|\boldsymbol{\xi}^g\|_*, \\ z_g\|\boldsymbol{\xi}^g\|_* = \lambda z_g\eta_g. \end{cases}$$

If $\mathbf{v}_{|g} \neq 0$, then $z_g$ cannot be equal to zero, which implies in turn that $\|\boldsymbol{\xi}^g\|_* = \lambda\eta_g$.

Reciprocally, starting from the optimality conditions of Lemma 3, we can derive the Karush–Kuhn–Tucker conditions displayed above. More precisely, we define for all $g \in \mathcal{G}$,

$$\tau_g \triangleq \lambda \eta_g \quad \text{and} \quad z_g \triangleq \|\mathbf{v}_{|g}\|.$$

The only condition that needs to be discussed is the complementary slackness. If $\mathbf{v}_{|g} = 0$, then it is easily satisfied. Otherwise, combining the definitions of $\tau_g$, $z_g$ and the fact that

$$\mathbf{v}_{|g}^\top \boldsymbol{\xi}^g = \|\mathbf{v}_{|g}\| \|\boldsymbol{\xi}^g\|_* \text{ and } \|\boldsymbol{\xi}^g\|_* = \lambda \eta_g,$$

we end up with the desired complementary slackness. □

### B.1.3 Proof of Lemma 4

*Proof.* When the vector $\mathbf{v}$ is already in the ball of $\|.\|_*$ with radius $t$, i.e., $\|\mathbf{v}\|_* \leq t$, the situation is simple, since the projection $\Pi_t^*(\mathbf{v})$ obviously gives $\mathbf{v}$ itself. On the other hand, a necessary and sufficient optimality condition for having

$$\boldsymbol{\kappa} = \Pi_t^*(\mathbf{v}) = \operatorname*{arg\,min}_{\|\mathbf{y}\|_* \leq t} \|\mathbf{v} - \mathbf{y}\|_2$$

is that the residual $\mathbf{v} - \boldsymbol{\kappa}$ lies in the normal cone of the constraint set (Borwein and Lewis, 2006), that is, for all $\mathbf{y}$ such that $\|\mathbf{y}\|_* \leq t$, $(\mathbf{v} - \boldsymbol{\kappa})^\top (\mathbf{y} - \boldsymbol{\kappa}) \leq 0$. The displayed result then follows from the definition of the dual norm, namely $\|\boldsymbol{\kappa}\|_* = \max_{\|\mathbf{z}\| \leq 1} \mathbf{z}^\top \boldsymbol{\kappa}$. □

### B.1.4 Proof of Lemma 5

*Proof.* The proof mostly relies on the optimality condition derived in Lemma 4. We thus have to prove that either

$$\boldsymbol{\kappa}^g = \mathbf{v}_{|g} - \boldsymbol{\kappa}^h_{|g}, \text{ if } \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^h_{|g}\|_* \leq t_g,$$

or

$$\|\boldsymbol{\kappa}^g\|_* = t_g \quad \text{and} \quad \boldsymbol{\kappa}^{g\top}(\mathbf{v}_{|g} - \boldsymbol{\kappa}^h_{|g} - \boldsymbol{\kappa}^g) = \|\boldsymbol{\kappa}^g\|_* \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^h_{|g} - \boldsymbol{\kappa}^g\|.$$

Note that the feasibility of $\boldsymbol{\kappa}^g$, i.e., $\|\boldsymbol{\kappa}^g\|_* \leq t_g$, is one of the hypothesis in the Lemma.

Let us first assume that $\|\boldsymbol{\kappa}^g\|_* < t_g$. We necessarily have that $\mathbf{v}_{|g}$ also lies in the interior of the ball of $\|.\|_*$ with radius $t_g$, and it holds that $\boldsymbol{\kappa}^g = \mathbf{v}_{|g}$. Since $g \subseteq h$, we have that the vector $\mathbf{v}_{|h} - \boldsymbol{\kappa}^g = \mathbf{v}_{|h} - \mathbf{v}_{|g}$ has only zero entries on $g$. As a result, $\boldsymbol{\kappa}^h_g = 0$ and we obtain

$$\boldsymbol{\kappa}^g = \mathbf{v}_{|g} = \mathbf{v}_{|g} - \boldsymbol{\kappa}^h_{|g},$$

which is the desired conclusion. From now on, we assume that $\|\boldsymbol{\kappa}^g\|_* = t_g$. It then remains to show that

$$\boldsymbol{\kappa}^{g\top}(\mathbf{v}_{|g} - \boldsymbol{\kappa}^h_{|g} - \boldsymbol{\kappa}^g) = \|\boldsymbol{\kappa}^g\|_* \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^h_{|g} - \boldsymbol{\kappa}^g\|.$$

We now distinguish the proof, depending on the used norm.

$\ell_2$ **norm:** in the particular case of the $\ell_2$ norm, the optimality condition for the projection amounts to check when equality holds in the Cauchy-Schwartz inequality, i.e., when the vectors have same signs and are linearly dependent. Thus, there exists $\rho_g, \rho_h > 0$ such that $\rho_g \boldsymbol{\kappa}^g = \mathbf{v}_{|g} - \boldsymbol{\kappa}^g$ and $\rho_h \boldsymbol{\kappa}^h = \mathbf{v}_{|h} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h$.

Note that the case $\rho_h = 0$ leads to $\mathbf{v}_{|h} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h = 0$, and therefore $\mathbf{v}_{|g} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h_{|g} = 0$ since $g \subseteq h$, which directly gives the result. The case $\rho_g = 0$ implies $\mathbf{v}_{|g} - \boldsymbol{\kappa}^g = 0$ and therefore $\boldsymbol{\kappa}^h_{|g} = 0$, giving the result as well. We can therefore assume now $\rho_h > 0$ and $\rho_g > 0$. Some algebra leads to

$$\boldsymbol{\kappa}^g = \frac{\rho_h + 1}{\rho_g \rho_h}(\mathbf{v}_{|g} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h_{|g}),$$

and consequently

$$\boldsymbol{\kappa}^{g\top}(\mathbf{v}_{|g} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h_{|g}) = \|\boldsymbol{\kappa}^g\|_2 \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h_{|g}\|_2.$$

$\ell_\infty$ **norm:** here, the optimality condition comes down to analyzing when equality holds in the $\ell_\infty$-$\ell_1$ Hölder inequality. Specifically, $\boldsymbol{\kappa}^g = \Pi^*_{t_g}(\mathbf{v}_{|g})$ holds if and only if for all $\boldsymbol{\kappa}^g_j \neq 0, j \in g$, we have

$$\mathbf{v}_j - \boldsymbol{\kappa}^g_j = \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^g\|_\infty \operatorname{sign}(\boldsymbol{\kappa}^g_j).$$

Looking at the same condition for $\boldsymbol{\kappa}^h$, we have that $\boldsymbol{\kappa}^h = \Pi^*_{t_h}(\mathbf{v}_{|h} - \boldsymbol{\kappa}_g)$ holds if and only if for all $\boldsymbol{\kappa}^h_j \neq 0, j \in h$, we have

$$\mathbf{v}_j - \boldsymbol{\kappa}^g_j - \boldsymbol{\kappa}^h_j = \|\mathbf{v}_{|h} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h\|_\infty \operatorname{sign}(\boldsymbol{\kappa}^h_j).$$

From those relationships we notably deduce that for all $j \in g$ such that $\boldsymbol{\kappa}^g_j \neq 0$, $\operatorname{sign}(\boldsymbol{\kappa}^g_j) = \operatorname{sign}(\mathbf{v}_j) = \operatorname{sign}(\boldsymbol{\kappa}^h_j) = \operatorname{sign}(\mathbf{v}_j - \boldsymbol{\kappa}^g_j) = \operatorname{sign}(\mathbf{v}_j - \boldsymbol{\kappa}^g_j - \boldsymbol{\kappa}^h_j)$. Let $j \in g$ such that $\boldsymbol{\kappa}^g_j \neq 0$. At this point, using the equalities we have just presented,

$$|\mathbf{v}_j - \boldsymbol{\kappa}^g_j - \boldsymbol{\kappa}^h_j| = \begin{cases} \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^g\|_\infty & \text{if } \boldsymbol{\kappa}^h_j = 0 \\ \|\mathbf{v}_{|h} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h\|_\infty & \text{if } \boldsymbol{\kappa}^h_j \neq 0 \end{cases}$$

Since $\|\mathbf{v}_{|g} - \boldsymbol{\kappa}^g\|_\infty \geq \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h_{|g}\|_\infty$ (which can be shown using the sign equalities above), and $\|\mathbf{v}_{|h} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h\|_\infty \geq \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h_{|g}\|_\infty$ (since $g \subseteq h$), we have

$$\|\mathbf{v}_{|g} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h_{|g}\|_\infty \geq |\mathbf{v}_j - \boldsymbol{\kappa}^g_j - \boldsymbol{\kappa}^h_j| \geq \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h_{|g}\|_\infty$$

and therefore for all $\boldsymbol{\kappa}^g_j \neq 0$, $j \in g$,

$$\mathbf{v}_j - \boldsymbol{\kappa}^g_j - \boldsymbol{\kappa}^h_j = \|\mathbf{v}_{|g} - \boldsymbol{\kappa}^g - \boldsymbol{\kappa}^h_{|g}\|_\infty \operatorname{sign}(\boldsymbol{\kappa}^g_j),$$

which gives the result. □

### B.1.5 Proof of Lemma 6

*Proof.* One notices first that the procedure `computeNorm` is called one time for each group $g$ in $\mathcal{G}$, computing a set of scalars $(\rho_g)_{g\in\mathcal{G}}$ in an order which is compatible with the convergence in one pass of Algorithm 3—that is, the children of a node are processed prior to the node itself. Following such an order, the update of the group $g$ in the original Algorithm 3 computes the variable $\boldsymbol{\xi}^g$ which updates implicitly the primal variable as follows

$$\mathbf{v}_{|g} \leftarrow \max\left(0, 1 - \frac{\lambda\eta_g}{\|\mathbf{v}_{|g}\|_2}\right)\mathbf{v}_{|g}.$$

It is now possible to show by induction that for all group $g$ in $\mathcal{G}$, after a call to the procedure `computeNorm`$(g)$, the auxiliary variable $\eta_g$ takes the value $\|\mathbf{v}_{|g}\|_2^2$ where $\mathbf{v}$ has the same value as during the iteration $g$ of Algorithm 3. Therefore, after calling the procedure `computeNorm`$(g_0)$, where $g_0$ is the root of the tree, the values $\rho_g$ correspond to the successive scaling factors of the variable $\mathbf{v}_{|g}$ obtained during the execution of Algorithm 3. After having computed all the scaling factors $\rho_g$, $g \in \mathcal{G}$, the procedure `recursiveScaling` ensures that each variable $j$ in $\{1, \ldots, p\}$ is scaled by the product of all the $\rho_h$, where $h$ is an ancestor of the variable $j$.

The complexity of the algorithm is easy to characterize: Each procedure `computeNorm` and `recursiveScaling` is called $p$ times, each call for a group $g$ has a constant number of operations plus as many operations as the number of children of $p$. Since each children can be called at most one time, the total number of operation of the algorithm is $O(p)$. □

### B.1.6 Proof of Lemma 7

*Proof.* Again, we notice that the order of the projections in Algorithm 5 is compatible with a convergence in one pass of Algorithm 3 with $\|.\| = \ell_\infty$. Then, it is easy to show that by computing a variable $\boldsymbol{\xi}^g = \Pi^*_{\lambda\eta_g}(\mathbf{v}_{|g})$, Algorithm 3 implicitly updates the primal variable with the formula as in Algorithm 5. Both algorithm have therefore the same output primal variable $\mathbf{v}$.

To analyze the complexity of the procedure, one notice first that a projection on the $\ell_1$-ball can be done in linear time (Brucker, 1984; Maculan and de Paula, 1989). The algorithm performs $g$ projection, each of them involving $|g|$ variables. By noticing that if $g$ and $h$ are two groups with the same depth in the tree, then $g \cap h = \emptyset$, it is easy to show that the number of variables involved in all the projections is less than or equal to $dp$, where $d$ is the depth of the tree. Since the projections are linear in the number of variables, the total complexity is therefore $O(dp)$. □

### B.1.7 Proof of Lemma 9

*Proof.* We first notice that on both $G$ and $G'$, the cost of a flow on the graph only depends on the flow on the arcs $(j, t)$, $j$ in $V_u$, which we have denoted by $\bar{\boldsymbol{\xi}}$ in $E$.

We will prove that finding a feasible flow $\pi$ on $G$ with a cost $c(\pi)$ is equivalent to finding a feasible flow $\pi'$ on $G'$ with the same cost $c(\pi) = c(\pi')$. We now use the concept of *path flow*, which is a flow vector in $G$ carrying the same positive value on every arc of a directed path between two nodes of $G$. It intuitively corresponds to sending a positive amount of flow along a path of the graph.

According to the definition of graph equivalence introduced in the Lemma, it is easy to show that there is a bijection between the arcs in $E$, and the paths in $E'$ with positive capacities on every arc. Given now a feasible flow $\pi$ in $G$, we build a feasible flow $\pi'$ on $G'$ which is a *sum* of path flows. More precisely, for every arc $a$ in $E$, we consider its equivalent path in $E'$, with a path flow carrying the same amount of flow as $a$. Therefore, each arc $a'$ in $E'$ has a total amount of flow that is equal to the sum of the flows carried by the path flows going over $a'$. It is also easy to show that this construction builds a flow on $G'$ (capacity and conservation constraints are satisfied) and that this flow $\pi'$ has the same cost as $\pi$, that is, $c(\pi) = c(\pi')$.

Conversely, given a flow $\pi'$ on $G'$, we use a classical path flow decomposition (see Proposition 1.1 in (Bertsekas, 1991)), saying that there exists a decomposition of $\pi'$ as a sum of path flows in $E'$. Using the bijection described above, we know that each path in the previous sums corresponds to a unique arc in $E$. We now build a flow $\pi$ in $G$, by associating with each path flow in the decomposition of $\pi'$, an arc in $E$ carrying the same amount of flow. The flow of every other arc in $E$ is set to zero. It is also easy to show that this builds a valid flow in $G$ that has the same cost as $\pi'$. □

### B.1.8 Proof of Lemma 11

*Proof.* By definition of $\Omega^*(\boldsymbol{\kappa})$, we have

$$\Omega^*(\boldsymbol{\kappa}) \triangleq \max_{\Omega(\mathbf{z})\leq 1} \mathbf{z}^\top \boldsymbol{\kappa}.$$

By introducing the primal variables $(\alpha_g)_{g\in\mathcal{G}} \in \mathbb{R}^{|\mathcal{G}|}$, we can rewrite the previous maximization problem as

$$\Omega^*(\boldsymbol{\kappa}) = \max_{\sum_{g\in\mathcal{G}} \eta_g \alpha_g \leq 1} \boldsymbol{\kappa}^\top \mathbf{z}, \quad \text{s.t.} \quad \forall\, g \in \mathcal{G},\ \|\mathbf{z}_g\|_\infty \leq \alpha_g,$$

with the additional $|\mathcal{G}|$ conic constraints $\|\mathbf{z}_g\|_\infty \leq \alpha_g$. This primal problem is convex and satisfies Slater's conditions for generalized conic inequalities, which implies that strong duality holds (Boyd and Vandenberghe, 2004). We now consider the Lagrangian $\mathcal{L}$ defined as

$$\mathcal{L}(\mathbf{z}, \alpha_g, \tau, \gamma_g, \boldsymbol{\xi}) = \boldsymbol{\kappa}^\top \mathbf{z} + \tau(1 - \sum_{g\in\mathcal{G}} \eta_g \alpha_g) + \sum_{g\in\mathcal{G}} \begin{pmatrix} \alpha_g \\ \mathbf{z}_g \end{pmatrix}^\top \begin{pmatrix} \gamma_g \\ \boldsymbol{\xi}_g^g \end{pmatrix},$$

with the dual variables $\{\tau, (\gamma_g)_{g\in\mathcal{G}}, \boldsymbol{\xi}\} \in \mathbb{R}_+ \times \mathbb{R}^{|\mathcal{G}|} \times \mathbb{R}^{p\times|\mathcal{G}|}$ such that for all $g \in \mathcal{G}$, $\boldsymbol{\xi}_j^g = 0$ if $j \notin g$ and $\|\boldsymbol{\xi}^g\|_1 \leq \gamma_g$. The dual function is obtained by taking the derivatives

of $\mathcal{L}$ with respect to the primal variables $\mathbf{z}$ and $(\alpha_g)_{g\in\mathcal{G}}$ and equating them to zero, which leads to

$$\begin{aligned} \forall j \in \{1,\dots,p\}, \quad \boldsymbol{\kappa}_j + \textstyle\sum_{g\in\mathcal{G}} \boldsymbol{\xi}_j^g &= 0 \\ \forall g \in \mathcal{G}, \qquad \tau\eta_g - \gamma_g &= 0. \end{aligned}$$

After simplifying the Lagrangian and flipping the sign of $\boldsymbol{\xi}$, the dual problem then reduces to

$$\min_{\boldsymbol{\xi}\in\mathbb{R}^{p\times|\mathcal{G}|},\tau} \tau \quad \text{s.t.} \quad \begin{cases} \forall j \in \{1,\dots,p\}, \boldsymbol{\kappa}_j = \textstyle\sum_{g\in\mathcal{G}} \boldsymbol{\xi}_j^g \text{ and } \boldsymbol{\xi}_j^g = 0 \text{ if } j \notin g, \\ \forall g \in \mathcal{G}, \|\boldsymbol{\xi}^g\|_1 \leq \tau\eta_g, \end{cases}$$

which is the desired result. □

## B.2 Proofs of Propositions

### B.2.1 Proof of Proposition 4

*Proof.* Assumption **(A)** ensures that the vectors $\boldsymbol{\alpha}^\star$ are bounded for $\mathbf{x}$ in $K$ and $\mathbf{D}$ in $\mathcal{D}$. Therefore, one can restrict the optimization problem (2.10) to a compact subset of $\mathbb{R}^p$. Under assumption **(C)**, the solution of Eq. (2.10) is unique and $\boldsymbol{\alpha}^\star$ is well-defined. Theorem 2 in Appendix A from Bonnans and Shapiro (1998) can be applied and gives us directly the first statement. Since $K$ is compact, and $\ell$ is continuously differentiable, the second statement follows immediately.

To prove the third claim, we will show that for all $\mathbf{x}$ in $K$, $\boldsymbol{\alpha}^\star(\mathbf{x},.)$ is Lipschitz with a constant independent of $\mathbf{x}$,[1] which is a sufficient condition for $\nabla f$ to be Lipschitz. First, the function optimized in Eq. (2.10) is continuous in $\boldsymbol{\alpha}$, $\mathbf{D}$, $\mathbf{x}$ and has a unique minimum, implying that $\boldsymbol{\alpha}^\star$ is continuous in $\mathbf{x}$ and $\mathbf{D}$.

Consider a matrix $\mathbf{D}$ in $\mathcal{D}$ and $\mathbf{x}$ in $K$ and denote by $\boldsymbol{\alpha}^\star$ the vector $\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})$, and again by $\Lambda$ the set of indices $j$ such that $|\mathbf{d}^{j\top}(\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}^\star)| = \lambda$. Since $\mathbf{d}^{j\top}(\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}^\star)$ is continuous in $\mathbf{D}$ and $\mathbf{x}$, there exists an open neighborhood $V$ around $(\mathbf{x},\mathbf{D})$ such that for all $(\mathbf{x}',\mathbf{D}')$ in $V$, and $j \notin \Lambda$, $|\mathbf{d}^{j\top\prime}(\mathbf{x}'-\mathbf{D}'\boldsymbol{\alpha}^{\star\prime})| < \lambda$ and $\boldsymbol{\alpha}_j^{\star\prime} = 0$, where $\boldsymbol{\alpha}^{\star\prime} = \boldsymbol{\alpha}^\star(\mathbf{x}',\mathbf{D}')$.

Denoting by $\mathbf{U}_\Lambda$ the matrix composed of the columns of a matrix $\mathbf{U}$ corresponding to the index set $\Lambda$ and similarly by $\mathbf{u}_\Lambda$ the vector composed of the values of a vector $\mathbf{u}$ corresponding to $\Lambda$, we consider the function $\tilde{\ell}$

$$\tilde{\ell}(\mathbf{x},\mathbf{D}_\Lambda,\boldsymbol{\alpha}_\Lambda) \triangleq \frac{1}{2}\|\mathbf{x}-\mathbf{D}_\Lambda\boldsymbol{\alpha}_\Lambda\|_2^2 + \lambda\|\boldsymbol{\alpha}_\Lambda\|_1,$$

Assumption **(C)** tells us that $\tilde{\ell}(\mathbf{x},\mathbf{D}_\Lambda,.)$ is strictly convex with a Hessian lower-bounded by $\kappa_2$. Let us consider $(\mathbf{x}',\mathbf{D}')$ in $V$. A simple calculation shows that

$$\tilde{\ell}(\mathbf{x},\mathbf{D}_\Lambda,\boldsymbol{\alpha}_\Lambda^{\star\prime}) - \tilde{\ell}(\mathbf{x},\mathbf{D}_\Lambda,\boldsymbol{\alpha}_\Lambda^\star) \geq \kappa_2\|\boldsymbol{\alpha}_\Lambda^{\star\prime} - \boldsymbol{\alpha}_\Lambda^\star\|_2^2.$$

[1] From now on, for a vector $\mathbf{x}$ in $\mathbb{R}^m$, $\boldsymbol{\alpha}^\star(\mathbf{x},.)$ denotes the function that associates with a matrix $\mathbf{D}$ verifying Assumption **(C)**, the optimal solution $\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})$. For simplicity, we will use these slightly abusive notation in the rest of the work.

Moreover, it is easy to show that $\tilde{\ell}(\mathbf{x}, \mathbf{D}_\Lambda, .) - \tilde{\ell}(\mathbf{x}', \mathbf{D}'_\Lambda, .)$ is Lipschitz with constant $e_1\|\mathbf{D}_\Lambda - \mathbf{D}'_\Lambda\|_{\mathrm{F}} + e_2\|\mathbf{x} - \mathbf{x}'\|_2$, where $e_1, e_2$ are constants independent of $\mathbf{D}, \mathbf{D}', \mathbf{x}, \mathbf{x}'$ and then, one can show that

$$\|\boldsymbol{\alpha}^{\star\prime} - \boldsymbol{\alpha}^\star\|_2 = \|\boldsymbol{\alpha}^{\star\prime}_\Lambda - \boldsymbol{\alpha}^\star_\Lambda\|_2 \leq \frac{1}{\kappa_2}(e_1\|\mathbf{D} - \mathbf{D}'\|_{\mathrm{F}} + e_2\|\mathbf{x} - \mathbf{x}'\|_2).$$

Therefore, $\boldsymbol{\alpha}^\star$ is locally Lipschitz. Since $K \times \mathcal{D}$ is compact, $\boldsymbol{\alpha}^\star$ is uniformly Lipschitz on $K \times \mathcal{D}$, which concludes the proof. □

### B.2.2 Proof of Proposition 5

*Proof.* Part of this proof is inspired by Bottou (1998). We prove the convergence of the sequence $\hat{f}_t(\mathbf{D}^t)$ by showing that the stochastic positive process

$$u_t \triangleq \hat{f}_t(\mathbf{D}^t) \geq 0,$$

is a quasi-martingale and use Theorem 3 from Fisk (1965) (see Appendix A), which states that if the sum of the "positive" variations of $u_t$ are bounded, $u_t$ is a quasi-martingale, which converges with probability one (see Theorem 3 for details). Computing the variations of $u_t$, we obtain

$$\begin{aligned} u_{t+1} - u_t &= \hat{f}_{t+1}(\mathbf{D}^{t+1}) - \hat{f}_t(\mathbf{D}^t) \\ &= \hat{f}_{t+1}(\mathbf{D}^{t+1}) - \hat{f}_{t+1}(\mathbf{D}^t) + \hat{f}_{t+1}(\mathbf{D}^t) - \hat{f}_t(\mathbf{D}^t) \\ &= \hat{f}_{t+1}(\mathbf{D}^{t+1}) - \hat{f}_{t+1}(\mathbf{D}^t) + \frac{\ell(\mathbf{x}^{t+1}, \mathbf{D}^t) - f_t(\mathbf{D}^t)}{t+1} + \frac{f_t(\mathbf{D}^t) - \hat{f}_t(\mathbf{D}^t)}{t+1}, \end{aligned} \tag{B.3}$$

using the fact that $\hat{f}_{t+1}(\mathbf{D}^t) = \frac{1}{t+1}\ell(\mathbf{x}_{t+1}, \mathbf{D}^t) + \frac{t}{t+1}\hat{f}_t(\mathbf{D}^t)$. Since $\mathbf{D}^{t+1}$ minimizes $\hat{f}_{t+1}$ on $\mathcal{D}$ and $\mathbf{D}^t$ is in $\mathcal{D}$, $\hat{f}_{t+1}(\mathbf{D}^{t+1}) - \hat{f}_{t+1}(\mathbf{D}^t) \leq 0$. Since the surrogate $\hat{f}_t$ upperbounds the empirical cost $f_t$, we also have $f_t(\mathbf{D}^t) - \hat{f}_t(\mathbf{D}^t) \leq 0$. To use Theorem 3, we consider the filtration of the past information $\mathcal{F}_t$ and take the expectation of Eq. (B.3) conditioned on $\mathcal{F}_t$, obtaining the following bound

$$\begin{aligned} \mathbb{E}[u_{t+1} - u_t | \mathcal{F}_t] &\leq \frac{\mathbb{E}[\ell(\mathbf{x}_{t+1}, \mathbf{D}^t)|\mathcal{F}_t] - f_t(\mathbf{D}^t)}{t+1} \\ &\leq \frac{f(\mathbf{D}^t) - f_t(\mathbf{D}^t)}{t+1} \\ &\leq \frac{\|f - f_t\|_\infty}{t+1}, \end{aligned}$$

For a specific matrix $\mathbf{D}$, the central-limit theorem states that $\mathbb{E}[\sqrt{t}(f(\mathbf{D}^t) - f_t(\mathbf{D}^t))]$ is bounded. However, we need here a stronger result on empirical processes to show that $\mathbb{E}[\sqrt{t}\|f - f_t\|_\infty]$ is bounded. To do so, we use the Lemma 13 in Appendix A, which is a corollary of Donsker theorem (see Van der Vaart, 1998, chap. 19.2). It is easy to show that in our case, all the hypotheses are verified, namely, $\ell(\mathbf{x}, .)$ is uniformly Lipschitz

and bounded since it is continuously differentiable on a compact set, the set $\mathcal{D} \subset \mathbb{R}^{m\times p}$ is bounded, and $\mathbb{E}_{\mathbf{x}}[\ell(\mathbf{x}, \mathbf{D})^2]$ exists and is uniformly bounded. Therefore, Lemma 13 applies and there exists a constant $\kappa > 0$ such that

$$\mathbb{E}[\mathbb{E}[u_{t+1} - u_t|\mathcal{F}_t]^+] \leq \frac{\kappa}{t^{\frac{3}{2}}}.$$

Therefore, defining $\delta_t$ as in Theorem 3, we have

$$\sum_{t=1}^{\infty} \mathbb{E}[\delta_t(u_{t+1} - u_t)] = \sum_{t=1}^{\infty} \mathbb{E}[\mathbb{E}[u_{t+1} - u_t|\mathcal{F}_t]^+] < +\infty.$$

Thus, we can apply Theorem 3, which proves that $u_t$ converges almost surely and that

$$\sum_{t=1}^{\infty} |\mathbb{E}[u_{t+1} - u_t|\mathcal{F}_t]| < +\infty \quad \text{a.s.}$$

Using Eq. (B.3) we can show that it implies the almost sure convergence of the positive sum

$$\sum_{t=1}^{\infty} \frac{\hat{f}_t(\mathbf{D}^t) - f_t(\mathbf{D}^t)}{t+1}.$$

Using Lemma 2 and the fact that the functions $f_t$ and $\hat{f}_t$ are bounded and Lipschitz, with a constant independent of $t$, it is easy to show that the hypotheses of Lemma 14 in Appendix A are satisfied. Therefore

$$f_t(\mathbf{D}^t) - \hat{f}_t(\mathbf{D}^t) \underset{t\to+\infty}{\longrightarrow} 0 \quad \text{a.s.}$$

Since $\hat{f}_t(\mathbf{D}^t)$ converges almost surely, this shows that $f_t(\mathbf{D}^t)$ converges in probability to the same limit. Note that we have in addition $\|f_t - f\|_\infty \to_{t\to+\infty} 0$ a.s. (see Van der Vaart, 1998, Theorem 19.4 (Glivenko-Cantelli)). Therefore,

$$f(\mathbf{D}^t) - \hat{f}_t(\mathbf{D}^t) \underset{t\to+\infty}{\longrightarrow} 0 \quad \text{a.s.}$$

and $f(\mathbf{D}^t)$ converges almost surely, which proves the second and third points. □

### B.2.3 Proof of Proposition 6

*Proof.* Since the sequences of matrices $\mathbf{B}^t, \mathbf{C}^t$ are in a compact set, it is possible to extract converging subsequences. Let us assume for a moment that these sequences converge respectively to two matrices $\mathbf{B}^\infty$ and $\mathbf{C}^\infty$. In that case, $\mathbf{D}^t$ converges to a matrix $\mathbf{D}^\infty$ in $\mathcal{D}$. Let $\mathbf{U}$ be a matrix in $\mathbb{R}^{m\times p}$. Since $\hat{f}_t$ upperbounds $f_t$ on $\mathbb{R}^{m\times p}$, for all $t$,

$$\hat{f}_t(\mathbf{D}^t + \mathbf{U}) \geq f_t(\mathbf{D}^t + \mathbf{U}).$$

Taking the limit when $t$ tends to infinity,

$$\hat{f}_\infty(\mathbf{D}^\infty + \mathbf{U}) \geq f(\mathbf{D}^\infty + \mathbf{U}).$$

Let $h_t > 0$ be a sequence that converges to 0. Using a first order Taylor expansion, and using the fact that $\nabla f$ is Lipschitz and $\hat{f}_\infty(\mathbf{D}^\infty) = f(\mathbf{D}^\infty)$ a.s., we have

$$f(\mathbf{D}^\infty) + \mathrm{Tr}(h_t \mathbf{U}^\top \nabla \hat{f}_\infty(\mathbf{D}^\infty)) + o(h_t \mathbf{U}) \geq f(\mathbf{D}^\infty) + \mathrm{Tr}(h_t \mathbf{U}^\top \nabla f(\mathbf{D}^\infty)) + o(h_t \mathbf{U}),$$

and it follows that

$$\mathrm{Tr}\left(\frac{1}{\|\mathbf{U}\|_\mathrm{F}} \mathbf{U}^\top \nabla \hat{f}_\infty(\mathbf{D}^\infty)\right) \geq \mathrm{Tr}\left(\frac{1}{\|\mathbf{U}\|_\mathrm{F}} \mathbf{U}^\top \nabla f(\mathbf{D}^\infty)\right),$$

Since this inequality is true for all $\mathbf{U}$, $\nabla \hat{f}_\infty(\mathbf{D}^\infty) = \nabla f(\mathbf{D}^\infty)$. A first-order necessary optimality condition for $\mathbf{D}^\infty$ being an optimum of $\hat{f}_\infty$ is that $-\nabla \hat{f}_\infty$ is in the *normal cone* of the set $\mathcal{D}$ at $\mathbf{D}^\infty$ (Borwein and Lewis, 2006). Therefore, this first-order necessary conditions is verified for $f$ at $\mathbf{D}^\infty$ as well. Since $\mathbf{B}^t, \mathbf{C}^t$ are asymptotically close to their accumulation points, $-\nabla f(\mathbf{D}^t)$ is asymptotically close the normal cone at $\mathbf{D}^t$ and these first-order optimality conditions are verified asymptotically with probability one. □

### B.2.4 Proof of Proposition 7

*Proof.* The proof largely relies on 5. We proceed by induction, by showing that we keep the optimality conditions of 3.6 satisfied after each update in Algorithm 3. By definition of Algorithm 3, note that the feasibility of $\boldsymbol{\xi}$ is always guaranteed. We consider the following induction hypothesis

$$\mathcal{H}(h) \triangleq \{\forall g \preceq h, \text{ it holds that } \boldsymbol{\xi}^g = \Pi^*_{\lambda\eta_g}([\mathbf{u} - \textstyle\sum_{g' \neq g} \boldsymbol{\xi}^{g'}]_{|g})\}.$$

Since the dual variables $\boldsymbol{\xi}$ are initially equal to zero, the summation over $g' \neq g$ in the definition of $\mathcal{H}$ can be instead taken over $g' \preceq h,\ g' \neq g$, leading to

$$\mathcal{H}(h) = \{\forall g \preceq h, \text{ it holds that } \boldsymbol{\xi}^g = \Pi^*_{\lambda\eta_g}([\mathbf{u} - \textstyle\sum_{g' \preceq h,\ g' \neq g} \boldsymbol{\xi}^{g'}]_{|g})\}.$$

We initialize the induction with the *first* group in $\mathcal{G}$, that, by definition of $\preceq$, does not contain any other group. The first step of Algorithm 3 easily shows that the induction hypothesis $\mathcal{H}$ is satisfied for this first group.

We now assume that $\mathcal{H}(h)$ is true and consider the next group $h'$, $h \preceq h'$, in order to prove that $\mathcal{H}(h')$ is also satisfied. We have for each group $g \subseteq h$,

$$\boldsymbol{\xi}^g = \Pi^*_{\lambda\eta_g}([\mathbf{u} - \textstyle\sum_{g' \preceq h,\ g' \neq g} \boldsymbol{\xi}^{g'}]_{|g}).$$

Following the update of the group $h'$, we have

$$\begin{aligned} \boldsymbol{\xi}^{h'} &= \Pi^*_{\lambda\eta_{h'}}([\mathbf{u} - \textstyle\sum_{g' \preceq h} \boldsymbol{\xi}^{g'}]_{|h'}) \\ &= \Pi^*_{\lambda\eta_{h'}}([\mathbf{u} - \textstyle\sum_{g' \preceq h',\ g' \neq h'} \boldsymbol{\xi}^{g'}]_{|h'}). \end{aligned}$$

At this point, we can apply 5 for each group $g \subseteq h$, which proves

$$\begin{aligned} \boldsymbol{\xi}^g &= \Pi^*_{\lambda\eta_g}([\mathbf{u} - \textstyle\sum_{g' \preceq h,\ g' \neq g} \boldsymbol{\xi}^{g'} - \boldsymbol{\xi}^{h'}]_{|g}) \\ &= \Pi^*_{\lambda\eta_g}([\mathbf{u} - \textstyle\sum_{g' \preceq h',\ g' \neq g} \boldsymbol{\xi}^{g'}]_{|g}). \end{aligned}$$

As a result, the induction hypothesis $\mathcal{H}(h')$ is true.

Therefore, after one complete pass over $g \in \mathcal{G}$, the dual variable $\boldsymbol{\xi}$ satisfies the optimality conditions for 3.6, which implies that the pair $\{\mathbf{v}, \boldsymbol{\xi}\}$ is optimal. Since strong duality holds, $\mathbf{v}$ is the solution of 3.3. □

### B.2.5 Proof of Proposition 8

*Proof.* Our algorithm splits recursively the graph into disjoints parts and processes each part recursively. The processing of one part requires an orthogonal projection onto an $\ell_1$-ball and a max-flow algorithm, which can both be computed in polynomial time. To prove that the procedure converges, it is sufficient to show that when the procedure `computeFlow` is called for a graph $(V, E, s, t)$ and computes a cut $(V^+, V^-)$, then the components $V^+$ and $V^-$ are both non-empty.

Suppose for instance that $V^- = \emptyset$. In this case, the capacity of the min-cut is equal to $\sum_{j \in V_u} \boldsymbol{\gamma}_j$, and the value of the max-flow is $\sum_{j \in V_u} \bar{\boldsymbol{\xi}}_j$. Using the classical max-flow/min-cut theorem (Ford and Fulkerson, 1956), we have equality between these two terms. Since, by definition of both $\boldsymbol{\gamma}$ and $\bar{\boldsymbol{\xi}}$, we have for all $j$ in $V_u$, $\bar{\boldsymbol{\xi}}_j \leq \boldsymbol{\gamma}_j$, we obtain a contradiction with the existence of $j$ in $V_u$ such that $\bar{\boldsymbol{\xi}}_j \neq \boldsymbol{\gamma}_j$.

Conversely, suppose now that $V^+ = \emptyset$. Then, the value of the max-flow is still $\sum_{j \in V_u} \bar{\boldsymbol{\xi}}_j$, and the value of the min-cut is $\lambda \sum_{g \in V_{gr}} \eta_g$. Using again the max-flow/min-cut theorem, we have that $\sum_{j \in V_u} \bar{\boldsymbol{\xi}}_j = \lambda \sum_{g \in V_{gr}} \eta_g$. Moreover, by definition of $\boldsymbol{\gamma}$, we also have $\sum_{j \in V_u} \bar{\boldsymbol{\xi}}_j \leq \sum_{j \in V_u} \boldsymbol{\gamma}_j \leq \lambda \sum_{g \in V_{gr}} \eta_g$, leading to a contradiction with the existence of $j$ in $V_u$ such that $\bar{\boldsymbol{\xi}}_j \neq \boldsymbol{\gamma}_j$. This proof holds for any graph that is equivalent to the canonical one. □

### B.2.6 Proof of Proposition 9

*Proof.* For a group structure $\mathcal{G}$, we first prove the correctness of our algorithm if the graph used is its associated canonical graph that we denote $G_0 = (V_0, E_0, s, t)$. We proceed by induction on the number of nodes of the graph. The induction hypothesis $\mathcal{H}(k)$ is the following:

*For all canonical graphs $G = (V = V_u \cup V_{gr}, E, s, t)$ associated with a group structure $\mathcal{G}_V$ with weights $(\eta_g)_{g \in \mathcal{G}_V}$ such that $|V| \leq k$, `computeFlow`$(V, E)$ solves the following optimization problem:*

$$\min_{(\boldsymbol{\xi}_j^g)_{j \in V_u, g \in V_{gr}}} \sum_{j \in V_u} \frac{1}{2}(\mathbf{u}_j - \sum_{g \in V_{gr}} \boldsymbol{\xi}_j^g)^2 \quad \text{s.t.} \quad \forall g \in V_{gr}, \ \sum_{j \in V_u} \boldsymbol{\xi}_j^g \leq \lambda \eta_g \quad \text{and} \quad \boldsymbol{\xi}_j^g = 0, \ \forall j \notin g. \tag{B.4}$$

Since $\mathcal{G}_{V_0} = \mathcal{G}$, it is sufficient to show that $\mathcal{H}(|V_0|)$ to prove the proposition.

We initialize the induction by $\mathcal{H}(2)$, corresponding to the simplest canonical graph, for which $|V_{gr}| = |V_u| = 1$). Simple algebra shows that $\mathcal{H}(2)$ is indeed correct.

We now suppose that $\mathcal{H}(k')$ is true for all $k' < k$ and consider a graph $G = (V, E, s, t)$, $|V| = k$. The first step of the algorithm computes the variable $(\boldsymbol{\gamma}_j)_{j \in V_u}$ by a projection on

the $\ell_1$-ball. This is itself an instance of the dual formulation of Eq. (3.6) in a simple case, with one group containing all variables. We can therefore use Lemma 10 to characterize the optimality of $(\boldsymbol{\gamma}_j)_{j\in V_u}$, which yields

$$\begin{cases} \sum_{j\in V_u}(\mathbf{u}_j-\boldsymbol{\gamma}_j)\boldsymbol{\gamma}_j = \left(\max_{j\in V_u}|\mathbf{u}_j-\boldsymbol{\gamma}_j|\right)\sum_{j\in V_u}\boldsymbol{\gamma}_j \text{ and } \sum_{j\in V_u}\boldsymbol{\gamma}_j = \lambda\sum_{g\in V_{gr}}\eta_g, \\ \text{or } \ \mathbf{u}_j-\boldsymbol{\gamma}_j = 0, \ \forall j\in V_u. \end{cases} \tag{B.5}$$

The algorithm then computes a max-flow, using the scalars $\boldsymbol{\gamma}_j$ as capacities, and we now have two possible situations:

1. If $\bar{\boldsymbol{\xi}}_j = \boldsymbol{\gamma}_j$ for all $j$ in $V_u$, the algorithm stops; we write $\mathbf{w}_j = \mathbf{u}_j - \bar{\boldsymbol{\xi}}_j$ for $j$ in $V_u$, and using Eq. (B.5), we obtain

$$\begin{cases} \sum_{j\in V_u}\mathbf{w}_j\bar{\boldsymbol{\xi}}_j = (\max_{j\in V_u}|\mathbf{w}_j|)\sum_{j\in V_u}\bar{\boldsymbol{\xi}}_j \ \text{ and } \ \sum_{j\in V_u}\bar{\boldsymbol{\xi}}_j = \lambda\sum_{g\in V_{gr}}\eta_g, \\ \text{or } \ \mathbf{w}_j = 0, \ \forall j\in V_u. \end{cases} \tag{B.6}$$

   We can rewrite the condition above as

$$\sum_{g\in V_{gr}}\sum_{j\in g}\mathbf{w}_j\boldsymbol{\xi}_j^g = \sum_{g\in V_{gr}}(\max_{j\in V_u}|\mathbf{w}_j|)\sum_{j\in V_u}\boldsymbol{\xi}_j^g.$$

   Since all the quantities in the previous sum are positive, this can only hold if for all $g\in V_{gr}$,

$$\sum_{j\in V_u}\mathbf{w}_j\boldsymbol{\xi}_j^g = (\max_{j\in V_u}|\mathbf{w}_j|)\sum_{j\in V_u}\boldsymbol{\xi}_j^g.$$

   Moreover, by definition of the max flow and the optimality conditions, we have

$$\forall g\in V_{gr}, \ \sum_{j\in V_u}\boldsymbol{\xi}_j^g \le \lambda\eta_g, \ \text{ and } \ \sum_{j\in V_u}\bar{\boldsymbol{\xi}}_j = \lambda\sum_{g\in V_{gr}}\eta_g,$$

   which leads to

$$\forall g\in V_{gr}, \ \sum_{j\in V_u}\boldsymbol{\xi}_j^g = \lambda\eta_g.$$

   By Lemma 10, we have shown that the problem (B.4) is solved.

2. Let us now consider the case where there exists $j$ in $V_u$ such that $\bar{\boldsymbol{\xi}}_j \neq \boldsymbol{\gamma}_j$. The algorithm splits the vertex set $V$ into two parts $V^+$ and $V^-$, which we have proven to be non-empty in the proof of Proposition 8. The next step of the algorithm removes all edges between $V^+$ and $V^-$ (see Figure 3.3). Processing $(V^+,E^+)$ and $(V^-,E^-)$ independently, it updates the value of the flow matrix $\boldsymbol{\xi}_j^g$, $j\in V_u$, $g\in V_{gr}$, and the corresponding flow vector $\bar{\boldsymbol{\xi}}_j$, $j\in V_u$. As for $V$, we denote by $V_u^+ \triangleq V^+\cap V_u$, $V_u^- \triangleq V^-\cap V_u$ and $V_{gr}^+ \triangleq V^+\cap V_{gr}$, $V_{gr}^- \triangleq V^-\cap V_{gr}$.

   Then, we notice that $(V^+,E^+,s,t)$ and $(V^-,E^-,s,t)$ are respective canonical graphs for the group structures $\mathcal{G}_{V^+} \triangleq \{g\cap V_u^+ \mid g\in V_{gr}\}$, and $\mathcal{G}_{V^-} \triangleq \{g\cap V_u^- \mid g\in V_{gr}\}$.

Writing $\mathbf{w}_j = \mathbf{u}_j - \bar{\boldsymbol{\xi}}_j$ for $j$ in $V_u$, and using the induction hypotheses $\mathcal{H}(|V^+|)$ and $\mathcal{H}(|V^-|)$, we now have the following optimality conditions deriving from Lemma 10 applied on Eq. (B.4) respectively for the graphs $(V^+, E^+)$ and $(V^-, E^-)$:

$$\forall g \in V_{gr}^+, g' \triangleq g \cap V_u^+, \begin{cases} \mathbf{w}_{g'}^\top \boldsymbol{\xi}_{g'}^g = \|\mathbf{w}_{g'}\|_\infty \sum_{j \in g'} \boldsymbol{\xi}_j^g \quad \text{and} \quad \sum_{j \in g'} \boldsymbol{\xi}_j^g = \lambda \eta_g, \\ \text{or} \quad \mathbf{w}_{g'} = 0, \end{cases} \tag{B.7}$$

and

$$\forall g \in V_{gr}^-, g' \triangleq g \cap V_u^-, \begin{cases} \mathbf{w}_{g'}^\top \boldsymbol{\xi}_{g'}^g = \|\mathbf{w}_{g'}\|_\infty \sum_{j \in g'} \boldsymbol{\xi}_j^g \quad \text{and} \quad \sum_{j \in g'} \boldsymbol{\xi}_j^g = \lambda \eta_g, \\ \text{or} \quad \mathbf{w}_{g'} = 0. \end{cases} \tag{B.8}$$

We will now combine Eq. (B.7) and Eq. (B.8) into optimality conditions for Eq. (B.4). We first notice that $g \cap V_u^+ = g$ since there are no arcs between $V^+$ and $V^-$ in $E$ (see the properties of the cuts discussed before this proposition). It is therefore possible to replace $g'$ by $g$ in Eq. (B.7). We will show that it is possible to do the same in Eq. (B.8), so that combining these two equations yield the optimality conditions of Eq. (B.4).

More precisely, we will show that for all $g \in V_{gr}^-$ and $j \in g \cap V_u^+$, $|\mathbf{w}_j| \leq \max_{l \in g \cap V_u^-} |\mathbf{w}_l|$, in which case $g'$ can be replaced by $g$ in Eq. (B.8). This result is relatively intuitive: $(s, V^+)$ and $(V^-, t)$ being an $(s,t)$-cut, all arcs between $s$ and $V^-$ are saturated, while there are unsaturated arcs between $s$ and $V^+$; one therefore expects the residuals $\mathbf{u}_j - \bar{\boldsymbol{\xi}}_j$ to decrease on the $V^+$ side, while increasing on the $V^-$ side. The proof is nonetheless a bit technical.

Let us show first that for all $g$ in $V_{gr}^+$, $\|\mathbf{w}_g\|_\infty \leq \max_{j \in V_u} |\mathbf{u}_j - \boldsymbol{\gamma}_j|$. We split the set $V^+$ into disjoint parts:

$$\begin{aligned}
V_{gr}^{++} &\triangleq \{g \in V_{gr}^+ \ \text{ s.t. } \ \|\mathbf{w}_g\|_\infty \leq \max_{j \in V_u} |\mathbf{u}_j - \boldsymbol{\gamma}_j|\}, \\
V_u^{++} &\triangleq \{j \in V_u^+ \ \text{ s.t. } \ \exists g \in V_{gr}^{++}, \ j \in g\}, \\
V_{gr}^{+-} &\triangleq V_{gr}^+ \setminus V_{gr}^{++} = \{g \in V_{gr}^+ \ \text{ s.t. } \ \|\mathbf{w}_g\|_\infty > \max_{j \in V_u} |\mathbf{u}_j - \boldsymbol{\gamma}_j|\}, \\
V_u^{+-} &\triangleq V_u^+ \setminus V_u^{++}.
\end{aligned}$$

As previously, we denote $V^{+-} \triangleq V_u^{+-} \cup V_{gr}^{+-}$ and $V^{++} \triangleq V_u^{++} \cup V_{gr}^{++}$. We want to show that $V_{gr}^{+-}$ is necessarily empty. We reason by contradiction and assume that $V_{gr}^{+-} \neq \varnothing$.

According to the definition of the different sets above, we observe that no arcs are going from $V^{++}$ to $V^{+-}$, that is, for all $g$ in $V_{gr}^{++}$, $g \cap V_u^{+-} = \varnothing$. We observe as well that the flow from $V_{gr}^{+-}$ to $V_u^{++}$ is the null flow, because optimality conditions (B.7) imply that for a group $g$ only nodes $j \in g$ such that $\mathbf{w}_j = \|\mathbf{w}_g\|_\infty$ receive some flow, which excludes nodes in $V_u^{++}$ provided $V_{gr}^{+-} \neq \varnothing$; Combining this fact and the inequality $\sum_{g \in V_{gr}^+} \lambda \eta_g \geq \sum_{j \in V_u^+} \boldsymbol{\gamma}_j$ (which is a direct consequence of the

minimum $(s,t)$-cut), we have as well

$$\sum_{g \in V_{gr}^{+-}} \lambda \eta_g \geq \sum_{j \in V_u^{+-}} \boldsymbol{\gamma}_j.$$

Let $j \in V_u^{+-}$, if $\bar{\boldsymbol{\xi}}_j \neq 0$ then for some $g \in V_{gr}^{+-}$ such that $j$ receives some flow from $g$, which from the optimality conditions (B.7) implies $\mathbf{w}_j = \|\mathbf{w}_g\|_\infty$; by definition of $V_{gr}^{+-}$, $\|\mathbf{w}_g\|_\infty > \mathbf{u}_j - \boldsymbol{\gamma}_j$. But since at the optimum, $\mathbf{w}_j = \mathbf{u}_j - \bar{\boldsymbol{\xi}}_j$, this implies that $\bar{\boldsymbol{\xi}}_j < \boldsymbol{\gamma}_j$, and in turn that $\sum_{j \in V_u^{+-}} \bar{\boldsymbol{\xi}}_j = \lambda \sum_{g \in V_{gr}^{+-}} \eta_g$. Finally,

$$\lambda \sum_{g \in V_{gr}^{+-}} \eta_g = \sum_{j \in V_u^{+-}, \bar{\boldsymbol{\xi}}_j \neq 0} \bar{\boldsymbol{\xi}}_j < \sum_{j \in V_u^{+-}} \boldsymbol{\gamma}_j$$

and this is a contradiction.

We now have that for all $g$ in $V_{gr}^{+}$, $\|\mathbf{w}_g\|_\infty \leq \max_{j \in V_u} |\mathbf{u}_j - \boldsymbol{\gamma}_j|$. The proof showing that for all $g$ in $V_{gr}^{-}$, $\|\mathbf{w}_g\|_\infty \geq \max_{j \in V_u} |\mathbf{u}_j - \boldsymbol{\gamma}_j|$, uses the same kind of decomposition for $V^-$, and follows along similar arguments. We will therefore not detail it.

To recap, we have shown that for all $g \in V_{gr}^{-}$ and $j \in g \cap V_u^{+}$, $|\mathbf{w}_j| \leq \max_{l \in g \cap V_u^{-}} |\mathbf{w}_l|$. Since there is no flow from $V^-$ to $V^+$, i.e., $\boldsymbol{\xi}_j^g = 0$ for $g$ in $V_{gr}^{-}$ and $j$ in $V_u^{+}$, we can now replace the definition of $g'$ in Eq. (B.8) by $g' \triangleq g \cap V_u$, the combination of Eq. (B.7) and Eq. (B.8) gives us optimality conditions for Eq. (B.4).

The proposition being proved for the canonical graph, we extend it now for an equivalent graph in the sense of Lemma 9. First, we observe that the algorithm gives the same values of $\boldsymbol{\gamma}$ for two equivalent graphs. Then, it is easy to see that the value $\bar{\boldsymbol{\xi}}$ given by the max-flow, and the chosen $(s,t)$-cut is the same, which is enough to conclude that the algorithm performs the same steps for two equivalent graphs. □

### B.2.7 Proof of Proposition 10

*Proof.* The convergence of the algorithm only requires to show that the cardinality of $V$ in the different calls of the function `computeFlow` strictly decreases. Similar arguments to those used in the proof of Proposition 8 can show that each part of the cuts $(V^+, V^-)$ are both non-empty. The algorithm thus requires a finite number of calls to a max-flow algorithm and converges in a finite and polynomial number of operations.

Let us now prove that the algorithm is correct for a canonical graph. We proceed again by induction on the number of nodes of the graph. More precisely, we consider the induction hypothesis $\mathcal{H}'(k)$ defined as:

*for all canonical graphs* $G = (V, E, s, t)$ associated with a group structure $\mathcal{G}_V$ and *such that* $|V| \leq k$, `dualNormAux`$(V = V_u \cup V_{gr}, E)$ *solves the following optimization problem:*

$$\min_{\boldsymbol{\xi}, \tau} \tau \quad \text{s.t.} \quad \forall j \in V_u, \boldsymbol{\kappa}_j = \sum_{g \in V_{gr}} \boldsymbol{\xi}_j^g, \text{ and } \forall g \in V_{gr}, \sum_{j \in V_u} \boldsymbol{\xi}_j^g \leq \tau \eta_g \text{ with } \boldsymbol{\xi}_j^g = 0 \text{ if } j \notin g. \tag{B.9}$$

We first initialize the induction by $\mathcal{H}(2)$ (i.e., with the simplest canonical graph, such that $|V_{gr}| = |V_u| = 1$). Simple algebra shows that $\mathcal{H}(2)$ is indeed correct.

We next consider a canonical graph $G = (V, E, s, t)$ such that $|V| = k$, and suppose that $\mathcal{H}'(k-1)$ is true. After the max-flow step, we have two possible cases to discuss:

1. If $\bar{\boldsymbol{\xi}}_j = \boldsymbol{\gamma}_j$ for all $j$ in $V_u$, the algorithm stops. We know that any scalar $\tau$ such that the constraints of Eq. (B.9) are all satisfied necessarily verifies $\sum_{g \in V_{gr}} \tau \eta_g \geq \sum_{j \in V_u} \boldsymbol{\kappa}_j$. We have indeed that $\sum_{g \in V_{gr}} \tau \eta_g$ is the value of an $(s,t)$-cut in the graph, and $\sum_{j \in V_u} \boldsymbol{\kappa}_j$ is the value of the max-flow, and the inequality follows from the max-flow/min-cut theorem (Ford and Fulkerson, 1956). This gives a lower-bound on $\tau$. Since this bound is reached, $\tau$ is necessarily optimal.

2. We now consider the case where there exists $j$ in $V_u$ such that $\bar{\boldsymbol{\xi}}_j \neq \boldsymbol{\kappa}_j$, meaning that for the given value of $\tau$, the constraint set of Eq. (B.9) is not feasible for $\boldsymbol{\xi}$, and that the value of $\tau$ should necessarily increase. The algorithm splits the vertex set $V$ into two non-empty parts $V^+$ and $V^-$ and we remark that there are no arcs going from $V^+$ to $V^-$, and no flow going from $V^-$ to $V^+$. Since the arcs going from $s$ to $V^-$ are saturated, we have that $\sum_{g \in V_{gr}^-} \tau \eta_g \leq \sum_{j \in V_u^-} \boldsymbol{\kappa}_j$. Let us now consider $\tau^\star$ the solution of Eq. (B.9). Using the induction hypothesis $\mathcal{H}'(|V^-|)$, the algorithm computes a new value $\tau'$ that solves Eq. (B.9) when replacing $V$ by $V^-$ and this new value satisfies the following inequality $\sum_{g \in V_{gr}^-} \tau' \eta_g \geq \sum_{j \in V_u^-} \boldsymbol{\kappa}_j$. The value of $\tau'$ has therefore increased and the updated flow $\boldsymbol{\xi}$ now satisfies the constraints of Eq. (B.9) and therefore $\tau' \geq \tau^\star$. Since there are no arcs going from $V^+$ to $V^-$, $\tau^\star$ is feasible for Eq. (B.9) when replacing $V$ by $V^-$ and we have that $\tau^\star \geq \tau'$ and then $\tau' = \tau^\star$.

To prove that the result holds for any equivalent graph, similar arguments to those used in the proof of Proposition 8 can be exploited, showing that the algorithm computes the same values of $\tau$ and same $(s,t)$-cuts at each step. □

### B.2.8 Proof of Proposition 11

*Proof.* The first point is proven in the proof of Proposition 4. The proof uses the strong convexity induced by the elastic-net term, when $\lambda_2 > 0$, and the compactness of $\mathcal{X}$ from Assumption **(A)**.

For the second point, we study the differentiability of $\boldsymbol{\alpha}^\star$ on sets that satisfy conditions which are more restrictive than the optimality conditions of Eq. (6.16). Concretely, let $\mathbf{D}$ be in $\mathcal{D}$, $\varepsilon > 0$ and $\mathbf{s}$ be in $\{-1, 0, +1\}^p$. The set $K_{\mathbf{s}}(\mathbf{D}, \varepsilon)$ characterizes the vectors $\mathbf{x}$ so that $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$ has the same signs as $\mathbf{s}$ (and same set of zero coefficients), and $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$ satisfies the conditions of Eq. (6.16), but with two additional constraints: (i) The magnitude of the non-zero coefficients in $\boldsymbol{\alpha}^\star$ should be greater than $\varepsilon$. (ii) The inequalities in Eq. (6.16) should be strict with a margin $\varepsilon$. The reason for imposing these assumptions is to restrict ourselves to points $\mathbf{x}$ in $\mathcal{X}$ that have a stable active set—that is, the set of non-zero coefficients $\Lambda$ of $\boldsymbol{\alpha}^\star$ should not change for small perturbations of $(\mathbf{x}, \mathbf{D})$, when $\mathbf{x}$ is in $K_{\mathbf{s}}(\mathbf{D}, \varepsilon)$.

Proving that there exists a constant $\kappa > 0$ satisfying the second point is then easy (if a bit technical): Let us suppose that $K_{\mathbf{s}}(\mathbf{D}, \varepsilon)$ is not empty (the case when it is empty is trivial). Since $\boldsymbol{\alpha}^\star$ is uniformly Lipschitz with respect to $(\mathbf{x}, \mathbf{D})$, so are the quantities $\mathbf{d}^{j\top}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star) - \lambda_2 \boldsymbol{\alpha}_j^\star$ and $\mathbf{s}_j \boldsymbol{\alpha}_j^\star$, for all $j$ in $\{1, \dots, p\}$. Thus, there exists $\kappa > 0$ independent of $\mathbf{x}$ and $\mathbf{D}$ such that for all $(\mathbf{x}', \mathbf{D}')$ satisfying $\|\mathbf{x} - \mathbf{x}'\|_2 \leq \kappa\varepsilon$ and $\|\mathbf{D} - \mathbf{D}'\|_{\mathrm{F}} \leq \kappa\varepsilon$, we have

$$\forall j \in \{1, \dots, p\}, \begin{cases} |\mathbf{d}'^{j\top}(\mathbf{x}' - \mathbf{D}'\boldsymbol{\alpha}^{\star\prime}) - \lambda_2 \boldsymbol{\alpha}_j^{\star\prime}| \leq \lambda_1 - \frac{\varepsilon}{2} & \text{if} \quad \mathbf{s}_j = 0, \\ \mathbf{s}_j \boldsymbol{\alpha}_j^{\star\prime} \geq \frac{\varepsilon}{2} & \text{if} \quad \mathbf{s}_j \neq 0. \end{cases}$$

where $\boldsymbol{\alpha}^{\star\prime}$ is short-hand for $\boldsymbol{\alpha}^\star(\mathbf{x}', \mathbf{D}')$, and $\mathbf{x}'$ is therefore in $K_{\mathbf{s}}(\mathbf{D}', \varepsilon/2)$. It is then easy to show that the active set $\Lambda$ of $\boldsymbol{\alpha}^\star$ and the signs of $\boldsymbol{\alpha}^\star$ are stable on $B_{\kappa\varepsilon}(\mathbf{x}) \times B_{\kappa\varepsilon}(\mathbf{D})$, and that $\boldsymbol{\alpha}_\Lambda^\star$ is given by the closed form of Eq. (6.17). $\boldsymbol{\alpha}^\star$ is therefore twice differentiable on $B_{\kappa\varepsilon}(\mathbf{x}) \times B_{\kappa\varepsilon}(\mathbf{D})$. □

### B.2.9 Proof of Proposition 12

*Proof.* The differentiability of $f$ with respect to $\mathbf{W}$ is easy using only the compactness of $\mathcal{Y}$ and $\mathcal{X}$ and the fact that $\ell_s$ is twice differentiable. We will therefore focus on showing that $f$ is differentiable with respect to $\mathbf{D}$, which is more difficult since $\boldsymbol{\alpha}^\star$ is *not* differentiable everywhere.

Given a small perturbation $\mathbf{E}$ in $\mathbb{R}^{m \times p}$ of $\mathbf{D}$, we have and compute

$$f(\mathbf{D}+\mathbf{E}, \mathbf{W}) - f(\mathbf{D}, \mathbf{W}) = \mathbb{E}_{\mathbf{y},\mathbf{x}}\Big[\nabla_{\boldsymbol{\alpha}} \ell_s(\mathbf{y}, \mathbf{W}, \boldsymbol{\alpha}^\star)^\top (\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}+\mathbf{E}) - \boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}))\Big] + O(\|\mathbf{E}\|_{\mathrm{F}}^2), \tag{B.10}$$

where the term $O(\|\mathbf{E}\|_{\mathrm{F}}^2)$ comes from the fact that $\boldsymbol{\alpha}^\star$ is uniformly Lipschitz and $\mathcal{X} \times \mathcal{D}$ is compact.

Let now choose $\mathbf{W}$ in $\mathcal{W}$ and $\mathbf{D}$ in $\mathcal{D}$. We have characterized in Lemma 11 the differentiability of $\boldsymbol{\alpha}^\star$ on some subsets of $\mathcal{X} \times \mathcal{D}$. We consider the set

$$K(\mathbf{D}, \varepsilon) \triangleq \bigcup_{\mathbf{s} \in \{-1,0,1\}^p} K_{\mathbf{s}}(\mathbf{D}, \varepsilon),$$

and denoting by $\mathbb{P}$ our probability measure, it is easy to show with a few calculations that $\mathbb{P}(\mathcal{X} \setminus K(\mathbf{D}, \varepsilon)) = O(\varepsilon)$. Using the constant $\kappa$ defined in Lemma 11, we obtain that $\mathbb{P}(\mathcal{X} \setminus K(\mathbf{D}, \|\mathbf{E}\|_{\mathrm{F}}/\kappa)) = O(\|\mathbf{E}\|_{\mathrm{F}})$. Since $\nabla_{\boldsymbol{\alpha}} \ell_s(\mathbf{y}, \mathbf{W}, \boldsymbol{\alpha}^\star)^\top (\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}+\mathbf{E}) - \boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})) = O(\|\mathbf{E}\|_{\mathrm{F}})$, the set $\mathcal{X} \setminus K(\mathbf{D}, \|\mathbf{E}\|_{\mathrm{F}}/\kappa)$ can be neglected (in the formal sense) when integrating with respect to $\mathbf{x}$ in the expectation of Eq. (B.10), and it is possible to show that

$$f(\mathbf{D}+\mathbf{E}, \mathbf{W}) - f(\mathbf{D}, \mathbf{W}) = \mathrm{Tr}\left(\mathbf{E}^\top g(\mathbf{D}, \mathbf{W})\right) + O(\|\mathbf{E}\|_{\mathrm{F}}^2),$$

where $g$ has the form given by Eq. (6.19). This shows that $f$ is differentiable with respect to $\mathbf{D}$, and its gradient $\nabla_{\mathbf{D}} f$ is $g$. □

# C

# Efficient Projection Algorithms

In this section, we address the problem of efficiently projecting a vector onto two sets of constraints, which allows us to extend our algorithm to various other formulations.

## C.1 A Linear-time Projection Algorithm on the Elastic-Net Constraint

Let $\mathbf{u}$ be a vector of $\mathbb{R}^m$. We consider the problem of projecting this vector onto the elastic-net constraint set:

$$\min_{\mathbf{v}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{v}\|_1 + \frac{\gamma}{2}\|\mathbf{v}\|_2^2 \leq \tau. \tag{C.1}$$

To solve efficiently the case $\gamma > 0$, we propose Algorithm 9, which extends Brucker (1984); Maculan and de Paula (1989) and Duchi et al. (2008), and the following lemma which shows that it solves our problem.

**Lemma 15 (Projection onto the elastic-net constraint set.)**
*For* $\mathbf{u}$ *in* $\mathbb{R}^m$, $\gamma \geq 0$ *and* $\tau > 0$, *Algorithm 9 solves Eq. (C.1).*

*Proof.* First, if $\mathbf{u}$ is a feasible point of (C.1), then $\mathbf{u}$ is a solution. We suppose therefore that it is not the case—that is, $\|\mathbf{u}\|_1 + \frac{\gamma}{2}\|\mathbf{u}\|_2^2 > \tau$. Let us define the Lagrangian of (C.1)

$$\mathcal{L}(\mathbf{v},\lambda) = \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 + \lambda(\|\mathbf{v}\|_1 + \frac{\gamma}{2}\|\mathbf{v}\|_2^2 - \tau).$$

For a fixed $\lambda$, minimizing the Lagrangian with respect to $\mathbf{v}$ admits a closed-form solution $\mathbf{v}^\star(\lambda)$, and a simple calculation shows that, for all $j$,

$$\mathbf{v}_j^\star(\lambda) = \frac{\operatorname{sign}(\mathbf{u}_j)(|\mathbf{u}_j| - \lambda)^+}{1+\lambda\gamma}.$$

Eq. (C.1) is a convex optimization problem. Since Slater's conditions are verified and strong duality holds, it is equivalent to the dual problem

$$\max_{\lambda\geq 0} \mathcal{L}(\mathbf{v}^\star(\lambda),\lambda).$$

Since $\lambda = 0$ is not a solution, denoting by $\lambda^\star$ the solution, the complementary slackness condition implies that

$$\|\mathbf{v}^\star(\lambda^\star)\|_1 + \frac{\gamma}{2}\|\mathbf{v}^\star(\lambda^\star)\|_2^2 = \tau. \tag{C.2}$$

Using the closed form of $\mathbf{v}^\star(\lambda)$ is possible to show that the function $\lambda \to \|\mathbf{v}^\star(\lambda)\|_1 + \frac{\gamma}{2}\|\mathbf{v}^\star(\lambda)\|_2^2$, is strictly decreasing with $\lambda$ and thus Eq. (C.2) is a necessary and sufficient condition of optimality for $\lambda$. After a short calculation, one can show that this optimality condition is equivalent to

$$\frac{1}{(1+\lambda\gamma)^2}\sum_{j\in S(\lambda)}\Big(|\mathbf{u}[j]| + \frac{\gamma}{2}|\mathbf{u}_j|^2 - \lambda(1+\frac{\gamma\lambda}{2})\Big) = \tau,$$

where $S(\lambda) = \{j \quad \text{s.t.} \quad |\mathbf{u}_j| \geq \lambda\}$. Suppose that $S(\lambda^\star)$ is known, then $\lambda^\star$ can be computed in closed-form. To find $S(\lambda^\star)$, it is then sufficient to find the index $k$ such that $S(\lambda^\star) = S(|\mathbf{u}_k|)$, which is the solution of

$$\max_{k\in\{1,\ldots,m\}} |\mathbf{u}_k| \quad \text{s.t.} \quad \frac{1}{(1+|\mathbf{u}_k|\gamma)^2}\sum_{j\in S(|\mathbf{u}_k|)}\Big(|\mathbf{u}_j| + \frac{\gamma}{2}|\mathbf{u}_j|^2 - |\mathbf{u}_k|(1+\frac{\gamma|\mathbf{u}_k|}{2})\Big) < \tau.$$

Lines 4 to 14 of Algorithm 9 are a modification of Duchi et al. (2008) to address this problem. A similar proof as Duchi et al. (2008) shows the convergence to the solution of this optimization problem in $O(m)$ in the average case, and lines 15 to 18 of Algorithm 9) compute $\lambda^\star$ after that $S(\lambda^\star)$ has been identified. Note that setting $\gamma$ to 0 leads exactly to the algorithm of Duchi et al. (2008). □

As for the dictionary learning problem, a simple modification to Algorithm 9 allows us to handle the non-negative case, replacing the scalars $|\mathbf{u}_j|$ by $\max(\mathbf{u}_j, 0)$ in the algorithm.

## C.2 A Homotopy Method for Solving the Fused Lasso Signal Approximation

Let $\mathbf{u}$ be a vector of $\mathbb{R}^m$. We define, following Friedman et al. (2007), the fused lasso signal approximation problem $\mathcal{P}(\gamma_1, \gamma_2, \gamma_3)$:

$$\min_{\mathbf{v}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 + \gamma_1\|\mathbf{v}\|_1 + \gamma_2\,\mathrm{FL}(\mathbf{v}) + \frac{\gamma_3}{2}\|\mathbf{v}\|_2^2, \tag{C.3}$$

the only difference with Friedman et al. (2007) being the addition of the last quadratic term. The method we propose to this problem is a homotopy, which solves $\mathcal{P}(\tau\gamma_1, \tau\gamma_2, \tau\gamma_3)$ for all possible values of $\tau$. In particular, for all $\varepsilon$, it provides the solution of the constrained problem

$$\min_{\mathbf{v}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 \quad \text{s.t.} \quad \gamma_1\|\mathbf{v}\|_1 + \gamma_2\,\mathrm{FL}(\mathbf{v}) + \frac{\gamma_3}{2}\|\mathbf{v}\|_2^2 \leq \varepsilon. \tag{C.4}$$

The algorithm relies on the following lemma

**Algorithm 9** Efficient projection on the elastic-net constraint.

**Require:** $\tau \in \mathbb{R}$; $\gamma \in \mathbb{R}$; $\mathbf{u} \in \mathbb{R}^m$;

**if** $\|\mathbf{u}\|_1 + \frac{\gamma}{2}\|\mathbf{u}\|_2^2 \leq \tau$ **then**
  **return** $\mathbf{v} \leftarrow \mathbf{u}$.
**else**
  $U \leftarrow \{1, \ldots, m\}$; $s \leftarrow 0$; $\rho \leftarrow 0$.
  **while** $U \neq \emptyset$ **do**
    Pick $k \in U$ at random.
    Partition $U$:

$$G = \{j \in U \text{ s.t. } |\mathbf{u}_j| \geq |\mathbf{u}_k|\},$$
$$L = \{j \in U \text{ s.t. } |\mathbf{u}_j| < |\mathbf{u}_k|\}.$$

    $\Delta\rho \leftarrow |G|$; $\Delta s \leftarrow \sum_{j \in G} |\mathbf{u}_j| + \frac{\gamma}{2}|\mathbf{u}_j|^2$.
    **if** $s + \Delta s - (\rho + \Delta\rho)(1 + \frac{\gamma}{2}|\mathbf{u}_k|)|\mathbf{u}_k| < \tau(1 + \gamma|\mathbf{u}_k|)^2$ **then**
      $s \leftarrow s + \Delta s; \rho \leftarrow \Delta\rho; U \leftarrow L$.
    **else**
      $U \leftarrow G \setminus \{k\}$.
    **end if**
  **end while**
  $a \leftarrow \gamma^2\tau + \frac{\gamma}{2}\rho$,
  $b \leftarrow 2\gamma\tau + \rho$,
  $c \leftarrow \tau - s$,
  $\lambda \leftarrow \frac{-b+\sqrt{b^2-4ac}}{2a}$

$$\forall j = 1, \ldots, n, \mathbf{v}_j \leftarrow \frac{\text{sign}(\mathbf{u}_j)(|\mathbf{u}_j| - \lambda)^+}{1 + \lambda\gamma}$$

  **return** $\mathbf{v}$.
**end if**

**Lemma 16**
*Let* $\mathbf{v}^\star(\gamma_1, \gamma_2, \gamma_3)$ *be the solution of Eq. (C.3), for specific values of* $\gamma_1, \gamma_2, \gamma_3$. *Then*

- $\mathbf{v}^\star(\gamma_1, \gamma_2, \gamma_3) = \frac{1}{1+\gamma_3}\mathbf{v}^\star(\gamma_1, \gamma_2, 0)$.
- *For all* $i$, $\mathbf{v}_i^\star(\gamma_1, \gamma_2, 0) = \text{sign}(\mathbf{v}_i^\star(0, \gamma_2, 0)) \max(|\mathbf{v}_i^\star(0, \gamma_2, 0)| - \lambda_1, 0)$—*that is,* $\mathbf{v}^\star(\gamma_1, \gamma_2, 0)$ *can be obtained by soft thresholding of* $\mathbf{v}^\star(0, \gamma_2, 0)$.

The first point can be shown by short calculation. The second one is proven in Friedman et al. (2007) by considering subgradient optimality conditions. This lemma shows that if one knows the solution of $\mathcal{P}(0, \gamma_2, 0)$, then $\mathcal{P}(\gamma_1, \gamma_2, \gamma_3)$ can be obtained in linear time.

It is therefore natural to consider the simplified problem

$$\min_{\mathbf{v}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 + \gamma_2\,\mathrm{FL}(\mathbf{v}). \tag{C.5}$$

With the change of variable $\mathbf{v}_1 = \mathbf{v}_1$ and $\mathbf{v}_i = \mathbf{v}_i - \mathbf{v}_{i-1}$ for $i > 1$, this problem can be recast as a weighted Lasso

$$\min_{\mathbf{v}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{u}-\mathbf{D}\mathbf{v}\|_2^2 + \sum_{i=1}^{m} w_i|\mathbf{v}_i|, \tag{C.6}$$

where $w_1 = 0$ and $w_i = \gamma_2$ for $i > 1$, and $\mathbf{D}_{ij} = 1$ if $i \geq j$ and 0 otherwise. We propose to use LARS (Efron et al., 2004) and exploit the specific structure of the matrix $\mathbf{D}$ to make this approach efficient, by noticing that:

- For a vector $\mathbf{w}$ in $\mathbb{R}^m$, computing $\mathbf{e} = \mathbf{D}\mathbf{w}$ requires $O(m)$ operations instead of $O(m^2)$, by using the recursive formula $\mathbf{e}_1 = \mathbf{w}_1$, $\mathbf{e}_{i+1} = \mathbf{w}_i + \mathbf{e}_i$.
- For a vector $\mathbf{w}$ in $\mathbb{R}^n$, computing $\mathbf{e} = \mathbf{D}^\top\mathbf{w}$ requires $O(m)$ operations instead of $O(m^2)$, by using the recursive formula $\mathbf{e}_n = \mathbf{w}_n$, $\mathbf{e}_{i-1} = \mathbf{w}_{i-1} + \mathbf{e}_i$.
- Let $\mathbf{\Gamma} = \{a_1, \ldots, a_p\}$ be an active set and suppose $a_1 < \ldots < a_p$. Then $(\mathbf{D}_\mathbf{\Gamma}^\top\mathbf{D}_\mathbf{\Gamma})^{-1}$ admits the closed form value

$$(\mathbf{D}_\mathbf{\Gamma}^\top\mathbf{D}_\mathbf{\Gamma})^{-1} = \begin{pmatrix} c_1 & -c_1 & 0 & \ldots & 0 & 0 \\ -c_1 & c_1+c_2 & -c_2 & \ldots & 0 & 0 \\ 0 & -c_2 & c_2+c_3 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & c_{p-2}+c_{p-1} & -c_{p-1} \\ 0 & 0 & 0 & \ldots & -c_{p-1} & c_{p-1}+c_p \end{pmatrix},$$

  where $c_p = \frac{1}{n+1-a_p}$ and $c_i = \frac{1}{a_{i+1}-a_i}$ for $i < p$.

This allows the implementation of this homotopy method without using matrix inversion or Cholesky factorization, solving Eq. (C.6) in $O(ms)$ operations, where $s$ is the number of non-zero values of the optimal solution $\mathbf{v}$.[1]

Adapting this method for solving Eq. (C.4) requires following the regularization path of the problems $\mathcal{P}(0, \tau\gamma_2, 0)$ for all values of $\tau$, which provides as well the regularization path of the problem $\mathcal{P}(\tau\lambda_1, \tau\lambda_2, \tau\lambda_3)$ and stops whenever the constraint becomes unsatisfied. This procedure still requires $O(ms)$ operations.

Note that in the case $\gamma_1 = 0$ and $\gamma_3 = 0$, when only the fused-lasso term is present in Eq (C.3), the same approach has been proposed in a previous work by Harchaoui and Lévy-Leduc (2008), and Harchaoui (2008) to solve Eq. (C.5), with the same tricks for improving the efficiency of the procedure.

[1] To be more precise, $s$ is the number of kinks of the regularization path. In practice, $s$ is roughly the same as the number of non-zero values of the optimal solution $\mathbf{v}$.

# D

# Software

## D.1 SPAMS, a SParse Modeling Software

We present in this section a software package called SPAMS. It implements several optimization methods for sparse methods, including our dictionary learning approach presented in Chapter 2, the LARS algorithm (Efron et al., 2004), a coordinate descent scheme for solving $\ell_1$-decomposition problems (Fu, 1998), a fast implementation of orthogonal matching pursuit, the proximal method of Nesterov (2007), and efficient projection algorithms that are presented in Appendix C. It is adapted for solving a large number of small/medium-scale sparse regularization problems with the square loss.

The software package is coded in C++ with a Matlab interface, and is compatible with Linux, Mac and Windows operating systems. It exploits multi-core CPUs when available. We now present the main functionalities of the software.

### D.1.1 Dictionary Learning and Matrix Factorization

The function `mexTrainDL` is the main component of the toolbox, implementing the learning algorithms of Chapter 2. It addresses the following optimization problem

$$\min_{\mathbf{D}\in\mathcal{D}} \mathbb{E}_{\mathbf{x}}\Big[\min_{\boldsymbol{\alpha}\in\mathcal{A}} \frac{1}{2}\|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda_1\|\boldsymbol{\alpha}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}\|_2^2\Big],$$

where $\mathbf{x}$ is drawn from a potentially infinite training set, $\mathcal{D}$ or $\mathcal{A}$ can be various convex sets as explained in Chapter 2, encoding a priori knowledge on $\mathbf{D}$ and the coefficients $\boldsymbol{\alpha}$, such as nonnegativity constraints. When only a finite training set $\mathbf{X} = [\mathbf{x}^1,\ldots,\mathbf{x}^n]$ in $\mathbb{R}^{m\times n}$ is available, the function addresses

$$\min_{\mathbf{D}\in\mathcal{D}} \frac{1}{n}\sum_{i=1}^{n}\Big[\min_{\boldsymbol{\alpha}^i\in\mathcal{A}} \frac{1}{2}\|\mathbf{x}^i-\mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda_1\|\boldsymbol{\alpha}^i\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}^i\|_2^2\Big],$$

which is a classical dictionary learning formulation. The function admits several computing modes, can use the parameter-free strategy proposed in Chapter 2, or use manually selected parameters $t_0$ and $\rho$. This function does not store the coefficients $\boldsymbol{\alpha}^i$ and does not require the whole training set to be uploaded into memory at the same time. When the problem is reasonably small, A variant of this function called `mexTrainDL_memory` can be used instead. We present an example of usage below:

```
>> I=im2double(imread('lena.png'));
>> % extract all overlapping 8x8 patches
>> X=im2cols(I,[8 8],'sliding');
>> param.lambda=0.15;    % regularization parameter
>> param.K=256;          % size of the dictionary
>> param.batchsize=200;  % size of the minibatch
>> param.iter=1000;      % 1000 iterations over minibatches
>> D=mexTrainDL(X,param);
```

This example (in Matlab) opens the image lena and learns a dictionary with 256 elements, with $\lambda_1 = 0.15$ and $\lambda_2$ is implicitly set to 0. More details can be found in the documentation of the software about the possibilities of this function and its extensions for solving various matrix factorization problems such as non-negative matrix factorization or sparse principal component analysis.

### D.1.2 Orthogonal Matching Pursuit

The function `mexOMP` is a fast implementation of the Orthogonal Matching Pursuit algorithm (see Mallat, 1999). Given a matrix of signals $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ in $\mathbb{R}^{m \times n}$ and a dictionary $\mathbf{D}$ in $\mathbb{R}^{m \times p}$, the algorithm returns a matrix of coefficients $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n]$ in $\mathbb{R}^{p \times n}$ which is an approximate solution of the following sequence of NP-hard problems

$$\forall i \in [\![1; n]\!], \quad \min_{\boldsymbol{\alpha}^i \in \mathbb{R}^p} \|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 \ \text{ s.t. } \ \|\boldsymbol{\alpha}^i\|_0 \leq L.$$

or

$$\forall i \in [\![1; n]\!], \quad \min_{\boldsymbol{\alpha}^i \in \mathbb{R}^p} \|\boldsymbol{\alpha}_i\|_0 \ \text{ s.t. } \ \|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 \leq \varepsilon.$$

For efficiency reasons, the method first computes the covariance matrix $\mathbf{D}^\top\mathbf{D}$, then for each signal $\mathbf{x}^i$, it computes $\mathbf{D}^\top\mathbf{x}^i$ and performs the decomposition with a Cholesky-based algorithm (see Cotter et al., 1999). We again present a Matlab example:

```
>> I=im2double(imread('lena.png'));
>> % extract all overlapping 8x8 patches
>> X=im2cols(I,[8 8],'sliding');
>> % remove the mean value of the patches and normalize them.
>> X=pre_process(X);
>> % upload a dictionary into memory
>> D=load('dictionary.mat');
>> param.eps=0;
>> param.L=10;       % set the sparsity parameter to 10
>> alpha=mexOMP(X,D,param);
```

Such a decomposition of all patches from the image lena can in fact be done efficiently. On a recent 8-cores 2.83Ghz computer, we have measured that our implementation is able to solve 230 000 signals per seconds.

### D.1.3 LARS algorithm

The function `mexLasso` is a fast implementation of the LARS algorithm (Efron et al., 2004) for solving the Lasso (Tibshirani, 1996) or the Elastic-net problems (Zou and Hastie, 2005). As in the previous section, we are given a matrix of signals $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ in $\mathbb{R}^{m \times n}$ and a dictionary $\mathbf{D}$ in $\mathbb{R}^{m \times p}$. The algorithm returns a matrix of coefficients $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n]$ in $\mathbb{R}^{p \times n}$ which is the exact solution of one of the following convex problems

$$\forall i \in [\![1; n]\!], \quad \min_{\boldsymbol{\alpha}^i \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \lambda_1 \|\boldsymbol{\alpha}^i\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}^i\|_2^2. \tag{D.1}$$

or

$$\forall i \in [\![1; n]\!], \quad \min_{\boldsymbol{\alpha}^i \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}^i\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\alpha}^i\|_1 \leq \lambda_1,$$

or

$$\forall i \in [\![1; n]\!], \quad \min_{\boldsymbol{\alpha}^i \in \mathbb{R}^p} \|\boldsymbol{\alpha}^i\|_1 \quad \text{s.t.} \quad \frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i\|_2^2 \leq \lambda_1, \tag{D.2}$$

For efficiency reaons, this implementation uses the same tricks as `mexOMP`, and is based on Cholesky decomposition. It also has an option to handle *nonnegativity constraints*. When the solution is very sparse and the problem size is reasonable, this approach can be very efficient. Moreover, it gives the solution with an exact precision, and its performance does not depend on the correlation of the dictionary elements, except when the solution is not unique. In such a case it is necessary to use a nonzero value for the elastic-net parameter $\lambda_2$. Here is an example in Matlab:

```
>> I=im2double(imread('lena.png'));
>> % extract all overlapping 8x8 patches
>> X=im2cols(I,[8 8],'sliding');
>> % remove the mean value of the patches and normalize them.
>> X=pre_process(X);
>> % upload a dictionary into memory
>> D=load('dictionary.mat');
>> param.lambda=0.15;
>> alpha=mexLasso(X,D,param);
```

On the same 8-cores computer used in the previous section, we have been able to solve 77 000 signal decompositions per second. This number can of course vary with the problem size and the level of regularization (see the benchmark in Section 1.4.5 for instance).

### D.1.4 Coordinate Descent

The function `mexCD` is an implementation a coordinate-descent approach for solving Eq. (D.1) and Eq. (D.2). For Eq. (D.2), the algorithm solves a sequence of problems of the form (D.1) using simple heuristics. Coordinate descent is very simple and in practice

very powerful. It performs well when the correlation between the dictionary elements is small (see the benchmark in Section 1.4.5). Again, we give a simple example

```
>> I=im2double(imread('lena.png'));
>> % extract all overlapping 8x8 patches
>> X=im2cols(I,[8 8],'sliding');
>> % remove the mean value of the patches and normalize them.
>> X=pre_process(X);
>> % upload a dictionary into memory
>> D=load('dictionary.mat');
>> param.lambda=0.15;     % regularization parameter
>> param.tol=1e-2;        % tolerance parameter
>> param.itermax=100;     % maximum number of cycles
>> alpha=mexCD(X,D,param);
```

On the same 8-core computer used in the previous section, we have been able to solve 93 000 signal decompositions per second. This number can of course vary with the problem size, the level of regularization and the amount of correlation among the dictionary elements (see the benchmark in Section 1.4.5).

### D.1.5 Miscellaneous

In addition to the main functions presented above, the toolbox offers other functions

- It implements matrix multiplications on sparse matrices that are sometimes significantly faster than the Matlab ones.
- It implements a conjugate gradient solver.
- It implements the proximal method of Nesterov (2007). However, this function will be removed in future release, since it becomes obsolete with the new software package presented in Section D.2.
- It implements the fast algorithm for performing projections on some convex sets which has been presented in Appendix C.
- Solvers for group-sparsity, (greedy approach for the non-convex setting, and block-coordinate descent algorithm for the convex one), are also implemented, but not yet freely available. They will be released in a next version of the toolbox.

In addition to that, we are planning to make the software open source, and we are welcoming new contributors to develop interfaces for other languages, such as Python[1] or R.[2]

[1] http://www.python.org
[2] http://www.r-project.org

## D.2 Efficient Sparse Solvers with Proximal Methods

The SPAMS software package we have presented in the previous section is well adapted for solving a large number of small and medium-scale sparse decomposition problems with the square loss, which is typical from the classical dictionary learning framework. We now present a new software package that is adapted for solving a wide range of possibly large-scale learning problems, with several combinations of losses and regularization terms. The method implements the proximal methods of Beck and Teboulle (2009), and includes the proximal solvers for the tree-structured regularization of Jenatton et al. (2010a), and the solver of Mairal et al. (2010c) for general structured sparse regularization, which is presented in Chapter 3.

As for SPAMS, the algorithms are implemented in C++ with a Matlab interface, and will be made freely available. The solver for structured sparse regularization norms includes a C++ max-flow implementation of the push-relabel algorithm of Goldberg and Tarjan (1986), with heuristics proposed by Cherkassky and Goldberg (1997).

This implementation also provides robust stopping criteria based on *duality gaps*. It can handle intercepts (unregularized variables). The general formulation that our software can solve take the form

$$\min_{\mathbf{w}\in\mathbb{R}^p} [g(\mathbf{w}) \triangleq f(\mathbf{w}) + \lambda\psi(\mathbf{w})],$$

where $f$ is a smooth loss function and $\psi$ is a regularization function. When one optimizes a matrix $\mathbf{W}$ in $\mathbb{R}^{p\times r}$ instead of a vector $\mathbf{w}$ in $\mathbb{R}^p$, we will write

$$\min_{\mathbf{W}\in\mathbb{R}^{p\times r}} [g(\mathbf{W}) \triangleq f(\mathbf{W}) + \lambda\psi(\mathbf{W})].$$

Note that the software can possibly handle nonnegativity constraints.

We start by presenting the type of regularization implemented in the software

### D.2.1 Regularization Functions

Our software can handle the following regularization functions $\psi$ for vectors $\mathbf{w}$ in $\mathbb{R}^p$:

- **The Tikhonov regularization**: $\psi(\mathbf{w}) \triangleq \frac{1}{2}\|\mathbf{w}\|_2^2$.
- **The $\ell_1$-norm**: $\psi(\mathbf{w}) \triangleq \|\mathbf{w}\|_1$.
- **The tree-structured sum of $\ell_2$-norms**: $\psi(\mathbf{w}) \triangleq \sum_{g\in\mathcal{G}} \eta_g \|\mathbf{w}_g\|_2$, where $\mathcal{G}$ is a tree-structured set of groups, as defined in Chapter 3, and the $\eta_g$ are positive weights.
- **The tree-structured sum of $\ell_\infty$-norms**: $\psi(\mathbf{w}) \triangleq \sum_{g\in\mathcal{G}} \eta_g \|\mathbf{w}_g\|_\infty$. See also Chapter 3 for the exact definition.
- **General sum of $\ell_\infty$-norms**: $\psi(\mathbf{w}) \triangleq \sum_{g\in\mathcal{G}} \eta_g \|\mathbf{w}_g\|_\infty$, where no assumption are made on the groups $\mathcal{G}$.

Our software also handles regularization functions $\psi$ on matrices $\mathbf{W}$ in $\mathbb{R}^{p\times r}$ for multi-task regression problems. In particular,

- **The $\ell_1/\ell_2$-norm**: $\psi(\mathbf{W}) \triangleq \sum_{i=1}^p \|\mathbf{W}_i\|_2$, where $\mathbf{W}_i$ denotes the $i$-th row of $\mathbf{W}$.
- **The $\ell_1/\ell_\infty$-norm**: $\psi(\mathbf{W}) \triangleq \sum_{i=1}^p \|\mathbf{W}_i\|_\infty$,
- **The multi-task tree-structured sum of $\ell_\infty$-norms**:

$$\psi(\mathbf{W}) \triangleq \sum_{i=1}^r \sum_{g\in\mathcal{G}} \eta_g \|\mathbf{w}_g^i\|_\infty + \gamma \sum_{g\in\mathcal{G}} \eta_g \max_{j\in g} \|\mathbf{W}_j\|_\infty, \tag{D.3}$$

  where the first double sums is in fact a sum of independent structured norms on the columns $\mathbf{w}^i$ of $\mathbf{W}$, and the right term is a tree-structured regularization norm applied to the $\ell_\infty$-norm of the rows of $\mathbf{W}$, thereby inducing the tree-structured regularization at the row level. $\mathcal{G}$ is here a tree-structured set of groups.

- **The multi-task general sum of $\ell_\infty$-norms** is the same as Eq. (D.3) except that the groups $\mathcal{G}$ are general overlapping groups.

All of these regularization terms for vectors or matrices can be coupled with nonnegativity constraints. It is also possible to add an intercept, which one wishes not to regularize, and we will include this possibility in the next section. After having presented the regularization terms which our software can handle, we present the various formulations that we address

### D.2.2 Problems Addressed

We present here regression or classification formulations and their multi-task variants.

#### Regression Problems with the Square Loss

Given a training set $\{\mathbf{x}^i, y_i\}_{i=1}^n$, with $\mathbf{x}^i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ for all $i$ in $[\![1; n]\!]$, we address

$$\min_{\mathbf{w}\in\mathbb{R}^p, b\in\mathbb{R}} \frac{1}{n}\sum_{i=1}^n \frac{1}{2}(y_i - \mathbf{w}^\top \mathbf{x}^i - b)^2 + \lambda\psi(\mathbf{w}),$$

where $b$ is an optional variable acting as an "intercept", which is not regularized, and $\psi$ can be any of the regularization functions presented above. Let us consider the vector $\mathbf{y}$ in $\mathbb{R}^n$ that carries the entries $y_i$. The problem without the intercept takes the following form, which we have already encountered in this thesis, but with different notations:

$$\min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}\|\mathbf{y} - \mathbf{X}^\top \mathbf{w}\|_2^2 + \lambda\psi(\mathbf{w}),$$

### Classification Problems with the Logistic Loss

The next formulation that our software can solve is the regularized logistic regression formulation. We are again given a training set $\{\mathbf{x}^i, y_i\}_{i=1}^n$, with $\mathbf{x}^i \in \mathbb{R}^p$, but the variables $y_i$ are now in $\{-1, +1\}$ for all $i$ in $[\![1; n]\!]$. The optimization problem we address is

$$\min_{\mathbf{w}\in\mathbb{R}^p, b\in\mathbb{R}} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i(\mathbf{w}^\top \mathbf{x}^i + b)} + \lambda\psi(\mathbf{w}),$$

with again $\psi$ taken to be one of the regularization function presented above.

### Multi-class Classification Problems with the Softmax Loss

We have also implemented a multi-class logistic classifier (or softmax). For a classification problem with $r$ classes, we are given a training set $\{\mathbf{x}^i, y_i\}_{i=1}^n$, where the variables $\mathbf{x}^i$ are still vectors in $\mathbb{R}^p$, but the $y_i$'s have integer values in $\{1, 2, \ldots, r\}$. The formulation we address is the following multi-class learning problem

$$\min_{\mathbf{W}\in\mathbb{R}^{p\times r}, \mathbf{b}\in\mathbb{R}^r} \frac{1}{n}\sum_{i=1}^{n} \log\Big(\sum_{j=1}^{r} e^{(\mathbf{w}^j - \mathbf{w}^{\mathbf{y}_i})^\top \mathbf{x}^i + \mathbf{b}_j - \mathbf{b}_{\mathbf{y}_i}}\Big) + \lambda\sum_{j=1}^{r}\psi(\mathbf{w}^j), \tag{D.4}$$

where $\mathbf{W} = [\mathbf{w}^1, \ldots, \mathbf{w}^r]$ and the optional vector $\mathbf{b}$ in $\mathbb{R}^r$ carries intercepts for each class.

### Multi-task Regression Problems with the Square Loss

We are now considering a problem with $r$ tasks, and a training set $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^n$, where the variables $\mathbf{x}^i$ are still vectors in $\mathbb{R}^p$, and $\mathbf{y}^i$ is a vector in $\mathbb{R}^r$. We are looking for $r$ regression vectors $\mathbf{w}^j$, for $j \in [\![1; r]\!]$, or equivalently for a matrix $\mathbf{W} = [\mathbf{w}^1, \ldots, \mathbf{w}^r]$ in $\mathbb{R}^{p\times r}$. The formulation we address is the following multi-task regression problem

$$\min_{\mathbf{W}\in\mathbb{R}^{p\times r}, \mathbf{b}\in\mathbb{R}^r} \frac{1}{r}\sum_{j=1}^{r}\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(\mathbf{y}_j^i - \mathbf{w}^\top\mathbf{x}^i - \mathbf{b}_j)^2 + \lambda\psi(\mathbf{W}),$$

where $\psi$ is any of the regularization function on matrices we have presented in the previous section. Note that by introducing the appropriate variables $\mathbf{Y}$, the problem without intercept could be equivalently rewritten

$$\min_{\mathbf{W}\in\mathbb{R}^{p\times r}} \frac{1}{2rn}\|\mathbf{Y} - \mathbf{X}^\top\mathbf{W}\|_{\mathrm{F}}^2 + \lambda\psi(\mathbf{W}).$$

### Multi-task Classification Problems with the Logistic Loss

The multi-task version of the logistic regression follows the same principle. We consider $r$ tasks, and a training set $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^n$, with the $\mathbf{x}^i$'s in $\mathbb{R}^p$, and the $\mathbf{y}^i$'s are vectors in $\{-1, +1\}^r$. We look for a matrix $\mathbf{W} = [\mathbf{w}^1, \ldots, \mathbf{w}^r]$ in $\mathbb{R}^{p\times r}$. The formulation is the following multi-task regression problem

$$\min_{\mathbf{W}\in\mathbb{R}^{p\times r}, \mathbf{b}\in\mathbb{R}^r} \frac{1}{r}\sum_{j=1}^{r}\frac{1}{n}\sum_{i=1}^{n}\log\Big(1 + e^{-\mathbf{y}_j^i(\mathbf{w}^\top\mathbf{x}^i + \mathbf{b}_j)}\Big) + \lambda\psi(\mathbf{W}).$$

### Multi-task and Multi-class Classification Problems with the Softmax Loss

The multi-task/multi-class version directly follows from the formulation of Eq. (D.4), but associates with each class a task, and as a consequence, regularizes the matrix $\mathbf{W}$ in a particular way:

$$\min_{\mathbf{W}\in\mathbb{R}^{p\times r},\mathbf{b}\in\mathbb{R}^r} \frac{1}{n}\sum_{i=1}^{n} \log\Big(\sum_{j=1}^{r} e^{(\mathbf{w}^j-\mathbf{w}^{\mathbf{y}_i})^\top \mathbf{x}^i+\mathbf{b}_j-\mathbf{b}_{\mathbf{y}_i}}\Big) + \lambda\psi(\mathbf{W}).$$

We now move to the computation of duality gaps, which is an important feature of our software.

## D.2.3 Duality Gaps with Fenchel Duality

We are going to use intensively Fenchel Duality, which has been presented in Section 1.4.1. Let us consider again the problem

$$\min_{\mathbf{w}\in\mathbb{R}^p} [g(\mathbf{w}) \triangleq f(\mathbf{w}) + \lambda\psi(\mathbf{w})], \tag{D.5}$$

We first notice that for all the formulations we have been interested in, $g(\mathbf{w})$ can be rewritten

$$g(\mathbf{w}) = \tilde{f}(\mathbf{X}^\top\mathbf{w}) + \lambda\psi(\mathbf{w}), \tag{D.6}$$

where $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ are training vectors, and $\tilde{f}$ is an appropriated smooth real-valued function of $\mathbb{R}^n$, and $\psi$ one of the regularization functions we have introduced.

Given a primal variable $\mathbf{w}$ in $\mathbb{R}^p$ and a dual variable $\boldsymbol{\kappa}$ in $\mathbb{R}^n$, we obtain using classical Fenchel duality rules (Borwein and Lewis, 2006), that the following quantity is a duality gap for problem (D.5):

$$\delta(\mathbf{w}, \boldsymbol{\kappa}) \triangleq g(\mathbf{w}) + \tilde{f}^*(\boldsymbol{\kappa}) + \lambda\psi^*(-\mathbf{X}\boldsymbol{\kappa}/\lambda),$$

where $\tilde{f}^*$ and $\psi^*$ are respectively the Fenchel conjugates of $\tilde{f}$ and $\psi$. Denoting by $\mathbf{w}^\star$ the solution of Eq. (D.5), the duality gap is interesting in the sense that it upperbounds the difference with the optimal value of the function:

$$\delta(\mathbf{w}, \boldsymbol{\kappa}) \geq g(\mathbf{w}) - g(\mathbf{w}^\star) \geq 0.$$

Similarly, we will consider pairs of primal-dual variables $(\mathbf{W}, \mathbf{K})$ when dealing with matrices.

During the optimization, sequences of primal variables $\mathbf{w}$ are available, and one wishes to exploit duality gaps for estimating the difference $g(\mathbf{w}) - g(\mathbf{w}^\star)$. This requires the following components:

- being able to efficiently compute $\tilde{f}^*$ and $\psi^*$.
- being able to obtain a "good" dual variable $\boldsymbol{\kappa}$ given a primal variable $\mathbf{w}$, such that $\delta(\mathbf{w}, \boldsymbol{\kappa})$ is close to $g(\mathbf{w}) - g(\mathbf{w}^\star)$.

We suppose that the first point is satisfied (we will detail these computations for every loss and regularization functions in the sequel), and explain how to choose $\boldsymbol{\kappa}$ in general (details will also be given in the sequel).

Let us first consider the choice that associates with a primal variable $\mathbf{w}$, the dual variable

$$\boldsymbol{\kappa}(\mathbf{w}) \triangleq \nabla \tilde{f}(\mathbf{X}^\top \mathbf{w}), \tag{D.7}$$

and let us compute $\delta(\mathbf{w}, \boldsymbol{\kappa}(\mathbf{w}))$. First, easy computations show that for all vectors $\mathbf{z}$ in $\mathbb{R}^n$, $\tilde{f}^*(\nabla \tilde{f}(\mathbf{z})) = \mathbf{z}^\top \nabla \tilde{f}(\mathbf{z}) - \tilde{f}(\mathbf{z})$, which gives

$$\begin{aligned}\delta(\mathbf{w}, \boldsymbol{\kappa}(\mathbf{w})) &= \tilde{f}(\mathbf{X}^\top \mathbf{w}) + \lambda\psi(\mathbf{w}) + \tilde{f}^*(\nabla \tilde{f}(\mathbf{X}^\top \mathbf{w})) + \lambda\psi^*(-\mathbf{X}\nabla \tilde{f}(\mathbf{X}^\top \mathbf{w})/\lambda), \\ &= \lambda\psi(\mathbf{w}) + \mathbf{w}^\top \mathbf{X}\nabla \tilde{f}(\mathbf{X}^\top \mathbf{w}) + \lambda\psi^*(-\mathbf{X}\nabla \tilde{f}(\mathbf{X}^\top \mathbf{w})/\lambda).\end{aligned}$$

We now use the classical Fenchel-Young inequality (see Borwein and Lewis, 2006, Proposition 3.3.4) on the function $\psi$, which gives

$$\delta(\mathbf{w}, \boldsymbol{\kappa}(\mathbf{w})) \geq \mathbf{w}^\top \mathbf{X}\nabla \tilde{f}(\mathbf{X}^\top \mathbf{w}) - \mathbf{w}^\top \mathbf{X}\nabla \tilde{f}(\mathbf{X}^\top \mathbf{w}) = 0,$$

with equality if and only if $-\mathbf{X}\nabla \tilde{f}(\mathbf{X}^\top \mathbf{w})$ belongs to $\partial\psi(\mathbf{w})$. Interestingly, we now that first-order optimality conditions for Eq. (D.6) gives that $-\mathbf{X}\nabla \tilde{f}(\mathbf{X}^\top \mathbf{w}^\star) \in \partial\psi(\mathbf{w}^\star)$. We have now in hand a non-negative function $\mathbf{w} \mapsto \delta(\mathbf{w}, \boldsymbol{\kappa}(\mathbf{w}))$ of $\mathbf{w}$, that upperbounds $g(\mathbf{w}) - g(\mathbf{w}^\star)$ and satisfying $\delta(\mathbf{w}^\star, \boldsymbol{\kappa}(\mathbf{w}^\star)) = 0$.

This is however not a sufficient property to make it a good measure of the quality of the optimization, and further work is required, that will be dependent on $\tilde{f}$ and $\psi$. We have indeed proven that $\delta(\mathbf{w}^\star, \boldsymbol{\kappa}(\mathbf{w}^\star))$ is always 0. However, for $\mathbf{w}$ different than $\mathbf{w}^\star$, $\delta(\mathbf{w}^\star, \boldsymbol{\kappa}(\mathbf{w}^\star))$ can be infinite, making it a non-informative duality-gap. The reasons for this can be one of the following:

- The term $\psi^*(-\mathbf{X}\nabla \tilde{f}(\mathbf{X}^\top \mathbf{w})/\lambda)$ might have an infinite value.
- Intercepts make the problem more complicated. One can write the formulation with an intercept by adding a row to $\mathbf{X}$ filled with the value 1, add one dimension to the vector $\mathbf{w}$, and consider a regularization function $\psi$ that does regularize the last entry of $\mathbf{w}$. This further complexifies the computation of $\psi^*$ and its definition, as shown in the next section.

Let us now detail how we proceed to solve these problems, but first without considering the intercept. The analysis is similar when working with matrices $\mathbf{W}$ instead of vectors $\mathbf{w}$.

### Duality Gaps without Intercepts

Let us show how to compute the Fenchel conjugate of the functions we have introduced. We now present the Fenchel conjugate of the loss functions $\tilde{f}$.

- **The square loss**

$$\tilde{f}(\mathbf{z}) = \tfrac{1}{2n}\|\mathbf{y}-\mathbf{z}\|_2^2,$$
$$\tilde{f}^*(\boldsymbol{\kappa}) = \tfrac{n}{2}\|\boldsymbol{\kappa}\|_2^2 + \boldsymbol{\kappa}^\top \mathbf{y}.$$

- **The logistic loss**

$$\tilde{f}(\mathbf{z}) = \tfrac{1}{n}\sum_{i=1}^n \log(1+e^{-y_i \mathbf{z}_i})$$
$$\tilde{f}^*(\boldsymbol{\kappa}) = \begin{cases} +\infty \text{ if } \exists\ i \in [\![1;n]\!] \text{ s.t. } y_i\boldsymbol{\kappa}_i \notin [-1,0], \\ \sum_{i=1}^n (1+y_i\boldsymbol{\kappa}_i)\log(1+y_i\boldsymbol{\kappa}_i) - y_i\boldsymbol{\kappa}_i \log(-y_i\boldsymbol{\kappa}_i) \text{ otherwise.} \end{cases}$$

- **The multiclass logistic loss (or softmax)**. The primal variable is now a matrix $\mathbf{Z}$, in $\mathbb{R}^{n\times r}$, which represents the product $\mathbf{X}^\top\mathbf{W}$. We denote by $\mathbf{K}$ the dual variable in $\mathbb{R}^{n\times r}$.

$$\tilde{f}(\mathbf{Z}) = \tfrac{1}{n}\sum_{i=1}^n \log\Big(\sum_{j=1}^r e^{\mathbf{Z}_{ij}-\mathbf{Z}_{i\mathbf{y}_i}}\Big)$$
$$\tilde{f}^*(\mathbf{K}) = \begin{cases} +\infty \text{ if } \exists i \in [\![1;n]\!] \text{ s.t. } \{\mathbf{K}_{ij}<0 \text{ and } j\neq \mathbf{y}_i\} \text{ or } \mathbf{K}_{i\mathbf{y}_i} < -1, \\ \sum_{i=1}^n \Big[\sum_{j\neq \mathbf{y}_i} \mathbf{K}_{ij}\log(\mathbf{K}_{ij}) + (1+\mathbf{K}_{i\mathbf{y}_i})\log(1+\mathbf{K}_{i\mathbf{y}_i})\Big]. \end{cases}$$

Our first remark is that the choice Eq. (D.7) ensures that $\tilde{f}(\boldsymbol{\kappa})$ is not infinite.

As for the regularization function, except for the Tikhonov regularization which is self-conjugate (it is equal to its Fenchel conjugate), we have considered functions that are norms. There exists therefore a norm $\|.\|$ such that $\psi(\mathbf{w}) = \|\mathbf{w}\|$, and we denote by $\|.\|_*$ its dual-norm. In such a case, the Fenchel conjugate of $\psi$ for a vector $\boldsymbol{\gamma}$ in $\mathbb{R}^p$ takes the form

$$\psi^*(\boldsymbol{\gamma}) = \begin{cases} 0 & \text{if } \|\boldsymbol{\gamma}\|_* \leq 1, \\ +\infty & \text{otherwise.} \end{cases}$$

It turns out that for almost all the norms we have presented, there exists (i) either a closed form for the dual-norm or (ii) there exists an efficient algorithm evaluating it. The only one which does not conform to this statement is the tree-structured sum of $\ell_2$-norms, for which we do not know how to evaluate it efficiently.

One can now slightly modify the definition of $\boldsymbol{\kappa}$ to ensure that $\psi^*(-\mathbf{X}\boldsymbol{\kappa}/\lambda) \neq +\infty$. A natural choice is

$$\boldsymbol{\kappa}(\mathbf{w}) \triangleq \min\Big(1, \frac{\lambda}{\|\mathbf{X}\nabla\tilde{f}(\mathbf{X}^\top\mathbf{w})\|_*}\Big)\nabla\tilde{f}(\mathbf{X}^\top\mathbf{w}),$$

which is the one we have implemented. With this new choice, it is easy to see that for all vectors $\mathbf{w}$ in $\mathbb{R}^p$, we still have $\tilde{f}^*(\boldsymbol{\kappa}) \neq +\infty$, and finally, we also have $\delta(\mathbf{w}, \boldsymbol{\kappa}(\mathbf{w})) < +\infty$ and $\delta(\mathbf{w}^\star, \boldsymbol{\kappa}(\mathbf{w}^\star)) = 0$, making it potentially a good duality gap.

### Duality Gaps with Intercepts

Even though adding an intercept does seem a simple modification to the original problem, it induces difficulties for finding good dual variables.

We recall that having an intercept is equivalent to having a problem of the type (D.6), by adding a row to $\mathbf{X}$ filled with the value 1, adding one dimension to the vector $\mathbf{w}$ (or one row for matrices $\mathbf{W}$), and by using a regularization function that does not depend on the last entry of $\mathbf{w}$ (or the last row of $\mathbf{W}$).

Suppose that we are considering a problem of type (D.6) of dimension $p+1$, but we are using a regularization function $\tilde{\psi} : \mathbb{R}^{p+1} \to \mathbb{R}$, such that for a vector $\mathbf{w}$ in $\mathbb{R}^{p+1}$, $\tilde{\psi}(\mathbf{w}) \triangleq \psi(\mathbf{w}_{[\![1;p]\!]})$, where $\psi : \mathbb{R}^p \to \mathbb{R}$ is one of the regularization function we have introduced. Then, considering a primal variable $\mathbf{w}$, a dual variable $\boldsymbol{\kappa}$, and writing $\boldsymbol{\gamma} \triangleq -\mathbf{X}\boldsymbol{\kappa}/\lambda$, we are interested in computing

$$\tilde{\psi}^*(\boldsymbol{\gamma}) = \begin{cases} +\infty \text{ if } \boldsymbol{\gamma}_{p+1} \neq 0 \\ \psi^*(\boldsymbol{\gamma}_{[\![1;p]\!]}) \text{ otherwise,} \end{cases}$$

which means that in order the duality gap not to be infinite, one needs in addition to ensure that $\boldsymbol{\gamma}_{p+1}$ be zero. Since the last row of $\mathbf{X}$ is filled with ones, this writes down $\sum_{i=1}^{p+1} \boldsymbol{\kappa}_i = 0$. For the formulation with matrices $\mathbf{W}$ and $\mathbf{K}$, the constraint is similar but for every column of $\mathbf{K}$.

Let us now detail how we proceed for every loss function to find a "good" dual variable $\boldsymbol{\kappa}$ satisfying this additional constraint, given a primal variable $\mathbf{w}$ in $\mathbb{R}^{p+1}$, we first define the auxiliary function

$$\boldsymbol{\kappa}'(\mathbf{w}) \triangleq \nabla \tilde{f}(\mathbf{X}^\top \mathbf{w}),$$

(which becomes $\mathbf{K}'(\mathbf{W}) \triangleq \nabla \tilde{f}(\mathbf{X}^\top \mathbf{W})$ for matrices), and then define another auxiliary function $\boldsymbol{\kappa}''(\mathbf{w})$ as follows, to take into account the additional constraint $\sum_{i=1}^{p+1} \boldsymbol{\kappa}_i = 0$.

- **For the square loss**, we define another auxiliary function:

  $$\boldsymbol{\kappa}''(\mathbf{w}) \triangleq \boldsymbol{\kappa}'(\mathbf{w}) - \frac{1}{n} \mathbf{1}_{p+1}^\top \boldsymbol{\kappa}'(\mathbf{w}) \mathbf{1}_{p+1}$$

  where $\mathbf{1}_{p+1}$ is a vector of size $p+1$ filled with ones. This step, ensures that $\sum_{i=1}^{p+1} \boldsymbol{\kappa}''(\mathbf{w})_i = 0$.

- **For the logistic loss**, the situation is slightly more complicated since additional constraints are involved in the definition of $\tilde{f}^*$.

  $$\boldsymbol{\kappa}''(\mathbf{w}) \triangleq \operatorname*{arg\,min}_{\boldsymbol{\kappa} \in \mathbb{R}^n} \|\boldsymbol{\kappa} - \boldsymbol{\kappa}'(\mathbf{w})\|_2^2 \quad \text{s.t.} \quad \sum_{i=1}^{n} \boldsymbol{\kappa}_i = 0 \text{ and } \forall i \in [\![1;n]\!],\ \boldsymbol{\kappa}_i \in [-1, 0].$$

  This problem can be solved in linear-time (see Brucker, 1984; Hochbaum and Hong, 1995) using a similar algorithm as for the projection onto the $\ell_1$-ball, since it is an instance of a *quadratic knapsack problem*.

- **For the multi-class logistic loss**, we proceed in a similar way, for every column $\mathbf{K}^j$ of $\mathbf{K}$, $j \in [\![1; r]\!]$:

$$\mathbf{K}''^j(\mathbf{w}) \triangleq \operatorname*{arg\,min}_{\boldsymbol{\kappa} \in \mathbb{R}^n} \|\mathbf{K}'^j - \boldsymbol{\kappa}'(\mathbf{w})\|_2^2 \text{ s.t. } \sum_{i=1}^n \boldsymbol{\kappa}_i = 0 \text{ and}$$
$$\forall i \in [\![1; n]\!], \ \{\boldsymbol{\kappa}_i \geq 0 \text{ if } j \neq \mathbf{y}_i\} \text{ and } \{\boldsymbol{\kappa}_i \geq -1 \text{ if } \mathbf{y}_i = j\}.$$

When the function $\psi$ is the Tykhonov regularization function, we end the process by setting $\boldsymbol{\kappa}(\mathbf{w}) = \boldsymbol{\kappa}''(\mathbf{w})$. When it is a norm, we choose, as before for taking into account the constraint $\|\mathbf{X}\boldsymbol{\kappa}\|_* \leq \lambda$,

$$\boldsymbol{\kappa}(\mathbf{w}) \triangleq \min\Big(1, \frac{\lambda}{\|\mathbf{X}\boldsymbol{\kappa}''(\mathbf{w})\|_*}\Big)\boldsymbol{\kappa}''(\mathbf{w}),$$

with a similar formulation for matrices $\mathbf{W}$ and $\mathbf{K}$.

Even though finding dual variables while taking into account the intercept requires quite a lot of engineering, notably implementing a quadratic knapsack solver, it can be done efficiently.

### D.2.4 Usage

We now present examples of problems that can be solved using our software. The toolbox contains 9 functions, `mexFista*`, `mexIsta*` and `mexProximal*`, where `*` is either (i) `Flat` for all regularization functions that do not involve structured sparsity, (ii) `Tree` for tree-structured regularization functions, including its multi-task version or (iii) `Graph` for a general sum of $\ell_\infty$-norms. `mexFista*` implements the algorithm FISTA of Beck and Teboulle (2009), `mexIsta*` its basic version ISTA, and `mexProximal*` solves a single instance of the proximal problem.

Let us now give three simple examples. First, a Lasso with intercept

```
>> % generate data
>> m=50; p=200;
>> y=randn(m,1);
>> Xt=randn(m,p);
>>
>> param.lambda=0.15; % regularization parameter
>> param.loss=0; % square loss
>> param.regul=0; % lasso
>> param.tol=1e-3; % stopping criterion on relative duality gap
>>
>> % add intercept
>> param.intercept=true;
>> Xt=[Xt ones(m,1)];
>>
```

```
>> w0=zeros(p+1,1);
>> w=mexFistaFlat(y,Xt,w0,param);
```

Then, a more complicated example, with a multi-task $\ell_1/\ell_2$ norm on matrices, without intercept

```
>> % generate data
>> m=50; p=200; r=10;
>> Y=randn(m,r);
>> Xt=randn(m,p);
>>
>> param.lambda=0.15; % regularization parameter
>> param.loss=0; % square loss
>> param.regul=4; % l1/l2 norm
>> param.tol=1e-3; % stopping criterion on relative duality gap
>>
>> W0=zeros(p,r);
>> W=mexFistaFlat(Y,Xt,W0,param);
```

Our last example involves solving the proximal operator corresponding to a tree-structured sparse regularization with sums of $\ell_\infty$-norms.

```
>> % generate data
>> m=50; p=200;
>> y=randn(m,1);
>>
>> param.lambda=0.15; % regularization parameter
>> param.loss=0; % square loss
>> param.regul=2; % tree-structured regularization
>> tree=load('tree.mat');  % load a tree structure
>>
>> u=mexProximalTree(y,tree,param);
```

For other possibilities, especially structured-sparsity solvers or the use of other loss functions, more details will be included in the official documentation of the software package.

# Bibliography

A. Agarwal and B. Triggs. Hyperfeatures - multilevel local coding for visual recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Graz, Austria, May 2006.

M. Aharon and M. Elad. Sparse and redundant modeling of image content using an image-signature-dictionary. *SIAM Journal on Imaging Sciences*, 1(3):228–247, July 2008.

M. Aharon, M. Elad, and A. M. Bruckstein. The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, November 2006.

N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transcations on Computers*, C-23(1):90–93, 1974.

R. K. Ahuja, T. L. Magnanti, and J. Orlin. *Network Flows*. Prentice Hall, 1993.

P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *Proceedings POCV*, 2006.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

S. P. Awate and R. T. Whitaker. Unsupervised, information-theoretic, adaptive image filtering for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 364–376, 2006.

M. Babenko and A. V. Goldberg. Experimental evaluation of a parametric flow algorithm. Technical report, Microsoft Research, 2006. MSR-TR-2006-77.

F. Bach. Consistency of the group Lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1224, 2008.

F. Bach. High-dimensional non-linear variable selection through hierarchical kernel learning. Technical report, arXiv:0909.0844, 2009.

F. Bach, J. Mairal, and J. Ponce. Convex sparse matrix factorizations. Technical report, 2008. Preprint arXiv:0812.1869.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. The MIT Press, 2011. to appear.

R. G. Baraniuk, V. Cevher, M. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 2010. to appear.

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.

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009.

M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. Comp. Graph. and Interact. Tech.*, pages 417–424, 2000.

D. P. Bertsekas. *Nonlinear programming*. Athena Scientific Belmont, 1999.

D. P. Bertsekas. *Linear Network Optimization*. MIT Press, 1991.

P. Bickel, Y. Ritov, and A. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Annals of statistics*, 37(4):1705–1732, 2009.

D. Blei and J. McAuliffe. Supervised topic models. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 121–128. MIT Press, 2008.

D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

D. Blei, T. Griffiths, and M. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):1–30, 2010.

J. F. Bonnans and A. Shapiro. Optimization problems with perturbations: A guided tour. *SIAM Review*, 40(2):202–227, 1998.

J. F. Bonnans and A. Shapiro. *Perturbation analysis of optimization problems*. Springer, 2000.

J. M. Borwein and A. S. Lewis. *Convex analysis and nonlinear optimization: Theory and examples*. Springer, 2006.

L. Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. 1998.

L. Bottou and O. Bousquet. The trade-offs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. MIT Press, 2008.

Y-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12): 1222–1239, November 2001.

D. M. Bradley and J. A. Bagnell. Differentiable sparse coding. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21, pages 113–120. 2009.

P. Brucker. An O(n) algorithm for quadratic knapsack problems. *Oper. Res. Lett.*, 3: 163–166, 1984.

A. Buades, B. Coll, and J.M. Morel. A review of image denoising algorithms, with a new one. *SIAM Multiscale Modelling and Simulation*, 4(2):490, 2005.

A. Buades, B. Coll, J-M. Morel, and C. Sbert. Self-similarity driven demosaicking. *IEEE Transactions on Image Processing*, 18(6):1192–1202, 2009.

W. L. Buntine. Variational extensions to em and multinomial pca. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2002.

E. Candes. Compressive sampling. In *Proceedings of the International Congress of Mathematicians*, volume 3, 2006.

E. Candes and D. L. Donoho. New tight frames of curvelets and the problem of approximating piecewise $\mathbf{C}^2$ images with piecewise $\mathbf{C}^2$ edges. *Comm. Pure Appl. Math.*, 57: 219–266, February 2004.

E. Candes and D. L. Donoho. Recovering edges in ill-posed inverse problems: Optimality of curvelet frames. *Annals of statistics*, 30(3):784–842, June 2002.

E. J. Candès, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier Analysis and Applications*, 14:877–905, 2008.

E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 2010.

J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

V. Cevher, M. F. Duarte, C. Hegde, and R. Baraniuk. Sparse signal recovery using markov random fields. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 257–264. 2009.

S. G. Chang, B. Yu, and M. Vetterli. Adaptive wavelet thresholding for image denoising and compression. *IEEE Transactions on Image Processing*, 9(9):1532–1546, 2000.

P. Chatterjee and P. Milanfar. Clustering-based denoising with locally learned dictionaries. *IEEE Transactions on Image Processing*, 18(7):1438–1451, 2009.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.

K. Chin, S. DeVries, J. Fridlyand, P.T. Spellman, R. Roydasgupta, W. L. Kuo, A. Lapuk, R. M. Neve, Z. Qian, T. Ryder, et al. Genomic and transcriptional aberrations linked to breast cancer pathophysiologies. *Cancer Cell*, 10(6):529–541, 2006.

R. R. Coifman and D. L. Donoho. Translation-invariant de-noising. *Lectures notes in statistics*, pages 125–125, 1995.

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Springer, 2010.

S. F. Cotter, J. Adler, B. Rao, and K. Kreutz-Delgado. Forward sequential algorithms for best basis selection. In *IEEE Proceedings of Vision Image and Signal Processing*, pages 235–244, 1999.

S. F. Cotter, B. D. Rao, K. Engan, and K. Kreutz-Delgado. Sparse solutions to linear inverse problems with multiple measurement vectors. *IEEE Transactions on Signal Processing*, 53(7):2477–2488, 2005.

A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212, 2004.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Inverse halftoning by pointwise shape-adaptive DCT regularized deconvolution. In *Proceedings of the International TICSP Workshop on Spectral Methods Multirate Signal Processing (SMMSP)*, 2006.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image restoration by sparse 3d transform-domain collaborative filtering. In *Proceedings of SPIE Electronic Imaging*, 2008.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

J. M. Danskin. The theory of max-min, and its application to weapons allocation problems. *Ökonometrie und Unternehmensforschung*, 1967.

A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007.

A. d'Aspremont, F. Bach, and L. El Ghaoui. Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9:1269–1294, 2008.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math*, 57:1413–1457, 2004.

M. Do and M. Vetterli. Framing pyramids. *IEEE Transactions on Signal Processing*, 51 (9):2329–2342, 2003a.

M. Do and M. Vetterli. *Contourlets, Beyond Wavelets*. Academic Press, New York, 2003b.

P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

D. L. Donoho. Wedgelets: Nearly minimax estimation of edges. *Annals of statistics*, 27 (3):859–897, June 1998.

D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4): 1289–1306, April 2006.

D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995.

M.F. Duarte, M.A. Davenport, D. Takhar, J.N. Laska, T. Sun, K.F. Kelly, and R.G. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, 2008.

M. Duarte-Carvajalino and G. Sapiro. Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. *IEEE Transactions on Image Processing*, 18(7):1395–1408, 2009.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the $\ell_1$-ball for learning in high dimensions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of statistics*, 32(2):407–499, 2004.

A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1999.

M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 54(12):3736–3745, December 2006.

M. Elad, J-L Starck, P. Querre, and D. L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (mca). *Applied and Computational Harmonic Analysis*, 19:340–358, November 2005.

K. Engan, S. O. Aase, and J. H. Husoy. Frame based signal compression using method of optimal directions (MOD). In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1999.

R. Eslami and H. Radha. Translation-invariant contourlet transform and its application to image denoising. *IEEE Transactions on Image Processing*, 15(11):3362–3374, 2006.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results, 2006.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007.

C. Févotte, N. Bertin, and J. L. Durrieu. Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural Computation*, 21 (3):793–830, 2009.

D. L. Fisk. Quasi-martingales. *Transactions of the American Mathematical Society*, 120 (3):359–388, 1965.

R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial grey scale. In *Proceedings of the Society of Information Display*, volume 17, pages 75–77, 1976.

A. Foi, V. Katkovnik, K. Egiazarian, and J. Astola. Inverse halftoning based on the anisotropic lpa-ici deconvolution. In *Proceedings of Int. TICSP Workshop Spectral Meth. Multirate Signal Process.*, 2004.

A. Foi, K. Dabov, V. Katkovnik, and K. Egiazarian. Shape-adaptive DCT for denoising and image reconstruction. In *Proceedings of SPIE Electronic Imaging*, 2006.

L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian J. Math.*, 8(3):399–404, 1956.

W. T. Freeman and E. H. Adelson. The design and the use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

J. Friedman, T. Hastie, H. Hölfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

W. J. Fu. Penalized regressions: The bridge versus the Lasso. *Journal of Computational and Graphical Statistics*, 7:397–416, 1998.

J. J. Fuchs. Recovery of exact sparse representations in the presence of bounded noise. *IEEE Transactions on Information Theory*, 51(10):3601–3608, 2005.

G. Gallo, M. E. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal of Computing*, 18:30–55, 1989.

P. Garrigues and B. Olshausen. Learning horizontal connections in a sparse coding model of natural images. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 505–512. MIT Press, Cambridge, MA, 2008.

G. Gasso, A. Rakotomamonjy, and S. Canu. Recovering sparse signals with non-convex penalties and dc programming. *IEEE Transactions on Signal Processing*, 57(12):4686–4698, 2009.

A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. In *Proc. of ACM Symposium on Theory of Computing*, pages 136–146, 1986.

G. H. Golub and C. F. Van Loan. *Matrix computations*. John Hopkins University Press, 1996.

R. Grosse, R. Raina, H. Kwong, and A. Y. Ng. Shift-invariant sparse coding for audio classification. In *Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

J. A. Guerrero-Colon, L. Mancera, and J. Portilla. Image restoration using space-variant gaussian scale mixtures in overcomplete pyramids. *IEEE Transactions on Image Processing*, 17(1):27–41, 2008.

B. K. Gunturk, Y. Altunbasak, and R. M. Mersereau. Color plane interpolation using alternating projections. *IEEE Transactions on Image Processing*, 11(9):997–1013, 2002.

B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau. Demosaicking: color filter array interpolation. *IEEE Sig. Proc. Mag.*, 22(1):44–54, 2005.

B. Haasdonk and D. Keysers. Tangent distance kernels for support vector machines. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2002.

E. T. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing. Technical report, Rice University„ 2007. CAAM Technical Report TR07-07.

Z. Harchaoui. *Méthodes à Noyaux pour la Détection*. PhD thesis, Télécom ParisTech, 2008.

Z. Harchaoui and C. Lévy-Leduc. Catching change-points with Lasso. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. MIT Press, 2008.

T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2009. 2nd Edition.

D. S. Hochbaum and S. P. Hong. About strongly polynomial time algorithms for quadratic optimization over submodular constraints. *Math. Program.*, 69(1):269–309, 1995.

A. E. Hoerl and R. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–377, 1936.

P. O. Hoyer. Non-negative sparse coding. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, 2002.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

P. O. Hoyer and A. Hyvärinen. Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, 11(3):191–210, 2000.

J. Huang, Z. Zhang, and D. Metaxas. Learning with structured sparsity. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

K. Huang and S. Aviyente. Sparse representation for signal classification. In *Advances in Neural Information Processing Systems*, 2006.

J. M. Hugues, D. J. Graham, and D. N. Rockmore. Quantification of artistic style through sparse coding analysis in the drawings of Pieter Bruegel the Elder. *Proceedings of the National Academy of Science*, 107(4):1279–1283, 2009.

L. Jacob, G. Obozinski, and J.-P. Vert. Group Lasso with overlap and graph Lasso. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

R. Jenatton, J-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. Technical report, 2009. Preprint arXiv:0904.3523v2.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010a.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. Technical report, 2010b. submitted, arXiv:1009.2139v2.

R. Jenatton, G. Obozinski, and F. Bach. Structured sparse principal component analysis. In *Proceedings of the Conference on AI and Statistics (AISTATS)*, 2010c.

C. R. Johnson, E. H. Hendriks, I. J. Berezhnoy, E. Brevdo, S. M. Hugues, I. Daubechies, J. Li, E. Postma, and J. Z. Want. Image processing for artist identification. *IEEE Signal Processing Magazine*, (37), July 2008.

N. Jojic, B. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2003.

I. T. Jolliffe, N. T. Trendafilov, and M. Uddin. A modified principal component technique based on the Lasso. *Journal of Computational and Graphical Statistics*, 12(3):531–547, 2003.

K. Kavukcuoglu, M. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU, 2008.

K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

C. Kervrann and J. Boulanger. Local adaptivity to variable smoothness for exemplar-based image denoising and representation. *International Journal of Computer Vision*, 79(1):45–69, 2008.

S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

T. D. Kite, N. Damera-Venkata, B. L. Evans, and A. C. Bovik. A fast, high-quality inverse halftoning algorithm for error diffused halftones. *IEEE Transactions on Image Processing*, 9(9):1583–1592, 2000.

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, February 2004.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

LSG Kovasznay and HM Joseph. Image processing. *Proceedings of the IRE*, 43(5): 560–570, 1955.

H. J. Kushner and G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2003.

H. Larochelle and Y. Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007. submitted.

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In David Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan Kaufman, 1990.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998a.

Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998b.

Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F-J. Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001.

H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 801–808. MIT Press, 2007.

H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

K. C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):684–698, 2005.

T. W. Lee and M. S. Lewicki. Unsupervised image classification, segmentation, and enhancement using ica mixture models. *IEEE Transactions on Image Processing*, 11 (3), 2002.

M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, 2000.

Y. Li and D. P. Huttenlocher. Sparse long-range random field and its application to image denoising. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008.

A. Di Lillo, G. Motta, and J. A. Storer. Texture classification based on discriminative features extracted in the frequency domain. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2007.

C. J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. In *Uncertainty in Artificial Intelligence*, 2009.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

A. C. Lozano, G. Świrszcz, and N. Abe. Group orthogonal matching pursuit for variable selection and prediction. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, pages 1150–1158. 2009.

S Lyu, D. N Rockmore, and H. Farid. A digital technique for art authentication. *Proceedings of the National Academy of Science*, 101(49):17006–17010, 2004.

N. Maculan and J. R. G. Galdino de Paula. A linear-time median-finding algorithm for projecting a vector on the simplex of Rn. *Operations Research Letters*, 8(4):219–222, 1989.

T. M. Mäenpää, M. Pietikäinen, and T. Ojala. Texture classification by multi-predicate local binary pattern operators. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2000.

J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics, revised edition*. John Wiley, Chichester, 1999.

J. Mairal, G. Sapiro, and M. Elad. Multiscale sparse image representation with learned dictionaries. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2007.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008a.

J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, January 2008b.

J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008c.

J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. *SIAM Multiscale Modelling and Simulation*, 7(1):214–241, April 2008d.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009a.

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, *Advances in Neural Information Processing Systems*, volume 21, pages 1033–1040. MIT Press, 2009b.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009c.

J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. Technical report, 2010a. submitted, arXiv:1009.5359v1.

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, 2010b.

J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In *Advances in Neural Information Processing Systems*, 2010c.

S. Mallat. *A Wavelet Tour of Signal Processing, Second Edition*. Academic Press, New York, September 1999.

S. Mallat and E. Le Pennec. Sparse geometric image representation with bandelets. *IEEE Transactions on Image Processing*, 14(4):423–438, 2005a.

S. Mallat and E. Le Pennec. Bandelet image approximation and compression. *SIAM Multiscale Modelling and Simulation*, 4(3):992–1039, 2005b.

S. Mallat and G. Peyré. Orthogonal bandlet bases for geometric images approximation. *Communication on Pure and Applied Mathematics*, 61(9):1173–1212, 2008.

S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2001.

D. R. Martin, C. C Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1), january 2004.

B. Matalon, M. Elad, and M. Zibulevsky. Improved denoising of images using modeling of the redundant contourlet transform. In *Proceedings of the SPIE conference wavelets*, 2005.

M. Métivier. *Semi-martingales*. Walter de Gruyter, 1983.

J. J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *C. R. Acad. Sci. Paris Sér. A Math.*, 255:2897–2899, 1962.

N. Murata. Statistical study on on-line learning. *On-line learning in neural networks*, pages 63–92, 1999.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 89:355–368, 1998.

R. Neelamani, R.D. Nowak, and R.G. Baraniuk. WInHD: Wavelet-based inverse halftoning via deconvolution. *Rejecta Mathematica*, 1(1):84–103, 2009.

S. Negahban, P. Ravikumar, M. J. Wainwright, and B. Yu. A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers. In *Advances in Neural Information Processing Systems*, 2009.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, 2007.

Y. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269:543–547, 1983. in russian.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

G. Obozinski, M. J. Wainwright, and M. I. Jordan. Union support recovery in high-dimensional multivariate regression. *UC Berkeley Technical Report 761*, August 2008.

G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 2009. Published online.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

A. Opelt and A. Pinz. Object localization with boosting and weak supervision for generic object recognition. In *Proceedings of the Scandinavian Conference on Image Analysis (SCIA)*, 2005.

M. R. Osborne, B. Presnell, and B. A. Turlach. On the Lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–37, 2000a.

M. R. Osborne, B. Presnell, and B. A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000b.

D. Paliy, V. Katkovnik, R. Bilcu, S. Alenius, and K. Egiazarian. Spatially adaptive color filter array interpolation for noiseless and noisy data. *Intern. J. of Imaging Sys. and Tech.*, 17(3), 2007.

C. Pantofaru, G. Dorkó, C. Schmid, and M. Hebert. Combining regions and patches for object class localization. In *Proceedings of the "Beyond Patches workshop", CVPR*, 2006.

P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

G. Peyré. Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, 34(1):17–31, May 2009.

J. Portilla, V. Strela, MJ Wainwright, and EP Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2003.

M. Prasad, A. Zisserman, A. Fitzgibbon, M. Pawan Kumar, and P.H.S. Torr. Learning class-specifc edges for object detection and segmentation. In *Proceedings ICVGIP*, 2006.

M. Protter and M. Elad. Image sequence denoising via sparse and redundant representations. *IEEE Transactions on Image Processing*, 18(1):27–36, 2009.

R. C. Puetter, T. R. Gosnell, and A. Yahil. Digital image reconstruction: deblurring and denoising. *Annu. Rev. Astron. Astrophys.*, 43, 2005.

A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An efficient projection for $\ell_{1,\infty}$ regularization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

T. Randen and J. H. Husoy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4), April 1999.

M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007a.

M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 1137–1144. MIT Press, 2007b.

X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2005.

S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

V. Roth and B. Fischer. The Group-Lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

L. I. Rudin and S. Osher. Total variation based image restoration with free local constraints. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 1994.

J. Salmon and E. Le Pennec. Nl-means and aggregation procedures. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2009.

M. W. Seeger. Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9:759–813, 2008.

S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Stochastic convex optimization. In *22nd Annual Conference on Learning Theory (COLT)*, 2009.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. 2004.

E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multi-scale transforms. *IEEE Transactions on Information Theory*, 38(2):587–607, September 1992.

K. Skretting and J. H. Husoy. Texture classification using sparse frame-based representations. *EURASIP J. Appl. Signal Process.*, (1), 2006.

P. Sprechmann, I. Ramirez, G. Sapiro, and Y. C. Eldar. Collaborative hierarchical sparse modeling. Technical report, 2010. Preprint arXiv:1003.0400v1.

J-L. Starck, E. Candes, and D. L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 11(6):670–684, 2002.

K.-K. Sung. *Learning and Example Selection for Object and Pattern Recognition*. PhD thesis, MIT, Artificial Intelligence Laboratory and Center for Biological and Computational Learning, 1996.

A. Szlam, M. Maggioni, and R. R. Coifman. Regularization on graphs with function-adapted diffusion processes. *Journal of Machine Learning Research*, 2007.

H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. *IEEE Transactions on Image Processing*, 16(2):349–366, 2007.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B*, 58(1):267–288, 1996.

R. Tibshirani and P. Wang. Spatial smoothing and hot spot detection for CGH data using the fused Lasso. *Biostatistics*, 9(1):18–29, 2008.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, 67(1):91–108, 2005.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of ill-posed problems*. V. H. Winston and Sons, 1977.

J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, October 2004.

J. A. Tropp. Algorithms for simultaneous sparse approximation. part ii: Convex relaxation. *Signal Processing, special issue "Sparse approximations in signal and image processing"*, 86:589–602, April 2006.

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, April 2006.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

B. A. Turlach, W. N. Venables, and S. J. Wright. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.

T. Tuytelaars and C. Schmid. Vector quantizing feature space with a regular lattice. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2007.

A. W. Van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

C. Wallraven, B. Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Nice, France, 2003.

S. Weisberg. *Applied Linear Regression*. Wiley, 1980.

Y. Weiss and W. T. Freeman. What makes a good model of natural images ? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, USA, June 2007.

Y. Weiss, H. Chang, and W. Freeman. Learning compressed sensing. In *Snowbird Learning Workshop, Allerton, CA*, 2007.

J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2005.

D. M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009a.

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*, 2010.

S. J. Wright, R. D. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009b.

T. T. Wu and K. Lange. Coordinate descent algorithms for Lasso penalized regression. *Annals of Applied Statistics*, 2(1):224–244, 2008.

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 2010. to appear.

G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. Technical report, 2010. preprint arXiv:1006.3056v1.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B*, 68:49–67, 2006.

R. Zass and A. Shashua. Nonnegative sparse PCA. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 1561–1568. MIT Press, 2007.

H. H. Zhang, Y. Liu, Y. Wu, and J. Zhu. Selection for the multicategory svm via adaptive sup-norm regularization. *Electronic Journal of Statistics*, 2:149–167, 2008.

L. Zhang and X. Wu. Color demosaicking via directional linear minimum mean square-error estimation. *IEEE Transactions on Image Processing*, 14(12):2167–2178, 2005.

P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. 37(6A):3468–3497, 2009.

S. C. Zhu and D. Mumford. Prior learning and gibbs reaction-diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1236–1250, 1997.

M. Zibulevsky and B. A. Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation*, 13(4):863–882, 2001.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*, 67(2):301–320, 2005.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.