Structured sparsity with convex penalty functions

Jean Manuel Morales

First Supervisor: Massimiliano Pontil

Second Supervisor: Mark Herbster

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University of London.

Department of Computer Science

University College London

May 28, 2012

Abstract

We study the problem of learning a sparse linear regression vector under additional conditions on the structure of its sparsity pattern. This problem is relevant in Machine Learning, Statistics and Signal Processing. It is well known that a linear regression can benefit from knowledge that the underlying regression vector is sparse. The combinatorial problem of selecting the nonzero components of this vector can be "relaxed" by regularising the squared error with a convex penalty function like the ℓ_1 norm. However, in many applications, additional conditions on the structure of the regression vector and its sparsity pattern are available. Incorporating this information into the learning method may lead to a significant decrease of the estimation error.

In this thesis, we present a family of convex penalty functions, which encode prior knowledge on the structure of the vector formed by the absolute values of the regression coefficients. This family subsumes the ℓ_1 norm and is flexible enough to include different models of sparsity patterns, which are of practical and theoretical importance. We establish several properties of these penalty functions and discuss some examples where they can be computed explicitly. Moreover, for solving the regularised least squares problem with these penalty functions, we present a convergent optimisation algorithm and proximal method. Both algorithms are useful numerical techniques taylored for different kinds of penalties.

Extensive numerical simulations highlight the benefit of structured sparsity and the advantage offered by our approach over the Lasso method and other related methods, such as using other convex optimisation penalties or greedy methods.

Acknowledgements

I am greatly indebted to my supervisor, Prof. Massimiliano (Massi) Pontil. He always proved to be very generous, and he gave me the wonderful opportunity to come to University College London to study Machine Learning, and to be involved in his research activity. He believed I could get a Ph.D. even after I spent three years working in the “real world”, corrupting my brain. He has always been very encouraging and supportive, both for my research and for my life in London. It was interesting as well as fun to work with him, and I had a stimulating and truly enjoyable experience.

I am grateful to my second supervisor, Mark Herbster. I benefited from graduate lectures by him I could attend early in my studies, and from being a teaching assistant a few years later. Most importantly, I had the opportunity of working with him in a research project. During that period he was pleasant and helpful, and made me understand his good ideas in simple terms, always with a graphical intuition. I learned a lot from him.

I would like to thank Charles Micchelli who collaborated with my supervisor and me in what would become some of the key results of this thesis. He has impressive knowledge and an evident passion for what he does. It is inspiring to work with him, and he provided me with invaluable suggestions.

I thank my examiners, Serge Guillas and Tijl De Bie, for their interest in this work and for their useful comments.

My gratitude also goes to Andreas Argyriou for many practical advise he offered me. He participated in a paper we wrote towards the end of my Ph.D., but he was here at UCL when I arrived and we shared a large number of discussions. He helped me a lot, and I admired his excellent research skills.

I am especially thankful to Luca Baldassarre for sharing his energy and enthusiasm. We worked a lot together, and I could discuss with him many technical aspects of the experiments that were an essential part of our research.

I am also indebted to the many students I met during these years. I had the fortune to learn and to grow in a lively and entertaining environment. Among all, I wish to single out Lorella

Campanale, for her brief, yet unmistakably high-energy, high-impact visit here in London.

I wish to thank also the people in Italy who encouraged and helped me to study abroad. In particular I am very thankful to Prof. Pietro Terna and to Prof. Hisao Fujita-Yashima who were kind enough to write a reference letter on a clearly too short a notice.

Thanks to Stefania, who really helped me a lot, *malgré tout*.

Thanks to Miriam and to Marisa, for making such a diverse and jolly company. It is impossible for me not to laugh with joy when I am with you.

Finally, I wish to thank my parents who raised me, loved me, supported me and encouraged me to pursue my dreams even if that meant they had to miss me during these years. I love you both dearly.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Machine Learning provides a set of techniques used to automatically analyse huge amount of data. Two of the most common goals for this analysis are description and prediction, for which we distinguish between Unsupervised Learning and Supervised Learning.

In Unsupervised Learning, the aim is to highlight structures of the data. One common method is to cluster together unlabelled points, so as to emphasise their similarity. Another common method is the analisys of the principal components of the data, useful both for assessing the main factors and for dimensionality reduction.

In Supervised Learning, the data is labelled and the aim is to build a model for the relationship between labels and data. Then, the model can be used to predict the labels of new data. As this thesis will focus on the supervised setting, we describe the objects of the analisys. The observations forming the data will be elements in a set $\mathcal{X}$, and each element will come with a label belonging to a set $\mathcal{Y}$. The training set is the collection of m pairs of observations and their labels, or $\{(x_i, y_i)\}_{i=1}^m \subseteq \mathcal{X} \times \mathcal{Y}$. We assume that the elements of this set are drawn randomly, independently and identically distributed, from the space $\mathcal{X} \times \mathcal{Y}$. This set will be used by the algorithm to *learn* the model.

As the input set $\mathcal{X}$, we will consider the Euclidean space $\mathbb{R}^n$. This is already general enough to include most of the common varieties of data, such as for instance time series, texts and images. As the output set $\mathcal{Y}$, we will consider the real line $\mathbb{R}$, so that we are performing a *regression* of the data. This is opposed to a *classification* of the data, which is the case when the set is finite (binary or multiple classes classification).

The object of the learning will be the determination of a predicting function $f : \mathcal{X} \to \mathcal{Y}$ belonging to a predefined class of functions $\mathcal{F}$ (in our case, the class of real valued functions). This function will have to encode the relationship between the input and the output: if the complexity of the function, the dimensionality of the data and the number of training points permit it, the function f will map exactly all the observations in the training set with their

respective labels. We refer to this situation as Interpolation.

Interpolation of data is not always possible nor desirable, because a function that interpolates the points of the training set will often perform poorly for prediction. When the function does not interpolate, there will be a difference between the prediction $f(x)$ and the actual value y for at least some points in the training set. This difference is usually measured with a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, which can simply be the squared difference $(y - f(x))^2$.

The Risk associated to any function f depends on the loss function, which is a design choice made using knowledge of the data, and on the joint distribution $\mathcal{D}$ of the data and the labels. The risk is defined as

$$R(f) = \int_{\mathcal{X}\times\mathcal{Y}} L(y, f(x)) d\mathcal{D}.$$

We want to find a function that minimises this quantity but, as $\mathcal{D}$ is unknown, this is usually impossible. An approximation to the risk minimisation is given by the empirical risk

$$R_{\text{emp}}(f) = \frac{1}{m} \sum_{i=1}^{m} L(y_i, f(x_i)),$$

which will be the object function of our minimisation problem.

If the data can be interpolated, there will be an infinite number of functions such that $R_{\text{emp}}(f) = 0$. To have a unique solution one common technique is Regularisation: we minimise the sum of the loss function and a penalty term P weighted with a coefficient $\rho > 0$. The learned function will be

$$\hat{f} = \underset{f\in\mathcal{F}}{\operatorname{argmin}} \left\{ R_{\text{emp}}(f) + \rho P(f) \right\}.$$

The penalty term will treat different functions in different ways, so for instance it can be used to penalise complex functions more: as a result, the solution function will tend to be simpler. Moreover, if P is strictly convex, the solution to the problem is unique.

We will apply regularisation to solve the problem of sparse estimation. This problem is becoming increasingly important in machine learning, as well as in statistics and signal processing, and consists in finding a sparse solution, that is one with few nonzero parameters. In its simplest form, we consider linear functions $f(x) = \sum_i \beta_i x_i$, that are completely defined by a coefficient vector $\beta \in \mathbb{R}^n$ (so we can consider both L and P as functions of β). The problem in this form consists in estimating the regression vector $\beta^* \in \mathbb{R}^n$ from a set of linear measurements $y \in \mathbb{R}^m$, obtained from the model

$$y = X\beta^* + \xi,$$

where X is an $m \times n$ matrix, which may be fixed or randomly chosen and $\xi \in \mathbb{R}^m$ is a vector which results from the presence of noise.

An important rationale for sparse estimation comes from the observation that in many practical applications the number of parameters n is much larger than the data size m, but the vector β^* is known to be sparse, that is, most of its components are equal to zero. Under this sparsity assumption and certain conditions on the data matrix X, it has been shown that regularization with the ℓ_1 norm, commonly referred to as the Lasso method[1] [49] , provides an effective means to estimate the underlying regression vector, see for example [7, 11, 28, 52] and references therein. Moreover, this method can reliably select the sparsity pattern of β^* [28], hence providing a valuable tool for feature selection.

In this thesis, we are interested in sparse estimation under additional conditions on the sparsity pattern of the vector β^*. In other words, not only do we expect this vector to be sparse but also that it is *structured sparse*, namely certain configurations of its nonzero components are to be preferred to others. The motivation for favouring structured sparsity arises in several applications, ranging from functional magnetic resonance imaging [16, 54], to scene recognition in vision [17], to multi-task learning [1, 25, 37] and to bioinformatics [44], see [24] for a discussion.

The prior knowledge that we consider in this thesis is that the vector $|\beta^*|$, whose components are the absolute value of the corresponding components of β^*, should belong to some prescribed convex subset Λ of the positive orthant. For certain choices of Λ this implies a constraint on the sparsity pattern as well. For example, the set Λ may include vectors with some desired monotonicity constraints, or other constraints on the "shape" of the regression vector. Unfortunately, the constraint that $|\beta^*| \in \Lambda$ is nonconvex and its implementation is computationally challenging. To overcome this difficulty, we propose a family of penalty functions, which are based on an extension of the ℓ_1 norm used by the Lasso method and involves the solution of a smooth convex optimisation problem. These penalty functions favour regression vectors β such that $|\beta| \in \Lambda$, thereby incorporating the structured sparsity constraints.

Precisely, we propose to estimate β^* as a solution of the convex optimization problem

$$\min \left\{ \|X\beta - y\|_2^2 + 2\rho\Omega(\beta|\Lambda) : \beta \in \mathbb{R}^n \right\} \tag{1.0.1}$$

where $\|\cdot\|_2$ denotes the Euclidean norm, ρ is a positive parameter and the penalty function takes the form

$$\Omega(\beta|\Lambda) = \inf \left\{ \frac{1}{2} \sum_{i \in \mathbb{N}_n} \left(\frac{\beta_i^2}{\lambda_i} + \lambda_i \right) : \lambda \in \Lambda \right\}.$$

As we shall see, a key property of the penalty function is that it exceeds the ℓ_1 norm of β when $|\beta| \notin \Lambda$, and it coincides with the ℓ_1 norm otherwise. This observation suggests a

[1] $P(\beta) = \sum_i |\beta_i|$, see Section 2.1.

heuristic interpretation of the method (1.0.1): among all vectors β which have a fixed value of the ℓ_1 norm, the penalty function Ω will encourage those for which $|\beta| \in \Lambda$. Moreover, when $|\beta| \in \Lambda$ the function Ω reduces to the ℓ_1 norm and, so, the solution of problem (1.0.1) is expected to be sparse. The penalty function therefore will encourage certain desired sparsity patterns. Indeed, the sparsity pattern of β is contained in that of the auxiliary vector λ at the optimum: if the set Λ allows only for certain sparsity patterns of λ, the same property will be "transferred" to the regression vector β.

There has been some recent research interest on structured sparsity, see [20, 22, 24, 29, 34, 56, 57] and references therein. Closest to our approach are penalty methods built around the idea of mixed ℓ_1-ℓ_2 norms. In particular, the Group Lasso method [57] assumes that the components of the underlying regression vector β^* can be partitioned into prescribed groups, such that the restriction of β^* to a group is equal to zero for most of the groups. This idea has been extended in [24, 58] by considering the possibility that the groups overlap according to certain hierarchical or spatially related structures. Although these methods have proved valuable in applications, they have the limitation that they can only handle more restrictive classes of sparsity, for example patterns forming only a single connected region. Our point of view is different from theirs and provides a means to designing more flexible penalty functions which maintain convexity while modeling richer model structures. For example, we will demonstrate that our family of penalty functions can model sparsity patterns forming multiple connected regions of coefficients.

In many case of interest the penalty function $\Omega(\beta|\Lambda)$ cannot be easily computed, and the solution to the associated regularization problem (1.0.1) is difficult to compute. We propose two methods for finding the solution. Firstly, a block coordinate descent algorithm inspired from [1], which is efficient but feasible only for a limited choice of set Λ. Secondly, an efficient proximal point method to solve regularised least squares with the penalty function $\Omega(\beta|\Lambda)$ in the general case of set Λ described above. The method combines a fast fixed point iterative scheme, which is inspired by recent work by [33] with an accelerated first order method equivalent to FISTA [5].

We present a numerical study of the statistical properties of the penalty terms, considering several different sparsity patterns. The error of the estimate is compared to what can be achieved with state of the art greedy methods that can handle similar structures, like StructOMP ([20]).

We also present efficiency experiments showing the performances of the proximal point method in solving the optimisation problem: not only it is faster than existing toolboxes, but it can handle much larger problems as well.

The thesis is organised as follows. In Chapter 2 there is background information about sparsity and structured sparsity, with a review of recent papers related to our work.

In Chapter 3 we present the technique, describing the properties of our penalty functions in Section 3.2, examples of the set Λ of special interest in Section 3.3, the dual problem in Section 3.4 and special cases of our functions in Section 3.5.

Next, in Chapter 4, we present several ways to solve numerically the proposed penalised problems. In Section 4.1 we present an alternating algorithm, and in Section 4.2 a proximal method with subgradient approximated by a the fixed point of a particular operator.

Part of this work, mostly from Chapter 3, appeared in the proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems (NIPS 2010) [32] (in particular Section 3.1, §3.2.2, §3.3.1, §3.3.2, §3.3.3, Section 4.1, and experiments from Section 5.1). An extende version the same paper is to appear in Advances in Computational Mathematics.

Chapter 2

Background

We consider the supervised problem of learning a linear model. We make two general assumptions on the model we wish to learn. The first one is of sparsity, that is the vast majority of the regression coefficients are zero. This is a widely used assumption both because it is realistic in many situations, and because it leads to computational advantages.

Secondly, we make the assumption that the nonzero components have a structure. The purpose of this assumption is to improve the estimate of the model by exploiting prior information. There are several different types of structures that can be formulated, leading to a very general problem that can be specified in many useful manners.

To find the estimate of the linear model we will explore two broad approaches. The first approach is to use greedy methods. These algorithms proceed iteratively including in the model a few components at a time, the ones that maximise the gain at the current step. Generally speaking, this approach is very fast but has the drawback that it can produce a suboptimal solution.

The second approach is to formulate the problem as a regularised convex problem, where the estimate is found as the minimiser of a balance between a loss function and a penalty term which promotes structured sparsity. Generally speaking, this approach is computational less appealing, but has the theoretical guarantee of a unique minimum. For this reason, we will follow this second approach in the thesis. This approach is also less intuitive, so we support it with a geometrical and a probabilistic interpretation.

Our problem is closely related to the area of compressive sensing, which belongs to the signal processing field of research. In compressive sensing, the signal of interest is sparse or closely sparse (compressible). We will discuss briefly this perspective presenting a greedy approach to it.

The chapter is organised as following. Section 2.1 contains an introduction to the linear problem and to sparsity. In Section 2.2, we review the greedy methods approach used to pur-

suit structured sparsity and in Section 2.3 we review the convex optimisation approach, with a selection of recent algorithms.

2.1 Standard sparsity

We consider a linear model

$$y = X\beta^* + \xi,$$

where $\beta^* \in \mathbb{R}^n$ is a vector of regression coefficients, $y \in \mathbb{R}^m$ is the vector of observations which depends linearly on $X \in \mathbb{R}^{m \times n}$, the data matrix, and $\xi \in \mathbb{R}^m$ is a vector of Gaussian noise. We address the problem of reconstructing the underlying vector β^* given the data.

The elements of the vector ξ are independent Gaussian $\mathcal{N}(0, \sigma^2)$, where the variance of the noise σ^2 is known. The noiseless version ($\xi = 0$) is treated similarly.

We consider the case when the matrix X is underdetermined: the number of observations m is (much) smaller than the dimension n. If we know, as it happens is many real situations, that the vector β^* has few nonzero components, then we can exploit this prior knowledge using some techniques to reconstruct it from (X, y). In particular, we usually assume that the underlying vector is at most s-sparse, that is it has at most s nonzero components. Usually, s is less than m, so it is much less than n. Under some properties on the data, the reconstruction of the underlying vector is possible.

We are interested in sparse models for several reasons. The sparsity assumption reduces the complexity of the learned function, which in turn has computational and statistical advantages. This assumption is also reasonable, as it appears naturally in several contexts. A sparse solution is also easier to interpret, because it is an implicit feature selection.

The problem is to find the vector $\hat{\beta}$ which solves

$$\min_{\beta \in \mathbb{R}^n} \{L(y, X\beta)\},$$

where L is an error function, a convex and differentiable function with respect to the second variable. For simplicity, we can consider the error term $\|y - X\beta\|_2^2$, but the logistic regression or other functions are possible as well.

In the case of interpolation described above, the error term can be zero. However, this can be an undesirable feature, because the predictive power of the regression could be poor. Moreover, it can add variability to the results. Two ways are usually followed to address these issues: either a greedy selection of variables or a regularisation of the problem. The next sections describes these approaches and some specific examples of recent and popular techniques.

It should be pointed out that a greedy selection refers to an algorithm that solves a problem (including regularised objective functions), while regularisation is a slight modification to

the original problem, which can be solved by different techniques (including greedy methods). We acknowledge that the two concepts are distinct, but we contrast them because in the literature they usually form the focus of two different approaches to sparsity: greedy methods are usually used to solve the noncovex problem of finding a sparse solution, while the focus of regularisation is to make a convex approximation to that problem.

Greedy algorithms can solve both convex and nonconvex problems, proceeding in an iterative way. At each step the model is enriched by adding the variable or group of variables that are more relevant, that is that best explain the output and minimise the residual.

The regularisation methods change the problem into

$$\min_{\beta \in \mathbb{R}^n} \{L(y, X\beta) + \rho P(\beta)\},$$

where P is a convex penalty function which encourages the vector β to have some properties. The penalty term is weighted with a nonnegative constant ρ which changes the amount of regularisation. As both L and P are convex functions, the problem is convex, the minimiser is unique and it is possible to find a solution numerically with standard techniques, see for instance [10].

Lasso. In the sparsity assumption on the vector β^* we are looking for a vector with at most s nonzero components out of n. This problem is combinatorial in nature, so computationally hard to solve. Conceptually it is the same as regularise with $P(\beta) = \#\{\beta_i : |\beta_i| > 0\}$, which is a nonconvex problem.

A common technique for sparsity (Lasso, see e.g. [49]) is to consider instead $P = \|\beta\|_1$. This problem is well studied, convex, and its solution is an approximation to the exact nonconvex problem.

The standard sparsity is an assumption only on the support of β^*, but no relationships between the nonzero variables are defined and exploited. In the Lasso, all components are treated alike. In the next sections, we will see how a structure on the support of β^* can be defined in different ways. This structure will be used as information to better recover vector β^*.

While describing structures of nonzero sparsity, it is often useful to put components of β into groups. A generic group of indices will be $J \subseteq \mathbb{N}_n = \{1, \dots, n\}$, and the vector containing only the components of the vector β indexed by the elements of the group J will be β_J. The set of all groups will be $\mathcal{J}$. In general, $\mathcal{J}$ needs not be a partition of $\mathbb{N}_n$, as some groups may overlap (i.e. some indices may belong to more than one group) or they may not exhaust the set $\mathbb{N}_n$ (i.e. some indices may not belong to any group).

2.2 Structured sparsity with greedy methods

Greedy methods for structured sparsity are iterative algorithms which estimate the model using only the features belonging to an active set of indices $A \subseteq \mathbb{N}_n$, and keep adding elements to A until a suitable stopping criterion is reached. At step t, the estimate $\beta^{(t)}$ is $|A|$-sparse, and the difference $r^{(t)} = y - X\beta^{(t)}$ between the output and the prediction is called the residual.

In the language of sparse approximation, we say that X is a dictionary of n atoms $X = [x_1, \ldots, x_n]$, and we assume that y has an s-term representation over the dictionary X. Also, it is often assumed that columns of X are normalised, so that each has 1 as the value of ℓ_2 norm.

The order in which components are chosen is set as to maximise the improvement of the estimate. Usually, an estimate is good if its residual is small.

These methods are usually fully deterministic, but the order in which they include components depends heavily on the data matrix X, so their performance may be highly variable. Moreover, the way to resolve ties may be random.

The simplest algorithm we consider is Orthogonal Matching Pursuit, OMP (see [50]), which can be extended quite naturally to recover clusters of nonzero components (Clustered and Sparse Regression algorithm, CaSpaR [42]).

Then we describe StructOMP ([20]), another extension of OMP based on information theory. This algorithm tries to learn a model which is as less complex as possible. For its flexibility and its superior performances to others greedy methods, it will be the focus of the experimental comparison with our convex optimisation approach.

Finally, we describe how the compressive sensing fits into the general sparsity discussion we are making, and we present the Model-based version (see [35]) for structured sparsity of the algorithm CoSaMP, similar to OMP.

The algorithms described in this chapter are summarised in Appendix C for easy reference.

In § 2.2.1 we present a simple greedy algorithm and a natural extension. In § 2.2.2 we describe the algorithm based on information theory. In § 2.2.3 we review the compressive sensing viewpoint on sparsity, presenting a greedy method solution to it.

2.2.1 OMP and CaSpaR

OMP. The algorithm Orthogonal Matching Pursuit (OMP), originally devised for sparse approximation, can be used for signal recovery as explained in [50]. The algorithm performs s iterations, repeatedly selecting the column of X that has the largest correlation with the current residual. OMP is equivalent to the statistical technique known as Forward Stepwise Regression (see e.g. [19]).

The algorithm will build a sequence of matrices $X^{(0)}$, $X^{(1)}$, ..., $X^{(s)}$, adding columns from X. Initially, we define the matrix $X^{(0)}$ as empty, and the residual $r^{(0)}$ as y: we have not yet an estimate, so the whole signal is unexplained. At the generic step t, the algorithm performs two operations: it selects the index of the feature to add to the model, and it produces a new estimate. The index is selected as

$$j^* = \underset{j=1,\dots,n}{\operatorname{argmax}} \left\{ |\langle r^{(t-1)}, x_j \rangle| \right\}, \tag{2.2.1}$$

which corresponds to the column mostly correlated with the residual. A tie may be broken deterministically by taking the lower index for which the maximum value is achieved. The column x_{j^*} is included in the active set, and is concatenated to the current matrix, $X^{(t)} = [X^{(t-1)}, x_{j^*}]$. Then, the new estimate for the signal is computed setting

$$\beta^{(t)} = \underset{\beta \in \mathbb{R}^{|A|}}{\operatorname{argmin}} \left\{ \|X^{(t)}\beta - y\|_2 \right\}. \tag{2.2.2}$$

Consequently, the new residual is now $r^{(t)} = y - X^{(t)}\beta^{(t)}$. Note that the vector $\beta^{(t)}$ has one component for each element in the active set, and that the estimate $\hat{\beta}$ for the full model in any given step t will be produced by setting $\hat{\beta}_j = \beta_j^{(t)}$ if $j \in A$, and $\hat{\beta}_j = 0$ otherwise.

The algorithm has a greedy approach in the sense that, at each step, it tries to minimise the residual as much as possible. If we consider the recurrence relation

$$r^{(t+1)} = y - \left[\sum_{j \in A \setminus \{j^*\}} x_j \beta_j^{(t+1)} + x_{j^*} \beta_{j^*}^{(t+1)} \right] = r^{(t)} - x_{j^*} \beta_{j^*}^{(t+1)},$$

then the length of the residual can be written as

$$\left\langle r^{(t+1)}, r^{(t+1)} \right\rangle = \left\langle r^{(t)}, r^{(t)} \right\rangle + \left(\beta_{j^*}^{(t+1)} \right)^2 - 2\beta_{j^*}^{(t+1)} \left\langle r^{(t)}, x_{j^*} \right\rangle.$$

Using the assumption that $\langle x_{j^*}, x_{j^*} \rangle = 1$, this expression supports the selecting criterion for j^* of Equation (2.2.1). Moreover, note that the residual is orthogonal to all the elements x_j for $j \in A$, so the new selected index j^* will not yet be in A, and no column will be selected twice.

This algorithm will run until s indices will be added to the model, so the number s must be known a priori. Alternatively, a tuning parameter τ can be used to stop the algorithm when the contribution of the added column, measured as the decrease of the residual, is negligible.

The solution to problem $(2.2.2)$ can be computed incrementally from the solution of the previous step, and is thus very efficient. The computational cost of the algorithm is dominated by the first step, that is computing $(2.2.1)$ (see [50]).

For the algorithm to achieve exact recovery, the output should come noiselessly from the input. Moreover, the matrix X should be incoherent, that is that the quantity

$$\mu(X) = \max_{1 \le j,k \le n} |\langle x_j, x_k \rangle|,$$

which is called the coherence of matrix X and which is the maximum inner product between different columns of X, should be small. Otherwise, the algorithm could select indices not in the support of the original vector.

The algorithm is very easy to implement, and can be fully analysed theoretically because of its simplicity. It does not promote a particular structure in the sparsity pattern of the estimate: we will focus on two of its many variants that indeed promote structure.

CaSpaR. In some applications, the underlying model is likely to be sparse and to consist of few connected regions of nonzero components. In [42], as a particular example, they assume that in a protein the mutations tend to cluster around "active sites". This is sustained by the knowledge of proteins' structures: it is in the active sites that proteins bind and have interactions with molecules. To exploit this assumption for prediction of mutations, they developed the algorithm Clustered and Sparse Regression (CaSpaR), a variant of forward stepwise regression procedures like OMP. Unlike the simplest original algorithm, each correlation $\left\langle r^{(t-1)}, x_j \right\rangle$ is weighted with a constant W_j so as to favour the selection of indices near the active set.

Initially, all weights are set to 1, so that no column is privileged. The two steps from Equations (2.2.1) and (2.2.2) are initially performed unchanged. At the generic iteration t, just before step $(2.2.1)$, a new set of weights is computed:

$$W_j = \frac{1}{|A|} \sum_{i \in A} K(d(i,j)),$$

for all $j \in \mathbb{N}_n$. The function d is a generic distance between two indices i and j, and K is a kernel function (non-negative integrable function). The weight for index j is the average of the distances, transformed by the kernel function, between j and all elements in A. The step from Equation (2.2.1) is now changed as

$$j^* = \underset{j \notin A}{\operatorname{argmax}} \left\{ W_j |\langle r^{(t-1)}, x_j \rangle| \right\}, \tag{2.2.3}$$

with the result that the selected index will be encouraged to belong to one of the clusters, defined by the distance d, of elements in set A.

The suggested choice for the kernel function is the mixture

$$K(x) = \alpha + (1-\alpha) K_e(x),$$

where K_e is the Epanechnikov kernel, that is $K_e(x) = \frac{3}{4}(1-x^2)$ for $|x| \leq 1$ and $K_e(x) = 0$ otherwise, although other mixtures are possible. The mixing parameter $\alpha \in [0,1]$ controls the effect that the distances between the index and the clusters has on the weights: when $\alpha = 1$, all weights become equal and the algorithm reduces to the original OMP.

The natural choice for the distance function is $d(i,j) = |i-j|$, but it could be modified according to the situation. For instance, if the model is embedded in a graph, then d could measure the length of the shortest path between nodes i and j.

The stopping criterion for the algorithm is $|\langle r^{(t-1)}, x_{j^*}\rangle| < \tau$, that is when the improvement of adding j^* drops below a positive threshold.

The mixing parameter α gives the theoretical guarantee that CaSpaR cannot be outperformed by the standard OMP. However, the ideal value for the parameter must be tuned via cross validation, and by the same means the threshold τ must be tuned as well. The grid search becomes a computational challenge, even if we stick with K_e: the use of a kernel with a parametrised bandwidth introduces a third parameter and another degree of complexity.

Even if the algorithm is flexible, as it can be customised by changing the definition of d, it is still suitable only for a particular structure, namely connected regions. The next variant of OMP is more general, and consequently less simple.

2.2.2 StructOMP

The algorithm StructOMP, proposed by [20], is a variant of standard OMP which is based on information theory. It lies on a generalisation of the concept of sparsity: a sparse vector has a low number of nonzero components, which means that it has a low content of information. The algorithm hinges on how the information is encoded, so that supports with particular structures may be easier to describe and hence are promoted. It is a variant of OMP in the sense that, at each step, it includes one or more columns into the model, so as to both maximise the decrease of the value of the loss function and minimise the increase of the complexity of the model.

Let F be a nonempty subset of the set $\mathbb{N}_n$ of indices. By definition, the coding length is a function $\mathrm{cl}(F)$ such that

$$\sum_{F\subseteq\mathbb{N}_n, F\neq\emptyset} 2^{-\mathrm{cl}(F)} \leq 1. \tag{2.2.4}$$

We use the coding length to define the complexity of the set F as the function $\mathrm{c}_s(F) = |F| + \mathrm{cl}(F)$, where $|F|$ is the number of elements of F. Finally, we can define the complexity of a coefficient vector $\beta \in \mathbb{R}^n$ as the complexity of the simplest set containing the support of β:

$$\mathrm{c}(\beta) = \min_{F\subseteq\mathbb{N}_n} \{\mathrm{c}_s(F) : \mathrm{supp}\,(\beta) \subseteq F\}. \tag{2.2.5}$$

Block sparsity. A first useful type of structured sparsity arises by considering blocks of variables. A block B is a set of indices, and the set $\mathcal{B}$ is the set of all blocks in which we are interested: the algorithm will promote models with a support that can be constructed as the union of few elements of $\mathcal{B}$.

The block set consists of a subset of the power set (all possible subsets) of $\mathbb{N}_n$, that is $\mathcal{B} \subseteq 2^{\mathbb{N}_n}$. To be well-formed, the sets must be an exhaustive collection of indices: $\cup_{B\in\mathcal{B}} B = \mathbb{N}_n$. This condition follows trivially if all singletons belong to the block set, or $\{j\} \in \mathcal{B}$, for all $j \in \mathbb{N}_n$. Moreover, this later assumption implies that any $F \subseteq \mathbb{N}_n$ can be expressed as a union of elements of $\mathcal{B}$, which does not follow from the first condition alone.

We assume that cl_0 is a coding length function for elements of the block set. A generic element $F \subseteq \mathbb{N}_n$ can be written as the union of elements of $\mathcal{B}$, so its coding length can be defined in terms of the function cl_0:

$$\mathrm{cl}(F) = \min \left\{ \sum_{B\in\mathcal{B}} [\mathrm{cl}_0(B) + 1] : F = \cup_{B\in\mathcal{B}} B \right\}. \tag{2.2.6}$$

This coding length can be used to define a cost function c for sets of indices F, using Equation (2.2.5).

For instance, we can consider the blocks of consecutive indices, that is $\mathcal{B} = \{\{a, a+1, \ldots, b\}, 1 \leq a < b \leq n\}$, and assume that each set has the same code length. Then, each of these blocks provides $2\log_2(n)$ bits of information, $\log_2(n)$ to store the position of the first index and $\log_2(n)$ to store the number of indices. Consequently, $\mathrm{cl}(B) = 2\log_2(n)$ for any $B \in \mathcal{B}$. Since $2^{-\mathrm{cl}(B)} = \frac{1}{n^2}$ and there are $|B| = \frac{n(n-1)}{2}$ such blocks, then Equation (2.2.4) is satisfied and cl is indeed a code length function. This block set can be used to promote models where the support is made of connected regions of indices, like in the CaSpaR algorithm.

The algorithm. StructOMP solves the problem

$$\hat{\beta} = \underset{\beta\in\mathbb{R}^n}{\operatorname{argmin}} \{L(\beta) : \mathrm{c}(\beta) \leq s\}, \tag{2.2.7}$$

where s is a parameter controlling the complexity of the learned vector. The focus is on the quadratic loss $L(\beta) = \|X\beta - y\|_2^2$, as it allows for some formula simplification.

The support F of the estimate $\hat{\beta}$ produced by Problem (2.2.7) is a union of blocks in $\mathcal{B}$. The blocks are chosen one per step, so at step t of the algorithm, a new block $B^{(t)}$ is added to the model: the support of the estimate at step t will be $\mathrm{supp}\left(\beta^{(t)}\right) = F^{(t)} = B^{(1)} \cup \ldots \cup B^{(t)}$. Note that the algorithm ignores blocks that can be expressed as the union of blocks already in the model. Moreover, it will automatically add all blocks for which the complexity of the support is not increased.

From step $t-1$ to step t, the new estimate is computed such that the loss function decreases as much as possible, and at the same time the cost increases as little as possible (because of the

information necessary to store block $B^{(t)}$). Then the goal is to maximise the gain ratio

$$\lambda(t) = \frac{L\left(\beta^{(t-1)}\right) - L\left(\beta^{(t)}\right)}{\mathsf{c}(\beta^{(t)}) - \mathsf{c}(\beta^{(t-1)})}. \tag{2.2.8}$$

When L is the quadratic loss, $\lambda(t)$ can be approximated by a function $\tilde{\phi}$ of the added block B:

$$\lambda(t) \approx \tilde{\phi}(B) = \frac{\|X^T_{B-F^{(t-1)}}(X\beta^{(t-1)} - y)\|_2^2}{\mathsf{c}(B \cup F^{(t-1)}) - \mathsf{c}(F^{(t-1)})},$$

which is easy to compute by testing all blocks in the block set. The algorithm terminates when the complexity of the current estimate $\hat{\beta}$ is larger than a threshold.

Other block sets. Grouping indices together in a single block creates a structure in the sparsity pattern. We revert to standard sparsity when the indices are not grouped, and each block is a singleton, or $\mathcal{B} = \{\{j\} : j \in \mathcal{J}\}$. A single index provides n bits of information, to store its position, so $\mathsf{cl}_0(\{j\}) = \log_2(n)$ for all $j \in \mathbb{N}_n$. Since a set F is uniquely expressible as the union of $|F|$ singletons, by Equation $(2.2.6)$ we have $\mathsf{cl}(F) = |F|\,(\log_2(n) + 1)$, and its complexity is $\mathsf{c}(F) = |F|\,(\log_2(2n) + 1)$.

The most general setting considered in [20] is the graph sparsity. The model is embedded into a graph G, so that each component is represented by a node (but the graph may contain other nodes as well, for further generality). In this case, the coding length is defined as a function of the neighbours of each node. The algorithm promotes structures of nonzero components clustered together. In fact, a connected region is easier to describe because its coding length is computed using only the information about the boundary.

A grid graph has a lattice of nodes: each one represents a pixel of an image and it is connected with its four adjacent pixels. Thanks to the specific topology of the grid, the clusters promoted by the algorithm are regions that are visually connected. The application is called denoising, and corresponds to the case when $X = I$ and y is the observed version, corrupted with noise, of the original image β^*. If we have reasons to assume that β^* has the property that it consists of a foreground of connected regions over a black background (zero values), then we can use the algorithm to recover it.

A second specific example of graph sparsity is given by a tree graph, where the blocks are all the connected subgraphs, including the single nodes. Again, a possible application is in image manipulations. Any image can be decomposed using a wavelet basis, that is a set of orthogonal functions that represent successive approximations, coarser to finer, of the data, and the corresponding wavelet coefficients. By construction, these functions arrange themselves hierarchically into a tree graph, where a wavelet is generated from the wavelet of the father

node. Here the assumption is that the nonzero coefficients of the image are clustered in the tree. The algorithm is performed in the wavelet space, and the estimated wavelet coefficients are used to reconstruct the estimated image.

This algorithm represents a good compromise between easy to use and effectiveness. It is conceptually simple, and it always involves just the selection of a block of variables at a time. Yet, it is general enough to encompass several different structures, even nonconvex constraints such as contiguous regions in a graph. Moreover, it can be easily adapted to other structures, as only the block set and the coding length needs to be redefined.

One possible drawback of the algorithm is that it can be computationally expensive. In fact, at each step all blocks must be evaluated for inclusion: the dimension of the block set is n from the the starting point of the standard sparsity, and more and more blocks are added in other cases. The coding length of the support, as per formula (2.2.6), depends on the best union of blocks that describes it, which leads to several different configurations to try.

Finally, because it is a greedy algorithm, it suffers from the possibility to be stuck into a local minimum. Consequently, it may happen that a block, which contains variables that do not belong to the true model, is selected because it maximises the gain ratio (2.2.8). Subsequently, more wrong variables are likely to be selected from the algorithm, because of the low information cost they carry. When this happens, the result estimate can be very poor.

2.2.3 Compressive Sensing

We review some concepts of compressive sensing following the exposition of [4]. The framework of compressive sensing is that we observe a vector $y = \Phi\beta$ that is generated from a signal $\beta \in \mathbb{R}^n$ via a measurement matrix Φ. Moreover, we assume that the signal can be represented as $\beta = \Psi\alpha$, where the square matrix Ψ contains a predefined basis, while the vector α contains the coefficients of the representation.

The signal β is said to be sparse if at most s elements of α are nonzero, where s is much smaller than n. This definition of sparsity is at the level of the representation of the signal, while the signal itself may not be sparse. We can always assume for simplicity that Ψ is the identity matrix, so that the signal has few nonzero components, and $y = \Phi\alpha$.

The assumption of compressive sensing is that a signal (or its representation) is sparse. This assumption can be relaxed by allowing the signal to be approximately sparse, or compressible. This means that its sorted coefficients decay fast enough to be approximated satisfyingly by an s-sparse vector.

More precisely, we consider a signal β for which the elements are ordered $\beta_{(1)}, \beta_{(2)}, \ldots$ by decreasing absolute value, that is $|\beta_{(i)}| \geq |\beta_{(j)}|$ for $i > j$. Suppose that the signal decays in

a power-law fashion, that is $|\beta_{(i)}| \leq Gi^{-1/r}$ for some constants G and r. We approximate β by the s-sparse vector β_s which minimises $e = \|\beta - \beta_s\|_p$, the error of the approximation computed with the ℓ_p norm. The vector β is called q-compressible if $r < p$ and $e \leq (rq)^{-1/p} G s^{-q}$, where $q = \frac{1}{r} - \frac{1}{p}$: that is the error of the best approximation decays as a power-law when s increases.

Restricted Isometry Property. Compressive sensing theory relies on the definition of a key property for matrices. We recall that an isometry is an $m \times n$ matrix A such that $\|Az\|_2^2 = \|z\|_2^2$ for all vectors $z \in \mathbb{R}^n$. That is, an isometry preserves the Euclidean length of any vector. This property can be restricted to be true only for s-sparse vectors, and to allow the length of the vector to be partially distorted. Precisely, a matrix A has the restricted isometry property if

$$(1 - \delta)\|z\|_2^2 \leq \|Az\|_2^2 \leq (1 + \delta)\|z\|_2^2$$

is true for all s-sparse vectors in $\mathbb{R}^n$. The positive constant δ controls the amount of relaxation.

This property is used to prove that the signal can be recovered. These results go beyond the scope of this thesis, so we just give an intuition. If the matrix $\Psi\Phi$ has the restricted isometry property (or just Ψ if $\Phi = I$), then each $m \times s$ submatrices are close (up to a constant) to be an isometry. This preserves distance and the information of a sparse or compressible signal, and thus guarantees that the signal can be recovered.

To recover a sparse signal, we should solve the problem $\min_\beta \{\|\beta\|_0\}$ such that $y = \Phi\beta$. This problem is an NP-hard combinatorial problem, and the recovery is not stable when the observation is noisy. A stable and feasible recovery can be made either by relaxing the problem using convex optimisation (see Section 2.3), or using an iterative greedy algorithms. Examples of greedy algorithms especially designed for compressive sensing include Iterative Hard Thresholding, IHT ([8]) and Compressive Sampling Matching Pursuit, CoSaMP ([35]).

Model-based CoSaMP. An extension of compressive sensing to structured signals was considered in [4], where the recovery algorithm CoSaMP is modified. We will not describe CoSaMP as it is similar to OMP (see § 2.2.1). The notion of compressible signal is extended to the one of structured compressible signal, and the original algorithm is adapted to handle general structures. In particular, tree sparsity and block sparsity are considered.

We define a structure by allowing only signals with supports belonging to the union of predefined sets. If Ω_m is one of the allowed supports, then we define $\mathcal{X}_m$ to be the subspace of $\mathbb{R}^n$ containing all signals β such that supp $(\beta) \subseteq \Omega_m$. The structured sparsity model is defined as $\mathcal{M} = \cup_{m \geq 1} \mathcal{X}_m$. The restricted isometry property used in the analysis of this algorithm is restricted to vectors $\beta \in \mathcal{M}$.

The algorithm depends on the computation of the best structured sparse approximation of the signal:

$$\mathbb{M}(\beta) = \operatorname*{argmin}_{z \in \mathcal{M}} \{\|\beta - z\|_2\},$$

which is the projection of a signal on the sparsity model $\mathcal{M}$. The algorithm has been adapted specifically for tree sparsity and block sparsity. In these two cases, known procedures compute the projection efficiently.

At each step, the support of the current estimate is merged with the support of $\mathbb{M}(X^T r)$, where r is the current residual. The resulting set, let it be T, is used to form a new estimate b: we set to $\Phi_T^\dagger y$ (the pseudo-inverse) the components of b indexed by T, and to 0 the components indexed by its complement $T^C = \mathbb{N}_n \backslash T$. The resulting vector is then pruned by projecting it back onto the sparsity model, to produce the signal estimate $\hat{\beta}^{(t)} = \mathbb{M}(b)$.

In model-based recovery, the class of structured compressible signals does not coincide (in fact, it is much larger) with the class of sparse signals. For this reason the restricted isometry property is not enough to assure recovery and other properties, we mention the nested approximation property (NAP) and the restricted amplification property (RAmP), which involves the residual subspaces of the model, must be used.

To define block sparsity as considered in [4], the signal must be regarded as a matrix, where each column corresponds to a block. This design is less general than the one supported by StructOMP, where blocks are not confined to particular positions and may overlap, leading for instance to connected regions.

The algorithm computes the function $\mathbb{M}$ twice at each step: first to extend the support of the estimate (by projecting $X^T r$ onto $\mathcal{M}$), then by pruning the estimate b, again projecting a vector onto $\mathcal{M}$. This can be expensive for a general structure, and a parallel can be drawn with StructOMP's step of computing $(2.2.6)$. In the case of StructOMP, however, the estimate is always within the current allowed support, and there is no need for pruning it.

2.3 Structured sparsity with convex optimisation

In the same way as the Lasso technique promotes sparsity, a convex penalty term can be used to promote structured sparsity. The completely convex nature of the problem has two main benefits. The first one is that the optimum always exists and is unique. The second benefit is that there are general efficient algorithms that can be used to compute this optimum.

The minimisation of the loss function subject to a constraint on the number of nonzero components of the support is a nonconvex problem. Even for standard sparsity, we approximate such number with the ℓ_1 norm. Likewise, a structure in the support of the model is nonconvex,

and must be approximated. It is evident, then, that the design of such a penalty term is not easy: not only this function has to be convex, but it must be able to promote a specific structure.

A particularly elegant extension to the ℓ_1 norm regularisation is the Group Lasso penalty ([56]), which is a mixture of ℓ_1 and ℓ_2. The components of the vector are grouped together and the penalty term is the sum of the ℓ_2 norms for each group. The effect is that each group will contain either zero or nonzero components. Apart from other variants of ℓ_1/ℓ_2 mixtures, or substituting the ℓ_2 for a general ℓ_p norm, a key extension to the Group Lasso is the Composite Absolute Penalty (CAP, [58]), which allows the groups to overlap. The effect is that certain components, belonging to different groups, are more penalised than others. In this way we can enforce a hierarchy among components, useful for important applications such as ANOVA models.

Other extensions to the Group Lasso and to CAP focus on the design of the groups in such a way as to promote one contiguous region in the model, both for 1D and 2D topologies. A support as a union of groups can be promoted using a variational problem, and the hierarchy of a two layered tree has been expressly studied.

The ℓ_1 norm as a penalty term is successful in promoting sparsity because its unit ball has nondifferentiable points along the axes. This geometrical intuition can be extended for group sparsity, as the set of nondifferentiable points are only on a small subsets of axes. Even if we cannot visualise in higher dimension, this idea, with its limits, can be helpful to gain some insights about the problem.

Finally, we discuss the Bayesian interpretation of the ℓ_1 regularisation, where the estimate of the Lasso model can be seen as the maximum a posteriori estimate of the model when we adopt the hypothesis that its prior distribution is Laplace. This interpretation can be extended to other penalty terms.

In § 2.3.1 we describe the Group Lasso and CAP. In § 2.3.2 other variants of the Group Lasso are described. The geometrical interpretation of the sparsity encouraging terms is described in § 2.3.3, while it Bayesian interpretation can be found in § 2.3.4.

2.3.1 Group Lasso and CAP

Group Lasso. In the Group Lasso [56], the penalty term takes the form

$$P(\beta) = \sum_{J \in \mathcal{J}} \sqrt{|J|} \|\beta_J\|_2,$$

that is we have a mixed ℓ_1/ℓ_2 norm. Group Lasso treats all the components in the same group in the same way, so they are all selected or discarded at the same time.

Each addend is multiplied by the coefficient $\sqrt{|J|}$, which is proportional to the dimension of the group, but a more general weighting can be used. Moreover, the ℓ_2 norm can be replaced by $\|\beta_J\|_{K_J}$, where $\|\eta\|_K = \sqrt{\eta^T K \eta}$ is the vector norm defined by a positive definite matrix K that can be different for each group.

CAP. The Composite Absolute Penalty family [58] extends the concept of group lasso in two ways. In the case of non-overlapping groups, the penalty is

$$P(\beta) = \sum_{J \in \mathcal{J}} {\|\beta_J\|_{\rho_J}}^{\gamma_0},$$

where, for generality, the ℓ_{γ_J} norm is used for group J, and the ℓ_{γ_0} norm to the γ_0-th power is computed for these norms.

The CAP was developed to consider a hierarchical structure on the components of β, defined by a Directed Acyclic Graph (DAG). Each node $v \in V$ of the DAG corresponds to a variable. If one variable is not included in the final model, then all the variables corresponding to descendants of v, that is the variables of nodes $D(v)$, are excluded from the model as well. To achieve this, the penalty function is modified not in the shape, but in the definition of groups:

$$P(\beta) = \sum_{J \in \mathcal{J}} \|(\beta_v, \beta_{D(v)})\|_{\gamma_v}.$$

This construction implies that, if $\|(\beta_v, \beta_{D(v)})\|_{\gamma_v}$ is zero, then $\|\beta_{D(v)}\|_{\gamma_v}$ must be zero. The enforced hierarchy is useful in many applications, noticeably in ANOVA, where if a main factor is excluded from the model, its mixed factors should be excluded as well.

Sparse Group Lasso. The sparse Group Lasso criterion proposed in [14] is a Group Lasso which encourages sparsity within each group, mixing the Group Lasso with an ℓ_1 norm penalty:

$$P(\beta) = \gamma_1 \sum_{J \in \mathcal{J}} \|\beta_J\|_2 + \gamma_2 \|\beta\|_1.$$

In the standard Group Lasso, the selected nonzero groups tend to be dense. Here, the ℓ_1 norm is added to help preventing this from happening. Parameters γ_1 and γ_2 should be estimated via cross validation.

Huber. Another hybrid ℓ_1/ℓ_2 penalisation, suggested in [40] and useful for standard sparsity, comes from the use of the inverse Huber function in the penalty term. We define

$$\mathcal{B}(\beta_i) = \begin{cases} |\beta_i| & |\beta_i| \leq 1 \\ \frac{\beta_i^2+1}{2} & |\beta_i| > 1 \end{cases},$$

and set the penalty term as

$$P(\beta) = \sum_i \mathcal{B}\left(\frac{\beta_i}{\tau}\right),$$

where τ is a scale parameter.

The function $\mathcal{B}$ is a scaled ℓ_1 norm for values smaller than τ, and is quadratic for larger values. The intention is to overcome two limitations of the lasso: that no more than m nonzero coefficients are selected (inconvenient in our case where $n > m$) and that it has less accurate prediction than ℓ_2.

2.3.2 Group Lasso variants

Contiguous regions. The Group Lasso when groups overlap has been investigated in [24]. The originality of the Structured Lasso lies in the selection of groups, while using the ℓ_2 norm within groups:

$$P(\beta) = \sum_{J \in \mathcal{J}} \|\beta_J\|_2.$$

If the components of β are thought to be aligned in a sequential order, then by considering all groups of the type $\{1, \ldots, k\}$ and $\{k, \ldots, n\}$, for $k = 1, \ldots, n$, it is possible to encourage the selection of a contiguous pattern of variables. The reason for this is that the ℓ_1 norm has the effect of excluding from the model any initial or final set of variables, leading to a pattern without holes.

In the same vein, if the components of β are thought to be on a grid, it is possible to construct the groups corresponding to all halfplanes starting from the four sides. The resulting nonzero pattern will be encouraged to be a rectangle. By adding more halfplanes with different orientation (e.g. all multiples of $\frac{\pi}{4}$), the sparsity pattern can be approximately convex. The approximation improves the more halfplanes are considered, though the complexity of the algorithm increases.

A weighting system allows a weight for each component within each group, so that a particular element β_i can have different weights, one for each group it belongs to. This leads to further generality, but no specific examples are suggested.

Overlapping groups. The goal in [22] is quite different: the support of the model is a union of K groups. This can be achieved by considering the Group Lasso with overlapping groups. In particular

$$P(\beta) = \inf_{v_{J_1}, \ldots, v_{J_K}} \left\{ \sum_{J \in \mathcal{J}} \|v_J\|_2 : \forall J \in \mathcal{J}, v_J \in \mathbb{R}^n, \text{supp}\,(v_J) \subseteq J, \sum_{J \in \mathcal{J}} v_J = \beta \right\}.$$

If the groups do not overlap and form a partition of $\mathbb{N}_n$, then the penalty is the same as the Group Lasso, because there exists a unique decomposition of β into vectors v_J such that for

each $\mathcal{J} \in \mathcal{J}$, the components of v_J indexed by J are the same as the components of β_J. In that case, the penalty induces the estimation of a sparse vector, whose support is the complement of a union of groups.

If groups overlap, then the support of the learned vector tends to be a union of groups. One application is the Graph Lasso: in an undirected graph, each vertex is a covariate. If overlapping groups are all linear subgraphs of length k, the penalty tends to select covariates connected to each other. Group Lasso with overlapping groups has been proved to be a norm, which can be useful if, for numerical reasons, the dual problem is addressed.

Hierarchical penalisation. The structure of a two layered tree for coefficients of β has been considered in [48]. The meaning of this graphical representation is that a node in the first layer represents group of variables (group J has a weight so that any variable β_i which belongs to J will be weighted with σ_{1i}) and a node in the second layer represents individual components, each one with its own weight σ_{2i}. The goal is to select a small number of groups and to shrink variables within each group.

The proposed penalty is

$$P(\beta) = \sum_i \frac{\beta_i^2}{\sqrt{\sigma_{1i}\sigma_{2i}}}.$$

The weights must be normalised: at the group level, it must hold that $\sum_{i \in J} \sigma_{1i} = \frac{1}{|J|}$, for all groups J and inside each group $\sum_{i \in \mathbb{N}_n} \sigma_{2i} = 1$. Interestingly, even if not evident, this penalty function is convex.

2.3.3 Geometric interpretation

In [18] we get a geometric argument as to why the regularisation using the ℓ_1 norm as penalty term is an effective way to select few nonzero components. We elaborate on it here showing how the same intuition applies to others penalty terms.

We consider two dimensions, and focus for simplicity on the square loss function. In this case, all the points with a fixed loss value lie on an ellipse centred around $\beta = X^\dagger y$, where $X^\dagger$ is the pseudoinverse of X. On the other hand, the set of points with a fixed penalty value is, in the ℓ_1 case, a diamond centred around the origin. This is the boundary of the scaled unit ball $B_1 = \{x : \|x\|_1 \leq 1\}$. See Figure 2.1.

We can imagine to expand these two shapes trying to find a balance between the loss function and the penalty term. The solution will be a single tangent point, because when the shapes are secant we can move inside one of the two, thus reducing the objective function. As a result of this process, it is clear that most of the time the solution will be a point on one axis.

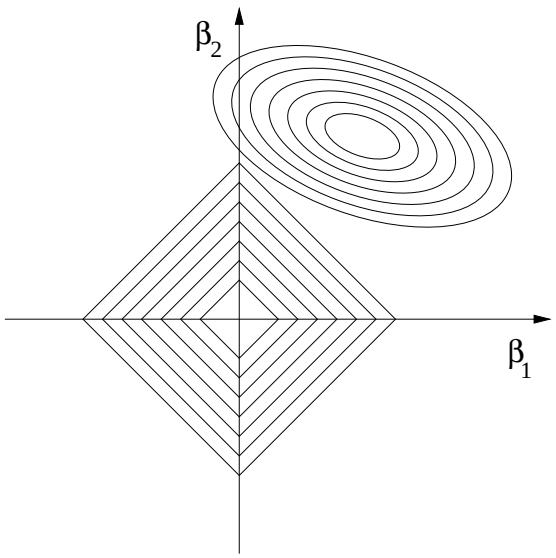


Figure 2.1: Geometrical intuition for the lasso.

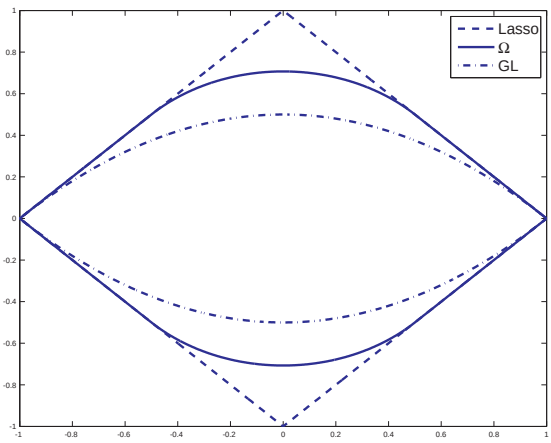


Figure 2.2: Unit balls of ℓ_1, Ω and Group Lasso.

Indeed, the only case when this will not happen is when the ellipse is tilted exactly 45 degrees and faces directly a side of the diamond.

It is evident then that the shape of the unit ball of the penalty function plays a central role in the determination of the set of points that are more likely to be the solution. The key feature of B_1 is to have non-differentiable points along the axes. Moreover, we note that the ℓ_1 norm promotes sparsity without structure, in the sense that neither axis is in a privileged position.

We consider now the unit ball of two structured sparsity penalties, the Group Lasso and the function Ω, which will be discussed in Chapter 3.

In Figure 2.2 we see how the two penalty functions are related to the ℓ_1 norm. Both functions can be defined in several ways. For the Group Lasso, we consider the hierarchically

overlapping groups $J_1 = \{x_1, x_2\}$ and $J_2 = \{x_2\}$, so that the unit ball is

$$B_{GL} = \{x : \|x\|_2 + |x_2| \leq 1\}.$$

For function Ω, we use the line graph penalty $\Lambda = \{\lambda : \lambda_1 \geq \lambda_2\}$ (see Section (3.3.2) for details). We can write the unit ball explicitly:

$$B_\Omega = \{x : \Omega(x|\Lambda) = 1\} = \begin{cases} \{x : |x_1| \geq |x_2|, \|x\|_1 \leq 1\} \\ \left\{x : |x_1| < |x_2|, \|x\|_2 \leq \sqrt{2}\right\}. \end{cases}$$

In both cases, the nondifferentiable points are on the x_1 axis: a sparse solution of the form $(\hat{x}, 0)$ will be far more likely than a sparse solution of the form $(0, \hat{x})$.

In three dimensions, the square loss produces an ellipsoid. All the nondifferentiable points of the unit ball of the penalty function are privileged candidates to be a solution. In three dimensions these points can form a curve, see Figure 3.3 for some examples of unit balls in three dimensions. Of course the intuition soon becomes useless as the number of dimensions increases, but nevertheless it is an important vehicle for understand how sparsity can be recovered with convex penalty functions.

2.3.4 Bayesian Lasso and MAP estimates

MAP estimates. The Maximum A Posteriori is an estimate of a parameter θ given an observed sample x. For any fixed value of the parameter, we assume that the probability distribution of the output is known, that is $p(x|\theta)$. Moreover, we assume to know the distribution of the parameter θ, that is $p(\theta)$. The MAP estimate $\hat{\theta}$ is given by the argmax of the posterior probability $p(\theta|x)$, which can be computed from the observations x.

By applying Bayes' theorem, we have that the posterior distribution of the parameter given the observations is proportional to the likelihood of observing the data given a parameter, $p(x|\theta)$, times the prior distribution of that parameter, $p(\theta)$.

The MAP estimate is defined as

$$\hat{\theta} = \text{argmax}\left\{p(\theta|x)\right\}.$$

In an equivalent way, we minimise the negative log-posterior distribution, that is

$$\hat{\theta} = \text{argmin}\left\{-log(f(\theta|x))\right\} = \text{argmin}\left\{-log(p(x|\theta)) - log(p(\theta))\right\}.$$

This second expression is easier to handle, and will lead to a link to regularisation.

MAP as regression. We consider the linear model $y = X\beta^* + \xi$, where the noise is Gaussian: $\xi \sim \mathcal{N}(0, \sigma^2 I_m)$. Then, the conditional distribution of y given a model β, that is the likelihood of the observation, is Gaussian: $y|\beta \sim \mathcal{N}(X\beta, \sigma^2 I_m)$.

The corresponding negative log-likelihood is $\frac{m}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\|y - X\beta\|_2^2$, and is maximised for the vector $\hat{\beta}$ which is the solution of the ordinary least squares. This vector is also the MAP estimate of the parameter when the unconditional distribution of β is assumed to be uniform (improper prior), because in that case its probability is a constant and does not effect the minimiser.

Making different assumptions on the prior distribution of the model β leads to MAP estimates that are equivalent to minimising the quadratic loss function plus a penalty term.

MAP as Lasso. The estimate of the Lasso technique, where we add $P(\beta) = \|\beta\|_1$, can be interpreted as the estimate which maximises the posterior distribution of β assuming that $y|\beta$ is Gaussian and that $\beta_i \sim \text{Laplace}\left(0, \frac{\tau}{\sqrt{2}}\right)$. The Laplace distribution with parameters 0 and $\frac{\tau}{\sqrt{2}}$ is $p(\beta_i) = \frac{1}{\tau\sqrt{2}} e^{-\frac{|\beta_i|}{\tau}\sqrt{2}}$.

In this case, to the quadratic loss function, we need to add the negative log-prior for vector β, which is $n\log\left(\tau\sqrt{2}\right) + \frac{\sqrt{2}}{\tau}\|\beta\|_1$. We recognise the resulting problem as the Lasso.

Note that, after normalisation, the ℓ_1 norm is multiplied by the tuning parameter γ which depends on the variances σ^2 and τ^2. In particular, $\gamma = 2\sigma^2\frac{\sqrt{2}}{\tau}$.

Ridge regression. For ridge regression, where $P(\beta) = \|\beta\|_2^2$, we have that the estimate can be interpreted as the MAP of β assuming that both $y|\beta$ and β are Gaussian. We assume that $\beta \sim \mathcal{N}\left(0, \tau^2 I_n\right)$, so that its negative log-prior is $\frac{n^2}{2}\log\left(2\pi\tau^2\right) + \frac{\|\beta\|_2^2}{2\tau^2}$.

We recover ridge regression problem where the ℓ_2 norm has the coefficient $\gamma = \frac{\sigma^2}{\tau^2}$, that is it depends on the variances of the y and β.

Bayesian Lasso. This variant of the Lasso was introduced in [41]. Here, the prior distribution of β_i given the value of σ^2 to be Laplacian, using a representation of Laplace distribution as mixture of normals. Specifically, we assume that $\beta|\sigma^2, \tau_1^2, \ldots, \tau_n^2 \sim \mathcal{N}(0, \sigma^2 D)$, where D is the diagonal matrix collecting the n auxiliary variables τ representing the variances of each component. Finally, we assume that σ^2 and the auxiliary variables τ are normally distributed. This formulation is useful for their approach to solve the problem based on Expectation-Maximisation (EM) algorithm.

Bayesian Lasso coincides with the variational formulation for the ℓ_1 norm. For any β, we have $\|\beta\|_1 = \frac{1}{2}\inf_\lambda \left\{\sum_i \left(\frac{\beta_i^2}{\lambda} + \lambda_i\right) : \lambda \in \mathbb{R}_{++}^n\right\}$, where $\mathbb{R}_{++}$ is the open positive orthant. We assume that $y|(\beta, \lambda)$ is Gaussian, and decompose the distribution of the vector β as $p(\beta, \lambda) = p(\beta|\lambda)p(\lambda)$.

The conditional distribution of β given λ is Normal, $\beta|\lambda \sim \mathcal{N}(0, \text{diag}(\lambda_1, \ldots, \lambda_n))$, and

its negative log-distribution is

$$\frac{n}{2}\log(2\pi) + \frac{1}{2}\log\left(\prod_i \lambda_i\right) + \frac{1}{2}\sum_i \frac{\beta_i^2}{\lambda_i}.$$

This term will be added to the negative log-prior of λ, so we choose a distribution such that the term $\frac{1}{2}\log\left(\prod_i \lambda_i\right)$ simplifies. This happens if $\lambda_i \sim \Gamma\left(\frac{3}{2}, \frac{1}{2}\right)$, as the negative log-prior distribution is

$$\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left(\prod \lambda_i\right) + \frac{1}{2}\|\lambda\|_1.$$

We conclude that the estimate

$$(\hat{\beta}, \hat{\lambda}) = \underset{\beta,\lambda}{\operatorname{argmin}} \left\{ \|y - X\beta\|_2^2 + \frac{\gamma}{2}\sum\left(\frac{\beta_i^2}{\lambda_i} + \lambda_i\right)\right\},$$

for $J(\beta) = \gamma\|\beta\|_1$, corresponds to a MAP estimate of parameters β and λ assuming that the likelihood of the observation is Gaussian, $y|(\beta,\lambda) \sim \mathcal{N}(X\beta, \sigma^2 I_m)$, that the conditional prior of β given λ is Gaussian, $\beta|\lambda \sim \mathcal{N}(0, \operatorname{diag}(\lambda_1, \ldots, \lambda_n))$, and finally that the prior distribution of λ is Gamma, $\lambda_i \sim \Gamma\left(\frac{3}{2}, \frac{1}{2}\right)$. After rescaling, we note that $\gamma = 2\sigma^2$. As we will see, this formulation resembles our proposed penalty function Ω, see Chapter 3.

Chapter 3

Modified ℓ_1 approach

Our aim is to exploit the prior knowledge of a structured sparse model by means of a convex regularization problem. The starting point is the Lasso technique, which promotes an unstructured solution by using the ℓ_1 norm as penalty term. It is well known that this norm can be rewritten as the infimum of a sum of quadratic functions. This variational formulation is convenient because the quadratic functions are smooth approximations from above of the ℓ_1 norm. This formulation depends on free auxiliary variables $\lambda \in \mathbb{R}^n_{++}$, the open positive orthant.

As we shall see, when the auxiliary variables are unconstrained, that is they belong to the positive orthant without further restrictions, we reduce to the Lasso technique. We propose to constraint these auxiliary variables within a subset Λ of the positive orthant. The result is a richly parametrised family of penalty functions $\Omega(\cdot|\Lambda)$. By imposing a structure on λ, we are indirectly imposing a similar structure on β, but we have the advantage that the resulting problem remains convex.

There are several convenient choices for the set Λ, some of them more general that others. For instance, by introducing relational constraints between components, or between differences of components, of vector λ, we can model hierarchical order on its coefficients, or contiguous regions of nonzero values. We analyse a selection of some of the many possibilities.

We also study function $\Omega(\cdot|\Lambda)$ in detail. Among the results, we derive the proximal operator of the function; we prove a number of properties, showing the conditions for which the function is a norm and deriving its dual; we show that other functions, such as the penalty terms for the Group Lasso and for Dirty Models, are indeed special cases of our function.

We begin by defining in detail our proposed penalty function in Section 3.1. Many important properties of this function are discussed in Section 3.2, while various members of the penalty family are proposed in Section 3.3. In Section 3.4 we propose a duality viewpoint which links the primal variables β of the model to the dual variables λ. Finally, in Section 3.5, we show some interesting special cases of the function Omega.

3.1 Proposed penalty

The prior knowledge that we consider is that the vector $|\beta^*|$, whose components are the absolute value of the corresponding components of β^*, should belong to some prescribed convex subset Λ of the positive orthant. For certain choices of Λ this implies a constraint on the sparsity pattern as well. For example, the set Λ may include vectors with some desired monotonicity constraints, or other constraints on the "shape" of the regression vector. Unfortunately, the constraint that $|\beta^*| \in \Lambda$ is nonconvex and its implementation is computational challenging. To overcome this difficulty, we propose a family of penalty functions, which are based on an extension of the ℓ_1 norm used by the Lasso method and involves the solution of a smooth convex optimization problem. These penalty functions incorporate the structured sparsity constraints.

Precisely, we propose to estimate β^* as a solution of the convex optimization problem

$$\min \left\{ \|X\beta - y\|^2 + 2\rho\Omega(\beta|\Lambda) : \beta \in \mathbb{R}^n \right\} \tag{3.1.1}$$

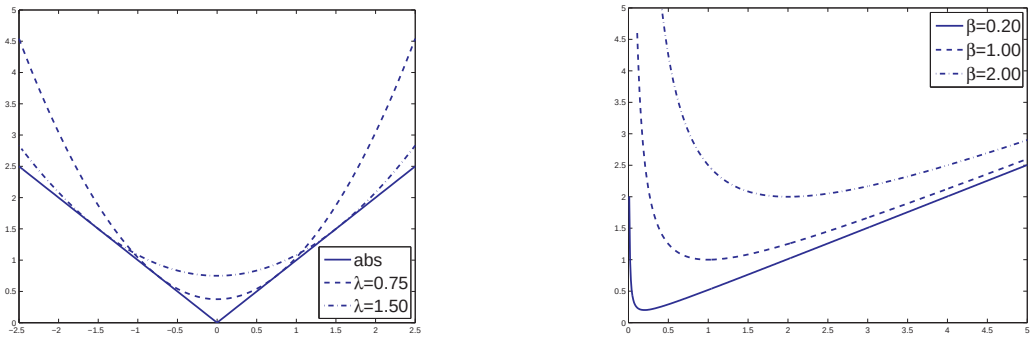
where $\|\cdot\|$ denotes the Euclidean norm, ρ is a positive parameter and the penalty function takes the form

$$\Omega(\beta|\Lambda) = \inf \left\{ \frac{1}{2} \sum_{i \in \mathbb{N}_n} \left(\frac{\beta_i^2}{\lambda_i} + \lambda_i \right) : \lambda \in \Lambda \right\}. \tag{3.1.2}$$

As we shall see, a key property of the penalty function is that it always exceeds the ℓ_1 norm of β unless $|\beta| \in \Lambda$ and it is strictly greater than the ℓ_1 norm otherwise. This observation suggests that the penalty function encourages the desired structured sparsity property.

Our approach also suggests that the parameter λ_i controls the degree of regularization on the corresponding regression coefficient β_i. The case that the set Λ consists of one point $\overline{\lambda}$ is instructive. In this case, the solution of the optimization problem (3.1.1) can be obtained explicitly as a solution to a Tikhonov regularization. It is important to realize that this optimization problem requires that all the components of $\overline{\lambda}$ are non-zero. However, the optimal solution, which we call $\beta(\overline{\lambda})$, can be shown to be defined even if some of the components of $\overline{\lambda}$ are zero. Indeed, when some of the components of the vector $\overline{\lambda}$ go to zero on some set $J \subseteq \mathbb{N}_n$, the same components of $\beta(\overline{\lambda})$ on this set go to zero as well. Moreover, the remaining components of $\beta(\overline{\lambda})$ on the complement of J provide a vector which solves the optimization problem restricted to all vectors whose components on J are zero. We will substantiate these observations in Section 3.4.

In this section, we provide some general comments on the penalty function. To this end, we let $\mathbb{R}^n_{++}$ be the open positive orthant, we let $\mathbb{N}_n$ be the set of positive integers up to n and



(a) (b)

Figure 3.1: (a): Function $\Gamma(\cdot,\lambda)$ for some values of $\lambda > 0$; (b): Function $\Gamma(\beta,\cdot)$ for some values of $\beta \in \mathbb{R}$.

define the function $\Gamma : \mathbb{R}^n \times \mathbb{R}^n_{++} \to \mathbb{R}$ by the formula

$$\Gamma(\beta,\lambda) = \frac{1}{2}\sum_{i\in\mathbb{N}_n}\left(\frac{\beta_i^2}{\lambda_i} + \lambda_i\right).$$

We let Λ be a nonempty subset of $\mathbb{R}^n_{++}$ and for every $\beta \in \mathbb{R}^n$, we define the penalty function $\Omega : \mathbb{R}^n \to \mathbb{R}$ at β as

$$\Omega(\beta|\Lambda) = \inf\left\{\Gamma(\beta,\lambda) : \lambda \in \Lambda\right\}. \tag{3.1.3}$$

Note that Γ is convex on its domain because each of its summands are likewise convex functions. Hence, when the set Λ is convex it follows that $\Omega(\cdot|\Lambda)$ is a convex function and (3.1.1) is a convex optimization problem.

An essential idea behind our construction of this function, is that, for every $\lambda \in \mathbb{R}_{++}$, the quadratic function $\Gamma(\cdot,\lambda)$ provides a smooth approximation to $|\beta|$ from above, which is exact at $\beta = \pm\lambda$. We indicate this graphically in Figure 3.1-a. This fact follows immediately by the arithmetic-geometric inequality, which states, for every $a, b \geq 0$ that $(a+b)/2 \geq \sqrt{ab}$.

A special case of the formulation (3.1.1) with $\Lambda = \mathbb{R}^n_{++}$ is the Lasso method, which is defined to be a solution of the optimization problem

$$\min\left\{\|y - X\beta\|^2 + 2\rho\|\beta\|_1 : \beta \in \mathbb{R}^n\right\}$$

where the ℓ_1-norm of the vector $\beta = (\beta_i : i \in \mathbb{N}_n) \in \mathbb{R}^n$ is defined as $\|\beta\|_1 = \sum_{i\in\mathbb{N}_n}|\beta_i|$. Indeed, using again the arithmetic-geometric inequality it follows that $\Omega(\beta|\mathbb{R}^n_{++}) = \|\beta\|_1$. Moreover, if for every $i \in \mathbb{N}_n$ $\beta_i \neq 0$, then the infimum is attained for $\lambda_i = |\beta_i|$. This important special case motivated us to consider the general method described above. The utility of (3.1.3) is that upon inserting it into (3.1.1) results in an optimization problem over λ and β with a continuously differentiable objective function. Hence, we have succeeded in expressing

a nondifferentiable convex objective function by one which is continuously differentiable on its domain.

3.2 Function properties

We have already seen that the proposed penalty term is a generalisation of the Lasso. In this section we explore many properties that belong to the function Ω.

The first result is the computation of the derivative of the function. This derivative depends on the value of the auxiliary vector at the minimum, so it cannot directly be used to solve the problem, but it is of theoretical relevance.

Next, we show that, when the set Λ is a convex cone, the function Ω is a norm. The assumption is not unrealistic, as it encompasses a very generic example like the Graph penalty (see § 3.3.3). This property is desirable because it lets the function inherit the properties of the norms.

We have a way to compose new penalties starting from a penalty set Λ and mixing it with a linear map. This rule was used for instance to design the Composite Wedge (see Section 5.1).

Writing explicitly the dual norm of the function, when Λ is a cone, is helpful as it allows to directly formulate the dual problem.

We show a necessary condition for the auxiliary variables vector to be the minimiser. This gives us one more insight about the role of this vector, and can be used as an alternative way to solve the problem.

Finally, function Ω has some properties of quasi homogeneity, which will be used in § 3.5.2 to define links with other algorithms.

In § 3.2.1 we present the derivative of Ω. The conditions for the function to be a norm are in § 3.2.2. In § 3.2.3 we present a linear composition rule. In § 3.2.4 we show what is the dual norm. The necessary condition for the auxiliary variables are in § 3.2.6. Properties of quasi homogeneity are in § 3.2.7.

3.2.1 Derivative of Ω

For any real numbers $a < b$, we define the parallelepiped $[a, b]^n = \{x : x = (x_i : i \in \mathbb{N}_n), a \leq x_i \leq b,\ i \in \mathbb{N}_n\}$.

Definition 3.2.1. *We say that the set Λ is admissible if it is convex and, for all $a, b \in \mathbb{R}$ with $0 < a < b$, the set $\Lambda_{a,b} := [a, b]^n \cap \Lambda$ is a nonempty, compact subset of the interior of Λ.*

Proposition 3.2.1. *If $\beta \in (\mathbb{R}\backslash\{0\})^n$ and Λ is an admissible subset of $\mathbb{R}^n_{++}$, then the infimum above is uniquely achieved at a point $\lambda(\beta) \in \Lambda$ and the mapping $\beta \mapsto \lambda(\beta)$ is continu-*

ous. Moreover, the function $\Omega(\cdot|\Lambda)$ is continuously differentiable and its partial derivatives are given, for any $i \in \mathbb{N}_n$, by the formula

$$\frac{\partial \Omega(\beta|\Lambda)}{\partial \beta_i} = \frac{\beta_i}{\lambda_i(\beta)}. \tag{3.2.1}$$

We postpone the proof of this proposition to Appendix B. We note that, since $\Omega(\cdot|\Lambda)$ is continuous, we may compute it at a vector β, some of which components are zero, as a limiting process. Moreover, at such a vector the function $\Omega(\cdot|\Lambda)$ is in general not differentiable, for example consider the case $\Omega(\beta|\mathbb{R}^n_{++}) = \|\beta\|_1$.

3.2.2 Conditions for being a norm

The next proposition provides a justification of the penalty function as a means to incorporate structured sparsity and establish circumstances for which the penalty function is a norm. To state our result, we denote by $\overline{\Lambda}$ the closure of the set Λ.

Proposition 3.2.2. *For every $\beta \in \mathbb{R}^n$, it holds that $\|\beta\|_1 \leq \Omega(\beta|\Lambda)$ and the equality holds* if and only if $|\beta| := (|\beta_i| : i \in \mathbb{N}_n) \in \overline{\Lambda}$. *Moreover, if Λ is a nonempty convex cone then the function $\Omega(\cdot|\Lambda)$ is a norm and we have that $\Omega(\beta|\Lambda) \leq \omega\|\beta\|_1$, where $\omega := \max\{\Omega(e_k|\Lambda) : k \in \mathbb{N}_n\}$ and $\{e_k : k \in \mathbb{N}_n\}$ is the canonical basis of $\mathbb{R}^n$.*

Proof. By the arithmetic-geometric inequality we have that $\|\beta\|_1 \leq \Gamma(\beta, \lambda)$, proving the first assertion. If $|\beta| \in \overline{\Lambda}$, there exists a sequence $\{\lambda^k : k \in \mathbb{N}\}$ in Λ, such that $\lim_{k\to\infty} \lambda^k = |\beta|$. Since $\Omega(\beta|\Lambda) \leq \Gamma(\beta, \lambda^k)$ it readily follows that $\Omega(\beta|\Lambda) \leq \|\beta\|_1$. Conversely, if $|\beta| \in \overline{\Lambda}$, then there is a sequence $\{\lambda^k : k \in \mathbb{N}\}$ in Λ, such $\gamma(\beta, \lambda^k) \leq \|\beta_1\| + 1/k$. This inequality implies that some subsequence of this sequence converges to a $\overline{\lambda} \in \overline{\Lambda}$. Using the arithmetic-geometric we conclude that $\overline{\lambda} = |\beta|$ and the result follows. To prove the second part, observe that if Λ is a nonempty convex cone, namely, for any $\lambda \in \Lambda$ and $t \geq 0$ it holds that $t\lambda \in \Lambda$, we have that Ω is positive homogeneous. Indeed, making the change of variable $\lambda' = \lambda/|t|$ we see that $\Omega(t\beta|\Lambda) = |t|\Omega(\beta|\Lambda)$. Moreover, the above inequality, $\Omega(\beta|\Lambda) \geq \|\beta\|_1$, implies that if $\Omega(\beta|\Lambda) = 0$ then $\beta = 0$. The proof of the triangle inequality follows from the homogeneity and convexity of Ω, namely $\Omega(\alpha + \beta|\Lambda) = 2\Omega\left((\alpha + \beta)/2|\Lambda\right) \leq \Omega(\alpha|\Lambda) + \Omega(\beta|\Lambda)$.

Finally, note that $\Omega(\beta|\Lambda) \leq \omega\|\beta\|_1$ if and only if $\omega = \max\{\Omega(\beta|\Lambda) : \|\beta\|_1 = 1\}$. Since Ω is convex the maximum above is achieved at an extreme point of the ℓ_1 unit ball. ■

This proposition indicates that the function $\Omega(\cdot|\Lambda)$ penalizes less vectors β which have the property that $|\beta| \in \Lambda$, thereby encouraging structured sparsity. Specifically, any permutation of

the coordinates of a vector β with the above property will incur in the same or a larger value of the penalty function. Moreover, for certain choices of the set Λ, some of which we describe below, the penalty function will encourage vectors which not only are sparse but also have sparsity patterns $(1_{\{|\beta_i|>0\}} : i \in \mathbb{N}_n) \in \Lambda$, where $1_{\{\cdot\}}$ denotes the indicator function. Note also that, the alternative formulation in which the constraint $|\beta| \in \Lambda$ is added directly as a constraint to the Lasso problem is not convex.

3.2.3 Composition of penalties

The next proposition presents a useful construction which may be applied to generate new penalty functions from available ones. It is obtained by composing a set $\Theta \subseteq \mathbb{R}^k_{++}$ with a linear transformation, modeling the sum of the components of a vector, across the elements of a prescribed partition $\mathcal{P} = \{P_\ell : \ell \in \mathbb{N}_k\}$ of $\mathbb{N}_n$. To describe our result we introduce the *group average map* $A_\mathcal{P} : \mathbb{R}^n \to \mathbb{R}^k$ induced by $\mathcal{P}$. It is defined, for each $\beta \in \mathbb{R}^n$, as $A_\mathcal{P}(\beta) = (\|\beta_{|P_\ell}\|_1 : \ell \in \mathbb{N}_k)$.

Proposition 3.2.3. *If $\Theta \subseteq \mathbb{R}^k_{++}$, $\beta \in \mathbb{R}^n$ and $\mathcal{P}$ is a partition of $\mathbb{N}_n$ then*

$$\Omega(\beta|A_\mathcal{P}^{-1}(\Theta)) = \Omega(A_\mathcal{P}(\beta)|\Theta).$$

Proof. The idea of the proof depends on two basic observations. The first uses the set theoretic formula

$$A_\mathcal{J}^{-1}(\Theta) = \bigcup_{\theta\in\Theta} A_\mathcal{J}^{-1}(\theta).$$

From this decomposition we obtain that

$$\Omega(\beta|A_\mathcal{J}^{-1}(\Theta)) = \inf\left\{\inf\left\{\Gamma(\beta,\lambda) : \lambda \in A_\mathcal{J}^{-1}(\theta)\right\} : \theta \in \Theta\right\}. \tag{3.2.2}$$

Next, we write $\theta = (\theta_\ell : \ell \in \mathbb{N}_k) \in \Theta$ and decompose the inner infimum as the sum

$$\sum_{\ell\in\mathbb{N}_k} \inf\left\{\frac{1}{2}\sum_{j\in J_\ell}\left(\frac{\beta_j^2}{\lambda_j} + \lambda_j\right) : \sum_{j\in J_\ell}\lambda_j = \theta_\ell, \lambda_j > 0, j \in J_\ell\right\}.$$

Now, the second essential step in the proof evaluates the infima in the second sum by Cauchy-Schwarz's inequality to obtain that

$$\inf\left\{\Gamma(\beta|\lambda) : \lambda \in A_\mathcal{J}^{-1}(\theta)\right\} = \sum_{\ell\in\mathbb{N}_k}\frac{1}{2}\left(\frac{\|\beta_{|J_\ell}\|_1^2}{\theta_\ell} + \theta_\ell\right).$$

We now substitute this formula into the right hand side of equation (3.2.2) to finish the proof. ■

3.2.4 Dual norm

When the set Λ is a nonempty convex cone, to emphasize that the function $\Omega(\cdot|\Lambda)$ is a norm we denoted it by $\|\cdot\|_\Lambda$. We end this section with the identification of the dual norm of $\|\cdot\|_\Lambda$ when Λ is a nonempty convex cone, which is defined as

$$\|\beta\|_{*,\Lambda} = \max\left\{\beta^\top u : u \in \mathbb{R}^n, \|u\|_\Lambda = 1\right\}.$$

Proposition 3.2.4. *If Λ is a nonempty convex cone, then there holds the equation*

$$\|\beta\|_{*,\Lambda} = \sup\left\{\sqrt{\frac{\sum_{i\in\mathbb{N}_n}\lambda_i\beta_i^2}{\sum_{i\in\mathbb{N}_n}\lambda_i}} : \lambda \in \Lambda\right\}.$$

Proof. By definition, $\varphi = \|\beta\|_{*,\Lambda}$ is the smallest constant φ such that, for every $\lambda \in \Lambda$ and $u \in \mathbb{R}^n$, it holds that

$$\frac{\varphi}{2}\sum_{i\in\mathbb{N}_n}\left(\frac{u_i^2}{\lambda_i} + \lambda_i\right) - \beta^\top u \geq 0.$$

Minimising the left hand side of this inequality for $u \in \mathbb{R}^n$ yields the equivalent inequality

$$\varphi^2 \geq \frac{\sum_{i\in\mathbb{N}_n}\lambda_i\beta_i^2}{\sum_{i\in\mathbb{N}_n}\lambda_i}.$$

Since this inequality holds for every $\lambda \in \Lambda$, the result follows by taking the supremum of the right hand side of the above inequality over this set. ■

The formula for the dual norm suggests that we introduce the set $\tilde{\Lambda} = \{\lambda : \lambda \in \Lambda, \sum_{i\in\mathbb{N}_n}\lambda_i = 1\}$. With this notation we see that the dual norm becomes

$$\|\beta\|_{*,\Lambda} = \sup\left\{\sqrt{\sum_{i\in\mathbb{N}_n}\lambda_i\beta_i^2} : \lambda \in \tilde{\Lambda}\right\}.$$

Moreover, a direct computation yields an alternate form for the original norm given by the equation

$$\|\beta\|_{\Lambda} = \inf\left\{\sqrt{\sum_{i\in\mathbb{N}_n}\frac{\beta_i^2}{\lambda_i}} : \lambda \in \tilde{\Lambda}\right\}.$$

Extreme points. Let $\text{ext}(\tilde{\Lambda})$ be the set of extreme points of $\tilde{\Lambda}$, that is all the points of $\tilde{\Lambda}$ that cannot be expressed as linear combination of other points in the same set. Since the function $\sum_{i\in\mathbb{N}_n}\lambda_i\beta_i^2$ is linear in λ, by the Fundamental Theorem of linear programming (see, for example, [6, Prop. B.21, p. 705]), we know that the optimum is always attained at an element of $\text{ext}(\tilde{\Lambda})$. This means that we can rewrite the expression for the dual norm as

$$\|\beta\|_{*,\Lambda} = \max\left\{\sqrt{\sum_{i\in\mathbb{N}_n}\lambda_i\beta_i^2} : \lambda \in \text{ext}(\tilde{\Lambda})\right\}.$$

The set of extreme points characterises the possible sparsity patterns allowed by the $\|\beta\|_{*,\Lambda}$ penalty. When Λ is a polyhedral convex cone, this set is finite, as is the case, for instance, of the Wedge and Tree penalties (see Sections 3.3.2 and 3.3.3). This, however, can be extended to cases when ext$(\tilde{\Lambda})$ is a countable infinite set.

Infimum convolution. We use the definition of dual norm to show how $\|\beta\|_\Lambda$ can be generalised by an infimum convolution, as suggested in [30]. In that paper, the authors define the norm

$$\|\beta\|_{\mathcal{M}} = \inf\left\{\sum_{M\in\mathcal{M}} \|v_M\| : v_M \in H, \sum_{M\in\mathcal{M}} Mv_M = \beta\right\},$$

where $\mathcal{M}$ is a finite or countably infinite set of linear operators and H is a real Hilbert space. In our setting, we can take $\mathcal{M}$ to be a set of matrices and H to be $\mathbb{R}^n$. They prove that the dual of this norm is

$$\|\beta\|_{\mathcal{M}^*} = \sum_{M\in\mathcal{M}} \|M\beta\|.$$

If we consider the set of diagonal matrices having as nonzero values the square roots of the extreme sets of $\tilde{\Lambda}$, that is

$$\mathcal{M} = \left\{\text{diag}(\sqrt{\lambda}) : \lambda \in \text{ext}(\tilde{\Lambda})\right\},$$

then we can see immediately that $\|M\beta\|$ becomes $\sqrt{\sum_{i\in\mathbb{N}_n} \lambda_i \beta_i^2}$. This implies that both $\|\cdot\|_{\mathcal{M}}$ and $\|\cdot\|_\Lambda$ share the same dual norm and are then equivalent.

For completeness, we prove that $\|\cdot\|_{\mathcal{M}}$ is a norm, adapting the proof of [30, Thm. 7, p. 10] to our setting. We prove the following.

Proposition 3.2.5. *Let $\mathcal{M}$ be a countably infinite set of real $n \times n$ symmetric matrices. We assume that, for every $x \in \mathbb{R}^n$, with $x \neq 0$, we have $Mx \neq 0$ for some $M \in \mathcal{M}$. Moreover, we assume*[1] *that $\sup_{M\in\mathcal{M}} |\!|\!| M |\!|\!| < \infty$. We define the set of vectors $\mathcal{V}(\mathcal{M}) = \{v : v = (v_M)_{M\in\mathcal{M}}, v_M \in \mathbb{R}^n\}$. Then*

$$\|\beta\|_{\mathcal{M}} = \inf\left\{\sum_{M\in\mathcal{M}} \|v_M\| : v \in \mathcal{V}(\mathcal{M}), \sum_{M\in\mathcal{M}} Mv_M = \beta\right\}$$

is a norm.

Proof. *(Nonnegative and positivity)* The function is clearly nonnegative, being a sum nonnegative terms. It is also positive if $\beta \neq 0$. Suppose then that $0 \neq \beta = \sum_{M\in\mathcal{M}} Mv_M$. Using the triangle inequality of the ℓ_2 norm, we can write

$$\|\beta\| \leq \sum_{M\in\mathcal{M}} \|Mv_M\| \leq \sum_{M\in\mathcal{M}} |\!|\!| M |\!|\!| \|v_M\| \leq \sup_{M\in\mathcal{M}} |\!|\!| M |\!|\!| \sum_{M\in\mathcal{M}} \|v_M\|,$$

[1] With the notation $|\!|\!| A |\!|\!|$ we refer to the operator norm, defined as $\sup\left\{\frac{\|Av\|}{\|v\|} : v \neq 0\right\}$.

where the middle inequality comes from the definition of operator norm. Now, taking the infimum with respect to v, we get

$$0 < \|\beta\| \leq \sup_{M \in \mathcal{M}} |||M||| \|\beta\|_{\mathcal{M}}$$

and, as $\sup_{M \in \mathcal{M}} |||M||| < \infty$ by hypothesis, we can conclude that $\|\beta\|_{\mathcal{M}} > 0$.

(Homogeneous) Scaling β by a constant a amounts to scaling all v_M by the same constant, which in turn, by homogeneity of the ℓ_2 norm, gives $|a|\|\beta\|_{\mathcal{M}}$.

(Triangle inequality) Let $\beta, \gamma \in \mathbb{R}^n$ and $w^\beta \in \mathcal{V}(\mathcal{M})$ be a set of auxiliary vectors such that $\sum_{M \in \mathcal{M}} M w_M^\beta = \beta$ and, for all $\epsilon > 0$, $\|\beta\|_{\mathcal{M}} + \epsilon \geq \sum_{M \in \mathcal{M}} \|w_M^\beta\|$. Let w^γ be a similar set of vectors for γ.

By definition,

$$\|\beta + \gamma\|_{\mathcal{M}} = \inf \left\{ \sum_{M \in \mathcal{M}} \|v_M\| : v \in \mathcal{V}(M), \sum_{M \in \mathcal{M}} M v_M = \beta + \gamma \right\}.$$

Since both $w^\beta \in \mathcal{V}(\mathcal{M})$ and $w^\gamma \in \mathcal{V}(\mathcal{M})$, then $w^\beta + w^\gamma \in \mathcal{V}(\mathcal{M})$. Moreover, we have $\sum_{M \in \mathcal{M}} M(w_M^\beta + w_M^\gamma) = \sum_{M \in \mathcal{M}} M w_m^\beta + \sum_{M \in \mathcal{M}} M w_m^\gamma = \beta + \gamma$. Then $w^\beta + w^\gamma$ is a feasible set of auxiliary vectors for $\beta + \gamma$, which cannot yield a value smaller than the infimum. That is

$$\|\beta + \gamma\|_{\mathcal{M}} \leq \sum_{M \in \mathcal{M}} \|w_M^\beta + w_M^\gamma\| \leq \sum_{M \in \mathcal{M}} \|w_M^\beta\| + \sum_{M \in \mathcal{M}} \|w_M^\gamma\| \leq \|\beta\|_{\mathcal{M}} + \|\gamma\|_{\mathcal{M}} + 2\epsilon.$$

where the second inequality is the triangle inequality for the ℓ_2 norm and the third inequality comes from the assumptions. Finally, since ϵ is free to go to 0, the triangle inequality for $\|\cdot\|_{\mathcal{M}}$ follows. ■

3.2.5 Dual norm of Lagrangian formulation

Suppose that $\omega : \mathbb{R}^n \to \mathbb{R}$ is a norm, and define the constraint set $\Lambda = \{\lambda : \omega(\lambda) \leq \alpha\}$, for a positive parameter α. The infimum in the definition of Ω can be written in the equivalent Lagrangian formulation

$$\inf \left\{ \frac{1}{2} \sum_{i \in \mathbb{N}_n} \left(\frac{\beta_i^2}{\lambda_i} + \lambda_i \right) + \gamma \omega(\lambda) \right\},$$

using the additional term $\gamma\omega(\lambda)$, where γ is a positive Lagrangian variable. Here we will prove that the Lagrangian formulation is a norm, and we derive its dual. Specifically, we have the following proposition.

Proposition 3.2.6. *If $\omega : \mathbb{R}^n \to \mathbb{R}$ is a norm, then the function*

$$\|\beta\|_\omega = \inf_{\lambda > 0} \left\{ \frac{1}{2} \sum_{i \in \mathbb{N}_n} \left(\frac{\beta_i^2}{\lambda_i} + \lambda_i \right) + \gamma \omega(\lambda) \right\}, \tag{3.2.3}$$

is a norm.

Proof. *(Nonnegative and positivity)* This function is clearly nonnegative, being a sum of nonnegative terms. It is also positive if $\beta \neq 0$. In fact, in order for $\|\beta\|_\omega$ to be zero, it must be $\omega(\lambda) = 0$, which gives an infinite value when $\beta \neq 0$. Indeed, the function is bounded by the value we obtain when $\lambda = |\beta|$, that is $\|\beta\|_\omega \leq \|\beta\|_1 + \gamma\omega(\beta)$.

(Homogeneous) Let $a > 0$ be a scalar. We can solve the problem

$$\|a\beta\|_\omega = \inf_{\lambda>0} \left\{ \frac{1}{2} \sum_{i\in\mathbb{N}_n} \left(\frac{(a\beta_i)^2}{\lambda_i} + \lambda_i \right) + \gamma\omega(\lambda) \right\},$$

by transforming the variables $\lambda_i \mapsto |a|\lambda_i$. That way we can collect a leading term $|a|$ and get, as desired, $|a|\|\beta\|_\omega$.

(Triangle inequality) The function $\sum_{i\in\mathbb{N}_n} \left(\frac{\beta_i^2}{\lambda_i} + \lambda_i \right)$ is jointly convex in λ and β, while $\omega(\lambda)$, being a norm, is convex in λ. Overall, we are taking the infimum with respect to λ of a function which is jointly convex, so the resulting function is convex (see, for instance, [10, Section 4.2.4]). Finally, since $\|\beta\|_\omega$ is convex and homogeneous, then it satisfies the triangle inequality, because $\|\beta + \xi\|_\omega = 2\|\frac{\beta+\xi}{2}\|_\omega \leq \|\beta\|_\omega + \|\xi\|_\omega$, as desired. ■

We can derive the dual norm of (3.2.3), in a similar way to what we did in § 3.2.4.

Proposition 3.2.7. *The dual norm of $\|\beta\|_\omega$ is*

$$\|\beta\|_{*,\omega} = \sup_{\lambda>0} \left\{ \sqrt{\frac{\sum_{i\in\mathbb{N}_n} \lambda_i\beta_i^2}{\sum_{i\in\mathbb{N}_n} \lambda_i + \gamma\omega(\lambda)}} \right\}.$$

Proof. By definition, $\varphi = \|\beta\|_{*,\omega}$ is the smallest constant φ such that for every $\lambda \in \Lambda$ and $u \in \mathbb{R}^n$, it holds that

$$\varphi \left(\frac{1}{2} \sum_{i\in\mathbb{N}_n} \left(\frac{u_i^2}{\lambda_i} + \lambda_i \right) + \gamma\omega(\lambda) \right) - \sum_{i\in\mathbb{N}_n} \beta_i u_i \geq 0.$$

We minimise over u to obtain the condition

$$\varphi^2 \geq \frac{\sum_{i\in\mathbb{N}_n} \beta_i^2\lambda_i}{\sum_{i\in\mathbb{N}_n} \lambda_i + \gamma\omega(\lambda)}.$$

Since this inequality holds for every $\lambda > 0$, the result follows by taking the supremum with respect to λ. ■

3.2.6 Equilibrium condition for optimality

In this section we establish a relationship between the point $\hat{\lambda}$, for which the infimum is attained, and the value of function Ω when the constraint set Λ is a cone.

Proposition 3.2.8. *Let Λ be a cone, β a vector in $\mathbb{R}^n$, and let $\hat{\lambda} \in \Lambda$ such that $\sum_{i\in\mathbb{N}_n}\left(\frac{\beta_i^2}{\hat{\lambda}_i}+\hat{\lambda}_i\right) \leq \sum_{i\in\mathbb{R}_n}\left(\frac{\beta_i^2}{\lambda_i}+\lambda_i\right)$ for all $\lambda \in \Lambda$. Then we have*

$$\Omega(\beta|\Lambda) = \|\hat{\lambda}\|_1 = \sum_{i\in\mathbb{N}_n}\frac{\beta_i^2}{\hat{\lambda}_i}. \tag{3.2.4}$$

Proof. We begin by noting that, within any ray R belonging to Λ, the minimum of $\frac{1}{2}\sum_{i\in\mathbb{N}_n}\left(\frac{\beta_i^2}{\lambda_i}+\lambda_i\right)$ is attained for a point $\lambda \in R$ for which the equilibrium condition $\sum_{i\in\mathbb{N}_n}\frac{\beta_i^2}{\lambda_i} = \sum_{i\in\mathbb{N}_n}\lambda_i$ is satisfied. Let v be any point in Λ, and rescale it to define the ray $R_v = \{\lambda : \lambda = kv, k \geq 0\} \subseteq \Lambda$. Then it is easy to see that $\hat{k} = \sqrt{\sum_{i\in\mathbb{N}_n}\frac{\beta_i^2}{v_i}}/\sqrt{\sum_{i\in\mathbb{N}_n} v_i}$ is the root of the derivative of $\frac{1}{2}\sum_{i\in\mathbb{N}_n}\left(\frac{\beta_i^2}{kv_i}+kv_i\right)$ with respect to k, that is $\hat{k}v$ is the minimiser within the ray. Moreover, $\sum_{i\in\mathbb{N}_n}\frac{\beta_i^2}{\hat{k}v_i} = \sqrt{\sum_{i\in\mathbb{N}_n}\frac{\beta_i^2}{v_i}\sum_{i\in\mathbb{N}_n} v_i} = \sum_{i\in\mathbb{N}_n}\hat{k}v_i$, so $\hat{k}v$ satisfies the equilibrium condition.

This is a necessary condition for optimality, so it must be satisfied by the point $\hat{\lambda}$ as well. Then $\sum_{i\in\mathbb{N}n}\frac{\beta_i^2}{\hat{\lambda}_i} = \sum_{i\in\mathbb{N}n}\hat{\lambda}_i = \frac{1}{2}\left(\sum_{i\in\mathbb{N}n}\frac{\beta_i^2}{\hat{\lambda}_i}+\sum_{i\in\mathbb{N}n}\hat{\lambda}_i\right) = \Omega(\beta|\Lambda)$ as required. ■

The equilibrium condition is satisfied by exactly one point for each ray in Λ, so when Λ consists of a single ray, it becomes a sufficient condition for optimality. In § 3.5.1 this case is explicitly considered.

The possibility of rescaling a vector suggests an alternative way of computing the value of Ω by restricting the value of $\sum_{i\in\mathbb{N}n}\frac{\beta_i^2}{v_i}$ and minimising $\|v\|_1$; the sought value will ensue after restoring the equilibrium condition by scaling. Precisely, we have the following result.

Proposition 3.2.9. *If Λ is a cone, then*

$$\Omega(\beta|\Lambda) = \min_{v\in\Lambda}\left\{\sqrt{\|v\|_1} : \sum_{i\in\mathbb{N}n}\frac{\beta_i^2}{v_i} = 1\right\}, \tag{3.2.5}$$

and if $\hat{v}$ is the point for which the minimum is attained, then $\hat{\lambda} = \hat{v}/\sqrt{\|\hat{v}\|_1}$ is the solution to the original problem, that is $\Omega(\beta|\Lambda) = \frac{1}{2}\sum_{i\in\mathbb{N}n}\left(\frac{\beta_i^2}{\hat{\lambda}_i}+\hat{\lambda}_i\right)$.

Proof. We call φ the value of the solution of (3.2.5) and prove that $\Omega(\beta|\Lambda) = \varphi$. Let $\hat{v}$ be the minimiser of (3.2.5), so that $\varphi = \sqrt{\|\hat{v}\|_1}$. Then we rescale this vector by $k = 1/\sqrt{\|\hat{v}\|_1}$ so as to satisfy the equilibrium condition: for the scaled vector $\lambda^\circ = k\hat{v}$ we have $\frac{1}{2}\sum_{i\in\mathbb{N}n}\left(\frac{\beta_i^2}{\lambda_i^\circ}+\lambda_i^\circ\right) = \sqrt{\|\hat{v}\|_1} = \varphi$, concluding that $\Omega(\beta|\Lambda) \leq \varphi$.

Let now $\hat{\lambda}$ be the minimiser of $\frac{1}{2}\sum_{i\in\mathbb{N}n}\left(\frac{\beta_i^2}{\lambda_i}+\lambda_i\right)$. We rescale this vector by $k = \sum_{i\in\mathbb{N}n}\frac{\beta_i^2}{\hat{\lambda}_i}$, so that the new vector $v^\circ = k\hat{\lambda}$ is a feasible point of (3.2.5), i.e. $\sum_{i\in\mathbb{N}n}\frac{\beta_i^2}{v_i^\circ} = 1$. Then $\sqrt{\|v^\circ\|_1} = \sqrt{\|\hat{\lambda}\|_1\sum_{i\in\mathbb{N}n}\frac{\beta_i^2}{\hat{\lambda}_i}} = \Omega(\beta|\Lambda)$, where the last equality follows from Proposition (3.2.8). We conclude that it is $\varphi \leq \Omega(\beta|\Lambda)$ as well, so consequently $\Omega(\beta|\Lambda) = \varphi$. ■

In a similar way it can be proved the related alternative way of computing the value of $\Omega(\beta|\Lambda)$ by changing the roles of $\sum_{i\in\mathbb{N}n}\frac{\beta_i^2}{v_i}$ and $\|v\|_1$, that is

$$\Omega(\beta|\Lambda)=\inf_{v\in\Lambda}\left\{\sqrt{\sum_{i\in\mathbb{N}n}\frac{\beta_i^2}{v_i}}:\|v\|_1=1\right\}. \tag{3.2.6}$$

3.2.7 Two quasihomogeneous properties

Let a be a nonnegative constant and Λ a cone, so that if $\lambda\in\Lambda$, then $\tilde{\lambda}=\sqrt{a}\lambda\in\Lambda$. Then

$$\frac{1}{2}\inf_{\lambda\in\Lambda}\left\{\sum_{i\in\mathbb{N}_n}\left(\frac{\beta_i^2}{\lambda_i}+a\lambda_i\right)\right\}=\sqrt{a}\Omega(\beta|\Lambda), \tag{3.2.7}$$

because $\frac{1}{\lambda_i}=\sqrt{a}\frac{1}{\tilde{\lambda}_i}$ and $a\lambda_i=\sqrt{a}\tilde{\lambda}_i$.

Something similar happens when a is the coefficient of the first term:

$$\frac{1}{2}\inf_{\lambda\in\Lambda}\left\{\sum_{i\in\mathbb{N}_n}\left(\frac{\beta_i^2}{a\lambda_i}+\lambda_i\right)\right\}=\frac{1}{\sqrt{a}}\Omega(\beta|\Lambda). \tag{3.2.8}$$

For this second case, we are looking for a transformation $\lambda_i\mapsto h\lambda_i$ for some positive coefficient h such that both $\frac{1}{\lambda_i}$ and λ_i will have a common coefficient, c, that can be collected outside the summation. Here we have $\frac{1}{ah\lambda_i}=c\frac{1}{\lambda_i}$ and $h\lambda_i=c\lambda_i$. Since $\frac{1}{ah}=c=h$, it readily follows that $h=\frac{1}{\sqrt{a}}$, implying the change of variables $\lambda_i\mapsto\frac{1}{\sqrt{a}}\lambda_i$. Analogous considerations lead to the first result.

When Λ is not a cone, then we can still bring the constant a outside the function. In that case, though, the function Ω will have a constraint set $\tilde{\Lambda}$ containing all $\tilde{\lambda}=\sqrt{a}\lambda$ or $\tilde{\lambda}=\frac{1}{\sqrt{a}}\lambda$ for all $\lambda\in\Lambda$.

3.3 Examples of set Λ

We discuss some specific instances of the set Λ and the associated penalty functions. These will prove to be important cases both from a theoretical and from a practical point of view.

The Box penalty introduces the constraint that the absolute value of each individual component of the vector is bound to be in an interval. This type of oracle information is hard to exploit because it is not the actual value to be confined in an interval: the absolute value regards as equal two possibilities of opposite signs, thus leading to a problem combinatorial in nature. This penalty has a closed form which resembles the Huber loss [40].

The Wedge penalty models the very natural assumption that the absolute values of the components of the model are sorted. We prove that this penalty has an analytical solution related to the Group Lasso. The penalty can be extended modelling a polynomial silhouette for the model: we constrain the differences of k-th order to be nonnegative for $k\geq 1$.

A natural generalisation of the Wedge is the Graph penalty, where a hierarchy on the absolute values are imposed from arbitrary topology. Specifically, we embed the model in a graph, so that each component is a node, and each directed edge an ordering constraint between components of these nodes. In general, this penalty has no closed form solution. However, we present a closed form solution when the graph is a tree.

As we shall see, the Tree penalty shares a general form with the Grid-C penalty, in which the sum of absolute values of the differences of arbitrary pairs of auxiliary variables is bounded, in a fashion similar to the Fused Lasso.

In § 3.3.1 we present the Box and its closed form. In § 3.3.2 we present the Wedge. In § 3.3.3 we present the Graph penalty. In § 3.3.4 we present Tree-C and Grid-C.

3.3.1 Box penalty

We proceed to discuss some examples of the set $\Lambda \subseteq \mathbb{R}^n_{++}$ which may be used in the design of the penalty function $\Omega(\cdot|\Lambda)$.

The first example, which is presented in this section, corresponds to the prior knowledge that the magnitude of the components of the regression vector should be in some prescribed intervals. We choose $a = (a_i : i \in \mathbb{N}_n)$, $b = (b_i : i \in \mathbb{N}_n) \in \mathbb{R}^n$, $0 \leq a_i \leq b_i$ and define the corresponding box as $B[a,b] := \{(\lambda_i : i \in \mathbb{N}_n) : \lambda_i \in [a_i, b_i],\ i \in \mathbb{N}_n\}$. The theorem below establishes the form of the box penalty. To state our result, we define, for every $t \in \mathbb{R}$, the function $(t)_+ = \max(0,t)$.

Theorem 3.3.1. *We have that*

$$\Omega(\beta|B[a,b]) = \|\beta\|_1 + \sum_{i\in\mathbb{N}_n} \left(\frac{1}{2a_i}(a_i - |\beta_i|)_+^2 + \frac{1}{2b_i}(|\beta_i| - b_i)_+^2 \right).$$

Moreover, the components of the vector $\lambda(\beta) := \operatorname{argmin}\{\Gamma(\beta,\lambda) : \lambda \in B[a,b]\}$ *are given by the equations* $\lambda_i(\beta) = |\beta_i| + (a_i - |\beta_i|)_+ - (|\beta_i| - b_i)_+$, $i \in \mathbb{N}_n$.

Proof. Since $\Omega(\beta|B[a,b]) = \sum_{i\in\mathbb{N}_n} \Omega(\beta_i|[a_i,b_i])$ it suffices to establish the result in the case $n = 1$. We shall show that if $a, b, \beta \in \mathbb{R}$, $a \leq b$ then

$$\Omega(\beta|[a,b]) = |\beta| + \frac{1}{2a}(a - |\beta|)_+^2 + \frac{1}{2b}(|\beta| - b)_+^2. \tag{3.3.1}$$

Since both sides of the above equation are continuous functions of β it suffices to prove this equation for $\beta \in \mathbb{R}\backslash\{0\}$. In this case, the function $\Gamma(\beta,\cdot)$ is strictly convex, and so, has a unique minimum in $\mathbb{R}_{++}$ at $\lambda = |\beta|$, see also Figure 3.1-b. Moreover, if $|\beta| \leq a$ the minimum occurs at $\lambda = a$, whereas if $|\beta| \geq b$, it occurs at $\lambda = b$. This establishes the formula for $\lambda(\beta)$.

Consequently, we have that

$$\Omega(\beta|[a,b]) = \begin{cases} |\beta|, & \text{if } |\beta| \in [a,b] \\ \frac{1}{2}\left(\frac{\beta^2}{a} + a\right), & \text{if } |\beta| < a \\ \frac{1}{2}\left(\frac{\beta^2}{b} + b\right), & \text{if } |\beta| > b. \end{cases}$$

Equation (3.3.1) now follows by a direct computation. ■

We also refer to [21, 40] for related penalty functions. Note that the function in equation (3.3.1) is a concatenation of two quadratic functions, connected together with a linear function. Thus, the box penalty will favor sparsity only for $a = 0$.

3.3.2 Wedge Penalty

In this section, we consider the case that the coordinates of the vector $\lambda \in \Lambda$ are ordered in a nonincreasing fashion. As we shall see, the corresponding penalty function favors regression vectors which are likewise nonincreasing.

We define the wedge

$$W = \{\lambda : \lambda = (\lambda_i : i \in \mathbb{N}_n) \in \mathbb{R}^n_{++}, \lambda_i \geq \lambda_{i+1},\ i \in \mathbb{N}_{n-1}\}.$$

Our next result describes the form of the penalty Ω in this case, for which we use the notation $\|\cdot\|_W$. To explain this result we require some preparation. We say that a partition $\mathcal{J} = \{J_\ell : \ell \in \mathbb{N}_k\}$ of $\mathbb{N}_n$ is *contiguous* if for all $i \in J_\ell, j \in J_{\ell+1}$, $\ell \in \mathbb{N}_{k-1}$, it holds that $i < j$. For example, if $n = 3$, partitions $\{\{1,2\},\{3\}\}$ and $\{\{1\},\{2\},\{3\}\}$ are contiguous but $\{\{1,3\},\{2\}\}$ is not.

Definition 3.3.1. *Given any two disjoint subsets $J, K \subseteq \mathbb{N}_n$ we define the region in $\mathbb{R}^n$*

$$Q_{J,K} = \left\{\beta : \beta \in \mathbb{R}^n, \frac{\|\beta_{|J}\|_2^2}{|J|} > \frac{\|\beta_{|K}\|_2^2}{|K|}\right\}. \tag{3.3.2}$$

Note that the boundary of this region is determined by the zero set of a homogeneous polynomial of degree two. We also need the following construction.

Definition 3.3.2. *For every subset $S \subseteq \mathbb{N}_{n-1}$ we set $k = |S| + 1$ and label the elements of S in increasing order as $S = \{j_\ell : \ell \in \mathbb{N}_{k-1}\}$. We associate with the subset S a contiguous partition of $\mathbb{N}_n$, given by $\mathcal{J}(S) = \{J_\ell : \ell \in \mathbb{N}_k\}$, where we define $J_\ell := [j_{\ell-1} + 1, j_\ell] \cap \mathbb{N}_n$, $\ell \in \mathbb{N}_k$, and set $j_0 = 0$ and $j_k = n$.*

A subset S of $\mathbb{N}_{n-1}$ also induces two regions in $\mathbb{R}^n$ which play a central role in the identification of the wedge penalty. First, we describe the region which "crosses over" the induced partition $\mathcal{J}(S)$. This is defined to be the set

$$O_S := \bigcap \left\{Q_{J_\ell, J_{\ell+1}} : \ell \in \mathbb{N}_{k-1}\right\}. \tag{3.3.3}$$

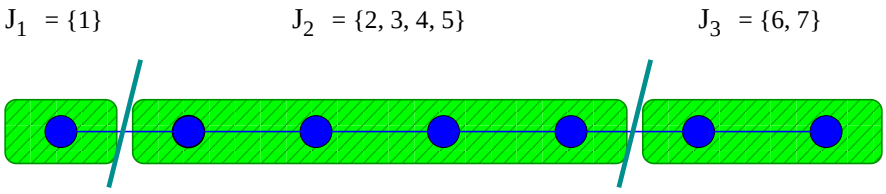


Figure 3.2: Partition of $\beta = (-1.477, 0.694, -0.173, -0.916, -1.126, 0.525, -0.957)$.

In other words, $\beta \in O_S$ if the average of the square of its components within each region J_ℓ strictly decreases with ℓ. The next region which is essential in our analysis is the "stays within" region, induced by the partition $\mathcal{J}(S)$. To identify this region we use the notation $J_{\ell,q} := \{j : j \in J_\ell, j \leq q\}$ and is defined by the equation

$$I_S := \bigcap \left\{ \overline{Q}_{J_\ell, J_{\ell,q}} : q \in J_\ell, \ell \in \mathbb{N}_k \right\} \tag{3.3.4}$$

where $\overline{Q}$ denotes the closure of the set Q. In other words, all vectors β within this region have the property that, for every set $J_\ell \in \mathcal{J}(S)$, the average of the square of a first segment of components of β within this set is not greater than the average over J_ℓ. We note that if S is the empty set the above notation should be interpreted as $O_S = \mathbb{R}^n$ and

$$I_S = \bigcap \{ \overline{Q}_{\mathbb{N}_n, \mathbb{N}_q} : q \in \mathbb{N}_n \}.$$

Figure 3.2 illustrates an example of a contiguous partition, along with the set $\mathcal{J}(S)$, for a vector $\beta = (-1.477, 0.694, -0.173, -0.916, -1.126, 0.525, -0.957)$. We can check that, for this vector, the partition $\{\{1\}, \{2, 3, 4, 5\}, \{6, 7\}\}$ belongs both to regions O and I. For the crosses over region it must be that $\beta_1^2 > \frac{\beta_2^2+\beta_3^2+\beta_4^2+\beta_5^2}{4} > \frac{\beta_6^2+\beta_7^2}{2}$, which is the case since $2.182 > 0.655 > 0.596$. For the stays with region it must be that $\frac{\beta_2^2+\beta_3^2+\beta_4^2+\beta_5^2}{4}$ is larger than β_2^2, $\frac{\beta_2^2+\beta_3^2}{2}$ and $\frac{\beta_2^2+\beta_3^2+\beta_4^2}{3}$, and moreover $\frac{\beta_6^2+\beta_7^2}{2} \geq \beta_6^2$. This is the case because $0.655 \geq \max(0.482, 0.256, 0.450)$ and $0.596 \geq 0.276$.

From the cross-over and stay-within sets we define the region

$$P_S = O_S \cap I_S.$$

Alternatively, we shall describe below the set P_S in terms of two vectors induced by a vector $\beta \in \mathbb{R}^n$ and the set $S \subseteq \mathbb{N}_{n-1}$. These vectors play the role of the Lagrange multiplier and the minimizer λ for the wedge penalty in the theorem below.

Definition 3.3.3. *For every vector $\beta \in (\mathbb{R}\backslash\{0\})^n$ and every subset $S \subseteq \mathbb{N}_{n-1}$ we let $\mathcal{J}(S)$ be the induced contiguous partition of $\mathbb{N}_n$ and define two vectors $\zeta(\beta, S) \in \mathbb{R}_+^{n+1}$ and $\delta(\beta, S) \in$*

$\mathbb{R}^n_{++}$ *by*

$$\zeta_q(\beta, S) = \begin{cases} 0, & \text{if } q \in S \cup \{0, n\}, \\ |J_{\ell,q}| - |J_\ell| \frac{\|\beta_{|J_{\ell,q}}\|_2^2}{\|\beta_{|J_\ell}\|_2^2}, & \text{if } q \in J_\ell, \ell \in \mathbb{N}_k \end{cases}$$

and

$$\delta_q(\beta, S) = \frac{\|\beta_{|J_\ell}\|_2}{\sqrt{|J_\ell|}},\ q \in J_\ell, \ell \in \mathbb{N}_k. \tag{3.3.5}$$

Note that the components of $\delta(\beta, S)$ are constant on each set J_ℓ, $\ell \in \mathbb{N}_k$.

Lemma 3.3.1. *For every $\beta \in (\mathbb{R}\backslash\{0\})^n$ and $S \subseteq \mathbb{N}_{k-1}$ we have that*

(a) $\beta \in P_S$ if and only if $\zeta(\beta, S) \geq 0$ *and* $\delta(\beta, S) \in \operatorname{int}(W)$*;*

(b) If $\delta(\beta, S_1) = \delta(\beta, S_2)$ and $\beta \in O_{S_1} \cap O_{S_2}$ then $S_1 = S_2$.

Proof. The first assertion follows directly from the definition of the requisite quantities. The proof of the second assertion is a direct consequence of the fact that the vector $\delta(\beta, S)$ is a constant on any element of the partition $\mathcal{J}(S)$ and strictly decreasing from one element to the next in that partition. ■

For the theorem below we introduce, for every $S \in \mathbb{N}_{n-1}$ the sets

$$U_S := P_S \cap (\mathbb{R}\backslash\{0\})^n.$$

We shall establishes not only that the collection of sets $\mathcal{U} := \{U_S : S \subseteq \mathbb{N}_{n-1}\}$ form a *partition* of $(\mathbb{R}\backslash\{0\})^n$, that is, their union is $(\mathbb{R}\backslash\{0\})^n$ and two distinct elements of $\mathcal{U}$ are disjoint, but also explicitly determine the wedge penalty on each element of $\mathcal{U}$.

Theorem 3.3.2. *The collection of sets $\mathcal{U} := \{U_S : S \subseteq \mathbb{N}_{n-1}\}$ form a partition of $(\mathbb{R}\backslash\{0\})^n$. For each $\beta \in (\mathbb{R}\backslash\{0\})^n$ there is a unique $S \subseteq \mathbb{N}_{n-1}$ such that $\beta \in \mathcal{U}_S$, and*

$$\|\beta\|_W = \sum_{\ell \in \mathbb{N}_k} \sqrt{|J_\ell|} \|\beta_{|J_\ell}\|_2, \tag{3.3.6}$$

where $k = |S|+1$. Moreover, the components of the vector $\lambda(\beta) := \operatorname{argmin}\{\Gamma(\beta, \lambda) : \lambda \in W\}$ are given by the equations $\lambda_j(\beta) = \mu_\ell,\ j \in J_\ell,\ \ell \in \mathbb{N}_k$, where

$$\mu_\ell = \frac{\|\beta_{|J_\ell}\|}{\sqrt{|J_\ell|}}. \tag{3.3.7}$$

Proof. First, let us observe that there are $n-1$ inequality constraints defining W. It readily follows that all vectors in this constraint set are *regular*, in the sense of optimization theory, see [6, p. 279]. Hence, we can appeal to [6, Prop. 3.3.4, p. 316 and Prop. 3.3.6, p. 322], which

state that $\lambda \in \mathbb{R}^n_{++}$ is a solution to the minimum problem determined by the wedge penalty, if and only if there exists a vector $\alpha = (\alpha_i : i \in \mathbb{N}_{n-1})$ with nonnegative components such that

$$-\frac{\beta_j^2}{\lambda_j^2} + 1 + \alpha_{j-1} - \alpha_j = 0, \quad j \in \mathbb{N}_n, \tag{3.3.8}$$

where we set $\alpha_0 = \alpha_n = 0$. Furthermore, the following complementary slackness conditions hold true

$$\alpha_j(\lambda_{j+1} - \lambda_j) = 0, \; j \in \mathbb{N}_{n-1}. \tag{3.3.9}$$

To unravel these equations, we let $\hat{S} := \{j : \lambda_j > \lambda_{j+1}, j \in \mathbb{N}_{n-1}\}$, which is the subset of indexes corresponding to the constraints that are not tight. When $k \geq 2$, we express this set in the form $\{j_\ell : \ell \in \mathbb{N}_{k-1}\}$ where $k = |\hat{S}|+1$. As explained in Definition 3.3.2, the set $\hat{S}$ induces the partition $\mathcal{J}(\hat{S}) = \{J_\ell : \ell \in \mathbb{N}_k\}$ of $\mathbb{N}_n$. When $k = 1$ our notation should be interpreted to mean that $\hat{S}$ is empty and the partition $\mathcal{J}(\hat{S})$ consists only of $\mathbb{N}_n$. In this case, it is easy to solve equations (3.3.8) and (3.3.9). In fact, all components of the vector λ have a common value, say $\mu > 0$, and by summing both sides of equation (3.3.8) over $j \in \mathbb{N}_n$ we obtain that

$$\mu^2 = \frac{\|\beta\|_2^2}{n}.$$

Moreover, summing both sides of the same equation over $j \in \mathbb{N}_q$ we obtain that

$$\alpha_q = -\frac{\sum_{j\in\mathbb{N}_q} \beta_j^2}{\mu^2} + q$$

and, since $\alpha_q \geq 0$ we conclude that $\beta \in I_{\hat{S}} = P_{\hat{S}}$.

We now consider the case that $k \geq 2$. Hence, the vector λ has equal components on each subset J_ℓ, which we denote by μ_ℓ, $\ell \in \mathbb{N}_{k-1}$. The definition of the set $\hat{S}$ implies that the sequence $\{\mu_\ell : \ell \in \mathbb{N}_k\}$ is strictly decreasing and equation (3.3.9) implies that $\alpha_j = 0$, for every $j \in \hat{S}$. Summing both sides of equation (3.3.8) over $j \in J_\ell$ we obtain that

$$-\frac{1}{\mu_\ell^2} \sum_{j\in J_\ell} \beta_j^2 + |J_\ell| = 0 \tag{3.3.10}$$

from which equation (3.3.7) follows. Since the μ_ℓ are strictly decreasing, we conclude that $\beta \in O_{\hat{S}}$. Moreover, choosing $q \in J_\ell$ and summing both sides of equations (3.3.8) over $j \in J_{\ell,q}$ we obtain that

$$0 \leq \alpha_q = -\frac{\|\beta_{|J_{\ell,q}}\|_2^2}{\mu_\ell^2} + |J_{\ell,q}|$$

which implies that $\beta \in \overline{Q}_{J_\ell,J_{\ell,q}}$. Since this holds for every $q \in J_\ell$ and $\ell \in N_k$ we conclude that $\beta \in I_{\hat{S}}$ and therefore, it follows that $\beta \in U_S$.

In summary, we have shown that $\alpha = \zeta(\beta, \hat{S})$, $\lambda = \delta(\beta, \hat{S})$, and $\beta \in U_{\hat{S}}$. In particular, this implies that the collection of sets $\mathcal{U}$ covers $(\mathbb{R}\backslash\{0\})^n$. Next, we show that the elements of

$\mathcal{U}$ are disjoint. To this end, we observe that, the computation described above can be *reversed*. That is to say, conversely for *any* $\hat{S} \subseteq \mathbb{N}_{n-1}$ and $\beta \in U_{\hat{S}}$ we conclude that $\delta(\beta, \hat{S})$ and $\zeta(\beta, \hat{S})$ solve the equations (3.3.8) and (3.3.9). Since the wedge penalty function is *strictly convex* we know that equations (3.3.8) and (3.3.9) have a unique solution. Now, if $\beta \in U_{S_1} \cap U_{S_2}$ then it must follow that $\delta(\beta, S_1) = \delta(\beta, S_2)$. Consequently, by part (b) in Lemma 3.3.1 we conclude that $S_1 = S_2$. ■

Note that the set S and the associated partition $\mathcal{J}$ appearing in the theorem is identified by examining the optimality conditions of the optimization problem (3.1.2) for $\Lambda = W$. There are 2^{n-1} possible partitions. Thus, for a given $\beta \in (\mathbb{R}\backslash\{0\})^n$, determining the corresponding partition is a challenging problem. We explain how to do this in Section 4.1.

An interesting property of the Wedge penalty, which is indicated by Theorem 3.3.2, is that it has the form of a Group Lasso penalty as in equation (3.5.4), with groups not fixed *a-priori* but depending on the location of the vector β. The groups are the elements of the partition $\mathcal{J}$ and are identified by certain convex constraints on the vector β. For example, for $n = 2$ we obtain that $\Omega(\beta|W) = \|\beta\|_1$ if $|\beta_1| > |\beta_2|$ and $\Omega(\beta|W) = \sqrt{2}\|\beta\|_2$ otherwise. For $n = 3$, we have that

$$\Omega(\beta|W) = \begin{cases} \|\beta\|_1, & \text{if } |\beta_1| > |\beta_2| > |\beta_3| & \mathcal{J} = \{\{1\},\{2\},\{3\}\} \\ \sqrt{2(\beta_1^2+\beta_2^2)} + |\beta_3|, & \text{if } |\beta_1| \leq |\beta_2| \text{ and } \frac{\beta_1^2+\beta_2^2}{2} > \beta_3^2 & \mathcal{J} = \{\{1,2\},\{3\}\} \\ |\beta_1| + \sqrt{2(\beta_2^2+\beta_3^2)}, & \text{if } |\beta_2| \leq |\beta_3| \text{ and } \beta_1^2 > \frac{\beta_2^2+\beta_3^2}{2} & \mathcal{J} = \{\{1\},\{2,3\}\} \\ \sqrt{3(\beta_1^2+\beta_2^2+\beta_3^2)}, & \text{otherwise} & \mathcal{J} = \{\{1,2,3\}\} \end{cases}$$

where we have also displayed the partition $\mathcal{J}$ involved in each case. We also present a graphical representation of the corresponding unit ball in Figure 3.3-a. For comparison we also graphically display the unit ball for the hierarchical Group Lasso with groups $\{1,2,3\},\{2,3\},\{3\}$ and two Group Lasso in Figure 3.3-b,c,d, respectively.

The wedge may equivalently be expressed as the constraint that the difference vector $D^1(\lambda) := (\lambda_{j+1} - \lambda_j : j \in \mathbb{N}_{n-1})$ is less than or equal to zero. This alternative interpretation suggests the k-th order difference operator, which is given by the formula

$$D^k(\lambda) = \left(\lambda_{j+k} + \sum_{\ell \in \mathbb{N}_k} (-1)^\ell \binom{k}{\ell} \lambda_{j+k-\ell} : j \in \mathbb{N}_{n-k} \right)$$

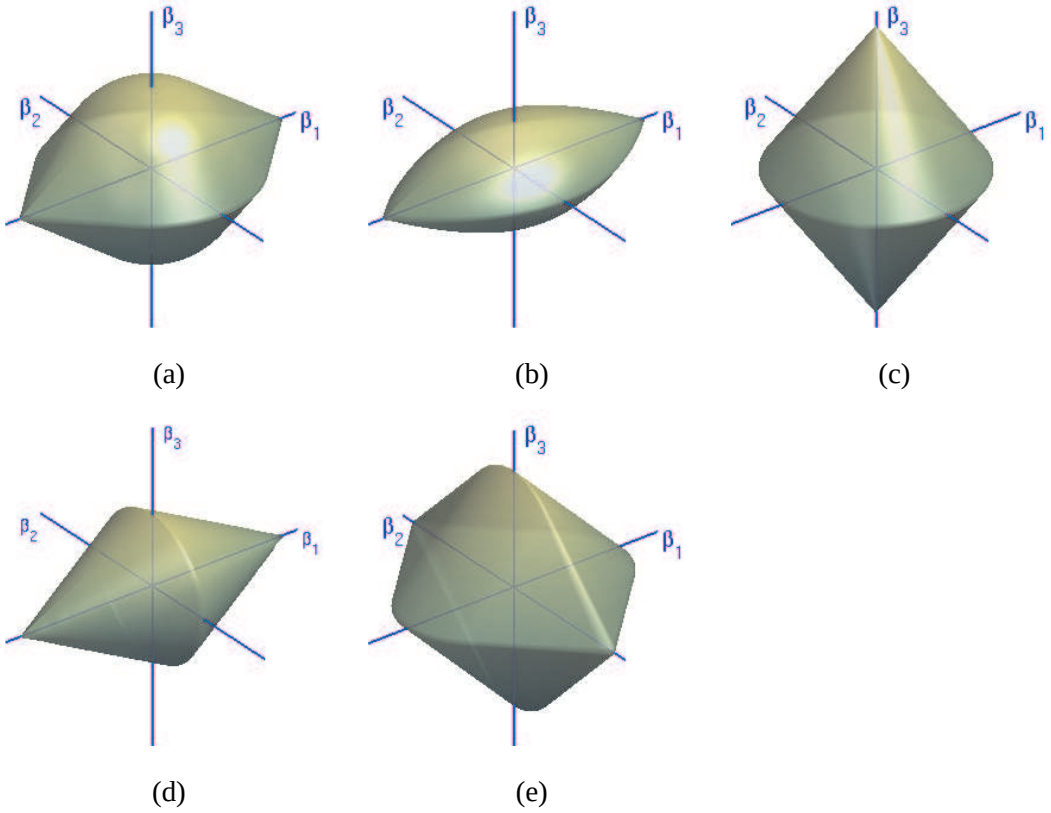


Figure 3.3: Unit ball of different penalty functions: (a) Wedge penalty $\Omega(\cdot|W)$; (b) hierarchical Group Lasso; (c) Group Lasso with groups $\{\{1,2\},\{3\}\}$; (d) Group Lasso with groups $\{\{1\},\{2,3\}\}$; (e) the penalty $\Omega(\cdot|W^2)$.

and the corresponding k-th wedge

$$W^k := \{\lambda : \lambda \in \mathbb{R}^n_{++},\ D^k(\lambda) \geq 0\}. \tag{3.3.11}$$

The associated penalty $\Omega(\cdot|W^k)$ encourages vectors whose sparsity pattern is concentrated on at most k different contiguous regions. Note that W^1 is not the wedge W considered earlier. Moreover, the 2-wedge includes vectors which have a convex "profile" and whose sparsity pattern is concentrated either on the first elements of the vector, on the last, or on both.

3.3.3 Graph penalty

In this section we present an extension of the wedge set which is inspired by previous work on the Group Lasso estimator with hierarchically overlapping groups [58]. It models vectors whose magnitude is ordered according to a graphical structure.

Let $G = (V, E)$ be a directed graph, where V is the set of n vertices in the graph and $E \subseteq V \times V$ is the edge set, whose cardinality is denoted by m. If $(v, w) \in E$ we say that there is a directed edge from vertex v to vertex w. The graph is identified by the $m \times n$ *incidence*

matrix, which we define as

$$A_{e,v} = \begin{cases} 1, & \text{if } e = (v, w) \in E,\ w \in V \\ -1, & \text{if } e = (w, v) \in E,\ w \in V \\ 0, & \text{otherwise.} \end{cases}$$

We consider the penalty $\|\cdot\|_{\Lambda_G}$ for the convex cone $\Lambda_G = \{\lambda : \lambda \in \mathbb{R}^n_{++}, A\lambda \geq 0\}$ and assume, from now on, that G is acyclic (DAG), that is, G has no directed loops. In particular, this implies that, if $(v, w) \in E$ then $(w, v) \notin E$. The wedge penalty described above is a special case of the graph penalty corresponding to a line graph. Let us now discuss some aspects of the graph penalty for an arbitrary DAG. As we shall see, our remarks lead to an explicit form of the graph penalty when G is a tree.

If $(v, w) \in E$ we say that vertex w is a child of vertex v and v is a parent of w. For every vertex $v \in V$, we let $C(v)$ and $P(v)$ be the set of children and parents of v, respectively. When G is a tree, $P(v)$ is the empty set if v is the root node and otherwise $P(v)$ consists of only one element, the parent of v, which we denote by $p(v)$.

Let $D(v)$ be the set of descendants of v, that is, the set of vertices which are connected to v by a directed path starting in v, and let $A(v)$ be the set of ancestors of v, that is, the set of vertices from which a directed path leads to v. We use the convention that $v \in D(v)$ and $v \notin A(v)$.

Every connected subset $V' \subseteq V$ induces a subgraph of G which is also a DAG. If V_1 and V_2 are disjoint connected subsets of V, we say that they are connected if there is at least one edge connecting a pair of vertices in V_1 and V_2, in either one or the other direction. Moreover, we say that V_2 is below V_1 — written $V_2 \Downarrow V_1$ — if V_1 and V_2 are connected and every edge connecting them departs from a node of V_1.

Definition 3.3.4. *Let G be a DAG. We say that $C \subseteq E$ is a cut of G if it induces a partition $\mathcal{V}(C) = \{V_\ell : \ell \in \mathbb{N}_k\}$ of the vertex set V such that $(v, w) \in C$ if and only if vertices v and w belong to two different elements of the partition.*

In other words, a cut separates a connected graph in two or more connected components such that every pair of vertices corresponding to a disconnected edge, that is an element of C, are in two different components. We also denote by $\mathcal{C}(G)$ the set of cuts of G, and by $D_\ell(v)$ the set of descendants of v within set V_ℓ, for every $v \in V_\ell$ and $\ell \in \mathbb{N}_k$. Figure 3.3 illustrates an example of a partition of a tree.

Next, for every $C \in \mathcal{C}(G)$, we define the regions in $\mathbb{R}^n$ by the equations

$$O_C = \bigcap \{Q_{V_1,V_2} :\ V_1, V_2 \in \mathcal{V}(C), V_2 \Downarrow V_1\} \tag{3.3.12}$$

and

$$I_C = \bigcap \left\{ \overline{Q}_{D_\ell(v),V_\ell} : \ell \in \mathbb{N}_k, v \in V_\ell \right\}. \tag{3.3.13}$$

These sets are the graph equivalent of the sets defined by equations (3.3.3) and (3.3.4) in the special case of the wedge penalty in Section 3.3.2. We also set $P_C = O_C \cap I_C$.

Moreover, for every $C \in \mathcal{C}(G)$, we define the sets

$$U_C := P_C \bigcap (\mathbb{R}\backslash\{0\})^n.$$

As of yet, we cannot extend Theorem 3.3.2 to the case of an arbitrary DAG, even if we suspect it to be true. However, we can accomplish this when G is a tree.

Lemma 3.3.2. *Let $G = (V, E)$ be a tree, let A be associated incidence matrix and let $z = (z_v : v \in V) \in \mathbb{R}^n$. The following facts are equivalent:*

(a) For every $v \in V$ it holds that

$$\sum_{u \in D(v)} z_u \geq 0.$$

(b) The linear system $A^\top \alpha = -z$ admits a non-negative solution for $\alpha = (\alpha_e : e \in E) \in \mathbb{R}^m$.

Proof. The incident matrix of a tree has the property that, for every $v \in V$ and $e \in E$,

$$\sum_{u \in D(v)} A_{eu} = -\delta_{e,(p(v),v)} \tag{3.3.14}$$

where, for every $e, e' \in E$, $\delta_{e,e'} = 1$ if $e = e'$ and zero otherwise. The the linear system in (b) can be written componentwise as

$$\sum_{e \in E} A_{eu} \alpha_e = -z_u.$$

Summing both sides of this equation over $u \in D(v)$ and using equation (3.3.14), we obtain the equivalent equations

$$\alpha_{(p(v),v)} = \sum_{u \in D(v)} z_u.$$

The result follows. ∎

Definition 3.3.5. *Let $G = (V, E)$ be a DAG. For every vector $\beta \in (\mathbb{R}\backslash\{0\})^n$ and every cut $C \in \mathcal{C}(G)$ we let $\mathcal{V}(C) = \{V_\ell : \ell \in \mathbb{N}_k\}$, $k \in \mathbb{N}_n$ be the partition of V induced by C, and*

define two vectors $\zeta(\beta, C) \in \mathbb{R}_+^{n-1}$ *and* $\delta(\beta, C) \in \mathbb{R}_{++}^n$. *The components of* $\zeta(\beta, C)$ *are given as*

$$\zeta_e(\beta, C) = \begin{cases} 0, & \text{if } e \in C, \\ |V_\ell| \frac{\|\beta_{|D_\ell(u)}\|_2^2}{\|\beta_{|V_\ell}\|_2^2} - |D_\ell(u)|, & \text{if } e = (u, v), u \in V_\ell, v \in D_\ell(u),\ \ell \in \mathbb{N}_k \end{cases}$$

whereas the components of $\delta(\beta, C)$ *are given by*

$$\delta_v(\beta, C) = \frac{\|\beta_{|V_\ell}\|_2}{\sqrt{|V_\ell|}},\ v \in V_\ell,\ \ell \in \mathbb{N}_k. \tag{3.3.15}$$

Note that the notation we adopt in this definition differs from that used in the case of line graph, given in Definition 3.3.3. However, Definition 3.3.5 leads to a more appropriate presentation of our results for a tree.

Proposition 3.3.1. *Let* $G = (V, E)$ *be a tree and* A *the associated incidence matrix. For every* $\beta \in (\mathbb{R}\backslash\{0\})^n$ *and every cut* $C \in \mathcal{C}(G)$ *we have that*

(a) $\beta \in P_C$ if and only if $\zeta(\beta, C) \geq 0$, $A\delta(\beta, C) \geq 0$ *and* $\delta_v(\beta, C) > \delta_w(\beta, C)$, *for all* $v \in V_1, w \in V_2$, $(v, w) \in E$, $V_1, V_2 \in \mathcal{V}(C)$;

(b) If $\delta(\beta, C_1) = \delta(\beta, C_2)$ *and* $\beta \in O_{C_1} \cap O_{C_2}$ *then* $C_1 = C_2$.

Proof. We immediately see that $\beta \in O_C$ if and only if $A\delta(\beta, C) \geq 0$ and $\delta_v(\beta, C) > \delta_w(\beta, C)$ for all $v \in V_1, w \in V_2$, $(v, w) \in E$, $V_1, V_2 \in \mathcal{V}(C)$. Moreover, by applying Lemma 3.3.2 on each element V_ℓ of the partition induced by C and choosing $z = (|V_\ell| \frac{\beta_v^2}{\|\beta_{|V_\ell}\|_2^2} - 1 : v \in V_\ell)$, we conclude that $\zeta(\beta, C) \geq 0$ if and only if $\beta \in I_C$. This proves the first assertion.

The proof of the second assertion is a direct consequence of the fact that the vector $\delta(\beta, C)$ is a constant on any element of the partition $\mathcal{V}(C)$ and strictly decreasing from one element to the next in that partition. ■

Theorem 3.3.3. *Let* $G = (V, E)$ *be a tree. The collection of sets* $\mathcal{U} := \{U_C : C \in \mathcal{C}(G)\}$ *form a partition of* $(\mathbb{R}\backslash\{0\})^n$. *Moreover, for every* $\beta \in (\mathbb{R}\backslash\{0\})^n$ *there is a unique* $C \in \mathcal{C}(G)$ *such that*

$$\|\beta\|_{\Lambda_G} = \sum_{V_\ell \in \mathcal{V}(\mathcal{C})} \sqrt{|V_\ell|} \|\beta_{|V_\ell}\|_2 \tag{3.3.16}$$

and the vector $\lambda(\beta) = (\lambda_v(\beta) : v \in V)$ *has components given by* $\lambda_v(\beta) = \mu_\ell,\ v \in V_\ell$, $\ell \in \mathbb{N}_k$, *where*

$$\mu_\ell = \sqrt{\frac{1}{n_\ell} \sum_{w \in V_\ell} \beta_w^2}. \tag{3.3.17}$$

Proof. The proof of this theorem proceeds in a fashion similar to that of Theorem 3.3.2. In this regard, Lemma 3.3.2 is crucial. By Karush-Kuhn-Tucker theory (see e.g. [6, Theorems 3.3.4,3.3.7]), λ is an optimal solution of the graph penalty if and only if there exists $\alpha \geq 0$ such that, for every $v \in V$

$$-\frac{\beta_v^2}{\lambda_v^2} + 1 - \sum_{e \in E} \alpha_e A_{ev} = 0$$

and the following complementary conditions hold true

$$\alpha_{(v,w)}(\lambda_w - \lambda_v) = 0,\ v \in V, w \in C(v). \tag{3.3.18}$$

We rewrite the first equation as

$$\alpha_{(p(v),v)} - \sum_{w \in C(v)} \alpha_{(v,w)} = \frac{\beta_v^2}{\lambda_v^2} - 1. \tag{3.3.19}$$

Now, if $\lambda \in \Lambda_G$ solves equations (3.3.18) and (3.3.19), then it induces a cut $C \subset E$ and a corresponding partition $\mathcal{V}(C) = \{V_\ell : \ell \in \mathbb{N}_k\}$ of V such that $\lambda_v = \mu_\ell$ for every $v \in V_\ell$. That is, $\lambda_v = \lambda_w$ for every $v, w \in V_\ell$, $\ell \in \mathbb{N}_k$, and $\alpha_e = 0$ for every $e \in C$. Therefore, summing equations (3.3.19) for $v \in V_\ell$ we get that

$$\mu_\ell = \frac{\|\beta_{|V_\ell}\|_2}{\sqrt{|V_\ell|}}.$$

Moreover, since $\mu_\ell > \mu_q$, if $V_q \Downarrow V_\ell$ we see that $\beta \in O_C$. Next, for every $\ell \in \mathbb{N}_k$ and $u \in V_\ell$ we sum both sides of equation (3.3.19) for $v \in D_\ell(u)$ to obtain that

$$\alpha_{(p(u),u)} = \frac{\|\beta_{|D_\ell(u)}\|_2^2}{\mu_\ell^2} - |D_\ell(u)|. \tag{3.3.20}$$

We see that $\beta \in I_C$ and conclude that $\beta \in U_C$.

In summary we have shown that the collection of sets $\mathcal{U}$ cover $(\mathbb{R}\backslash\{0\})^n$. Next, we show that the elements of $\mathcal{U}$ are disjoint. To this end, we observe that, the computation described above can be *reversed*. That is to say, conversely for *any* partition $C = \{V_1, \ldots, V_k\}$ of V and $\beta \in U_C$ we conclude by Proposition 3.3.1 that the vectors $\delta(\beta, C)$ and $\zeta(\beta, C)$ solves the KKT optimality conditions. Since this solution is unique if $\beta \in U_{C_1} \cap U_{C_2}$ then it must follow that $\delta(\beta, C_1) = \delta(\beta, C_2)$, which implies that $C_1 = C_2$. ■

Theorems 3.3.2 and 3.3.3 fall into the category of a set $\Lambda \subseteq \mathbb{R}^n$ chosen in the form of a polyhedral cone, that is

$$\Lambda = \{\lambda : \lambda \in \mathbb{R}^n, A\lambda \geq 0\}$$

where A is an $m \times n$ matrix. Furthermore, in the line graph of Theorem 3.3.2 and also the extension in Theorem 3.3.3 the matrix A only has elements which are $-1, 1$ or 0. These two

examples that we considered led to explicit description of the norm $\|\cdot\|_\Lambda$. However, there are seemingly simple cases of a matrix A of this type where the explicit computation of the norm $\|\cdot\|_\Lambda$ seem formidable, if not impossible. For example, if $m = 2$, $n = 4$ and

$$A = \begin{bmatrix} -1 & -1 & 1 & 0 \\ 0 & -1 & -1 & 1 \end{bmatrix}$$

we are led by KKT to a system of equations that, in the case of two active constraints, that is, $A\lambda = 0$, are the common zeros of two *fourth order* polynomials in the vector $\lambda \in \mathbb{R}^2$.

3.3.4 Tree-C and Grid-C

We consider two more sets Λ of the form

$$\Lambda = \{\lambda \in \mathbb{R}^n_{++} : A\lambda \in S\}$$

where S is a convex set and A is a $k \times n$ matrix. Two main choices of interest are when S is a convex cone or the unit ball of a norm. We shall refer to the corresponding set Λ as *conic constraint* or *norm constraint* set, respectively. We next discuss two specific examples, which highlight the flexibility of our approach and help us understand the sparsity patterns favoured by each choice.

Within the conic constraint sets, we may choose $S = \mathbb{R}^k_{++}$, so that $\Lambda = \{\lambda \in \mathbb{R}^n_{++} : A\lambda \geq 0\}$, which can be used to encourage hierarchical sparsity. In [32] they considered the set $\Lambda = \{\lambda \in \mathbb{R}^n_{++} : \lambda_1 \geq \cdots \geq \lambda_n\}$ and derived an explicit formula of the corresponding regularizer $\Omega(\beta|\Lambda)$. Note that for a generic matrix A the penalty function cannot be computed explicitly. In Chapter 4, we show how to overcome this difficulty.

Within the norm constraint sets, we may choose S to be the ℓ_1-unit ball and A the edge map of a graph G with edge set E, so that $\Lambda = \left\{\lambda \in \mathbb{R}^n_{++} : \sum_{(i,j)\in E} |\lambda_i - \lambda_j| \leq 1\right\}$. This set can be used to encourage sparsity patterns consisting of few connected regions/subgraphs of the graph G. For example if G is a 1D-grid we have that $\Lambda = \{\lambda \in \mathbb{R}^n_{++} : \sum_{i=1}^{n-1} |\lambda_{i+1} - \lambda_i| \leq 1\}$, so the corresponding penalty will favour vectors β whose absolute values are constant.

3.4 Duality

In this section, we comment on the utility of our class of penalty functions, which is fundamentally based on their construction as constrained infimum of quadratic functions. To emphasize this point both theoretically and computationally, we discuss the conversion of the regularization variational problem over $\beta \in \mathbb{R}^n$, namely

$$\mathcal{E}(\Lambda) = \inf\{E(\beta, \lambda) : \beta \in \mathbb{R}^n, \lambda \in \Lambda\} \tag{3.4.1}$$

where

$$E(\beta, \lambda) := \|y - X\beta\|^2 + 2\rho\Gamma(\beta, \lambda),$$

into a variational problem over $\lambda \in \Lambda$.

To explain what we have in mind, we introduce the following definition.

Definition 3.4.1. *For every $\lambda \in \mathbb{R}^n_+$, we define the vector $\beta(\lambda) \in \mathbb{R}^n$ as*

$$\beta(\lambda) = \mathrm{diag}(\lambda) M(\lambda) X^\top y$$

where $M(\lambda) := (\mathrm{diag}(\lambda) X^\top X + \rho I)^{-1}$.

Note that $\beta(\lambda) = \mathrm{argmin}\{E(\beta, \lambda) : \beta \in \mathbb{R}^n\}$.

Theorem 3.4.1. *For $\rho > 0$, $y \in \mathbb{R}^m$, any $m \times n$ matrix X and any nonempty convex set Λ we have that*

$$\mathcal{E}(\Lambda) = \min \left\{ \rho y^\top \left(X \mathrm{diag}(\lambda) X^\top + \rho I \right)^{-1} y + \rho \mathrm{tr}(\mathrm{diag}(\lambda)) : \lambda \in \overline{\Lambda} \cap \mathbb{R}^n_+ \right\} \tag{3.4.2}$$

Moreover, if $\hat{\lambda}$ is a solution to this problem, then $\beta(\hat{\lambda})$ is a solution to problem (3.4.1).

Proof. We substitute the formula for $\Omega(\beta|\Lambda)$ into the right hand side of equation (3.4.1) to obtain that

$$\mathcal{E}(\Lambda) = \inf \{H(\lambda) : \lambda \in \Lambda\} \tag{3.4.3}$$

where we define

$$H(\lambda) = \min \{E(\beta, \lambda) : \beta \in \mathbb{R}^n\}.$$

A straightforward computation confirms that

$$H(\lambda) = \rho y^\top \left(X \mathrm{diag}(\lambda) X^\top + \rho I \right)^{-1} y + \rho \mathrm{tr}(\mathrm{diag}(\lambda)).$$

Since $H(\lambda) \geq \rho \mathrm{tr}(\mathrm{diag}(\lambda))$, we conclude that any minimising sequence for the optimization problem on the right hand side of equation (3.4.3) must have a subsequence which converges. These remarks confirm equation (3.4.2).

We now prove the second claim. For $\lambda \in \mathbb{R}^n_{++}$ a direct computation confirms that

$$\Gamma(\beta(\lambda), \lambda) = \frac{1}{2} \left(y^\top X M(\lambda) \mathrm{diag}(\lambda) M(\lambda) X^\top y + \mathrm{tr}(\mathrm{diag}(\lambda)) \right).$$

Note that the right hand side of this equation provides a continuous extension of the left hand side to $\lambda \in \mathbb{R}^n_+$. For notational simplicity, we still use the left hand side to denote this *continuous extension*.

By a limiting argument, we conclude, for every $\lambda \in \overline{\Lambda}$, that

$$\Omega(\beta(\lambda)|\Lambda) \leq \Gamma(\beta(\lambda), \lambda). \tag{3.4.4}$$

We are now ready to complete the proof of the theorem. Let $\hat{\lambda}$ be a solution for the optimization problem (3.4.2). By definition, it holds, for any $\beta \in \mathbb{R}^n$ and $\lambda \in \overline{\Lambda}$, that

$$\|y - X\beta(\hat{\lambda})\|^2 + 2\rho\Gamma(\beta(\hat{\lambda}), \hat{\lambda}) = H(\hat{\lambda}) \leq H(\lambda) \leq \|y - X\beta\|^2 + 2\rho\Gamma(\beta, \lambda).$$

Combining this inequality with inequality (3.4.4) evaluated at $\lambda = \hat{\lambda}$, we conclude that

$$\|y - X\beta(\hat{\lambda})\|^2 + 2\rho\Omega(\beta(\hat{\lambda})|\Lambda) \leq \|y - X\beta\|^2 + 2\rho\Gamma(\beta, \lambda)$$

from which the result follows. ■

An important consequence of the above theorem is a method to find a solution $\hat{\beta}$ to the optimization problem (3.4.1) from a solution to the optimization problem (3.4.2). We illustrate this idea in the case that $X = I$.

Corollary 3.4.1. *It holds that*

$$\min\left\{\|\beta - y\|_2^2 + 2\rho\Omega(\beta|\Lambda) : \beta \in \mathbb{R}^n\right\} = \rho \min\left\{\sum_{i\in\mathbb{N}_n} \frac{y_i^2}{\lambda_i + \rho} + \lambda_i : \lambda \in \overline{\Lambda}\right\}. \tag{3.4.5}$$

Moreover, if $\hat{\lambda}$ is a solution of the right optimization problem then the vector $\beta(\hat{\lambda}) = (\beta_i(\hat{\lambda}) : i \in \mathbb{N}_n)$, defined as

$$\beta_i(\hat{\lambda}) = \frac{\hat{\lambda}_i y_i}{\hat{\lambda}_i + \rho} \tag{3.4.6}$$

is a solution of the right problem.

We further discuss some examples of the set Λ for which we are able to solve this problem analytically. If $\Lambda = \mathbb{R}^n_{++}$, for which Ω is the ℓ_1 norm, the solution to problem (3.4.5) is $\hat{\lambda} = (|y| - \rho)_+$, and the corresponding regression vector is obtained by the well-known "soft thresholding" formula $\beta(\hat{\lambda}) = (|y| - \rho)_+ \text{sign}(y)$.

If $\Lambda = [a, b]$, we solve the problem (3.4.5) by appealing to Theorem 3.3.1 and a change of variables. We obtain that $\hat{\lambda}_i = |y_i| - \rho + (a - |y_i| + \rho)_+ - (|y_i| - \rho - b)_+$ for $i \in \mathbb{N}_n$, and we can compute $\beta(\hat{\lambda})$ accordingly.

For the Wedge and Tree penalties we find that the solution to the problem (3.4.5) is

$$\hat{\lambda} = (\lambda(y) - \rho)_+,$$

where $\lambda(y)$ is given by Theorem 3.3.2 or Theorem 3.3.3 respectively. To see why this must be true, we focus on the most general case of the Tree, and we follow the proof of Theorem 3.3.3.

Note that the only difference from the problem solved in that theorem is that now the variables corresponding to the leaves of the graph should be greater than ρ. For this reason, to the slackness conditions of (3.3.18), we have the additional conditions

$$\alpha_v(\rho - \lambda_v) = 0,$$

for all leaves v of the tree graph.

Now let v be a generic leaf node of the graph. In order for λ_v to be a minimiser of the unconditional version of problem (3.4.6), it should cancel the derivative. It means that it should be $\lambda_v = |y_v| - \rho$. Two cases can follow.

If $|y_v| \leq \rho$, and because of the constraint that $\lambda_v \geq 0$, we have to conclude that $\hat{\lambda}_v = 0$, and consequently $\alpha_v > 0$. The problem is thus reduced by one dimension, as we can repeat our reasoning for a different leaf without node v.

If, on the other hand, $|y_v| > \rho$, then the optimal value is achieved for $\hat{\lambda}_n = |y_n| - \rho > 0$. This implies that the corresponding slackness variable is $\alpha_v = 0$, and we can continue to follow the original proof having assigned in either way a value to both the leaves nodes and the new slackness variables.

Finally, we note that Corollary 3.4.1 and the examples following it extend to the case that $X^\top X = I$ by replacing throughout the vector y by the vector $X^\top y$. In the statistical literature this setting is referred to as orthogonal design.

3.5 Special cases

For particular choices of the constraints set Λ, the function Ω reduces to known penalty terms. We have already seen that this is true for the ℓ_1 norm.

For trivial sets like a ray or a point, the function Ω reduces to the ℓ_2 norm or the squared ℓ_2 norm. For another simple structure, where all components of λ are equal inside a group, the Group Lasso penalty with no overlapping groups is recovered.

A dirty model (see [23]) splits the model into the sum of two vectors and penalises each of them indepently using two different functions. For a particular set Λ, we obtain the dirty model ℓ_2^2/ℓ_1, where the ℓ_1 norm penalises an auxiliary vector which is close to the model, the distance being measured with the squared ℓ_2 norm.

We also consider dirty models in which the distance is measured with the function Ω itself. Specifically, we prove that the dirty models Ω/ℓ_1 and Ω/ℓ_2^2, both corresponds to special cases of the function Ω.

The Overlapping groups technique (see [22]) discussed in § 2.3.2 is not in general a special case of the Ω function. However, it can be expressed in a related way and, under some

assumptions on the grouping, can be recovered for some choice of Λ.

In § 3.5.1 we present the special cases of ℓ_2 norm and Group Lasso. The dirty model ℓ_2^2/ℓ_1 is presented in § 3.5.2, while the dirty models involving Ω, that is Ω/ℓ_1 and Ω/ℓ_2^2, appear in § 3.5.3 and § 3.5.4 respectively. In § 3.5.5, the Overlapping groups case is discussed.

3.5.1 Euclidean norm and Group Lasso

ℓ_2 **norm**. The ℓ_2 norm is obtained when the set Λ consists of a single ray. In general, let $\Lambda = \{\lambda : \lambda = av, a > 0\}$, where v is a given and fixed vector in the positive orthant. The value of Ω will be

$$\Omega(\beta|\Lambda) = \frac{1}{2} \inf_{a>0} \left\{ \frac{1}{a} \sum_{i\in\mathbb{N}_n} \frac{\beta_i^2}{v_i} + a \sum_{i\in\mathbb{N}_n} v_i \right\} = \sqrt{\sum_{i\in\mathbb{N}_n} \frac{\beta_i^2}{v_i} \sum_{i\in\mathbb{N}_n} v_i}. \tag{3.5.1}$$

In fact, the lower bound provided by the arithmetic-geometric inequality is achieved by $\hat{a} = \sqrt{\sum_{i\in\mathbb{N}_n} \frac{\beta_i^2}{v_i}} / \sqrt{\sum_{i\in\mathbb{N}_n} v_i}$.

In the special case when vector v has all components equal to a common positive value k, we find that the expressions $\sum_{i\in\mathbb{N}_n} \frac{\beta_i^2}{v_i}$ and $\sum_{i\in\mathbb{N}_n} v_i$ simplify to $\frac{1}{k}\|\beta\|_2^2$ and nk respectively. Then we have

$$\Omega(\beta|\Lambda) = \sqrt{n}\|\beta\|_2, \tag{3.5.2}$$

for $\Lambda = \{\lambda : \lambda = (a, \ldots, a), a > 0\}$.

ℓ_2 **norm squared**. We obtain the square of the ℓ_2 norm when the set Λ is a singleton. When the only element of Λ is a given vector v in the positive orthant, $\Omega(\beta|\Lambda) = \frac{1}{2}\sum_{i\in\mathbb{N}_n} \frac{\beta_i^2}{v_i} + \frac{1}{2}\|v\|_1$.

The special case when $v = (k, \ldots, k)$ for $k > 0$ is again of interest, leading to

$$\Omega(\beta|\Lambda) = \frac{1}{2}\left(\frac{\|\beta\|_2^2}{k} + nk \right), \tag{3.5.3}$$

for $\Lambda = \{(k, \ldots, k)\}$.

Group Lasso. Finally, we note that a normalized version of the Group Lasso penalty [57] is also included in our setting as a special case. If $\{J_\ell : \ell \in \mathbb{N}_k\}$, $k \in \mathbb{N}_n$, form a partition of the index set $\mathbb{N}_n$, the corresponding Group Lasso penalty is defined as

$$\Omega_{\mathrm{GL}}(\beta) = \sum_{\ell\in\mathbb{N}_k} \sqrt{|J_\ell|}\, \|\beta_{|J_\ell}\|_2, \tag{3.5.4}$$

where, for every $J \subseteq \mathbb{N}_n$, we use the notation $\beta_{|J} = (\beta_j : j \in J)$. It is an easy matter to verify that $\Omega_{\mathrm{GL}} = \Omega(\cdot|\Lambda)$ for $\Lambda = \{\lambda : \lambda \in \mathbb{R}^n_{++}, \lambda_j = \theta_\ell,\ j \in J_\ell,\ \ell \in \mathbb{N}_k,\ \theta_\ell > 0\}$.

3.5.2 Dirty model ℓ_2^2/ℓ_1

In this and in the following subsections we show how some particular examples of *dirty* linear models can be recast as special cases of function Ω.

The dirty model framework decomposes the underlying vector β as a sum of two terms, $\beta - \alpha$ and α, which are penalised independently with different penalty terms. The result is in contrast with the corresponding *clean* models, the ones obtained with just one of the two penalties.

As an example, [23] propose to penalise the matrix model of a multiple regression problem with two penalties[2]: $\|\cdot\|_{1,\infty}$, to encourage block sparsity, and $\|\cdot\|_{1,1}$, to encourage standard sparsity. One claim of the paper is that under general conditions on the data matrices, the dirty model outperforms both clean models.

We begin by considering the simple dirty setting for ℓ_2^2/ℓ_1 norms (the ℓ_2 is squared):

$$\min_{\alpha,\beta} \left\{ \|y - X\beta\|_2^2 + \|\beta - \alpha\|_2^2 + \rho\|\alpha\|_1 : \alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n \right\}. \tag{3.5.5}$$

In ridge regression we penalise the vector β with the ℓ_2 norm alone. Here, instead, we use the ℓ_2 norm to penalise the *distance* of β from an auxiliary vector α, which in turn is encouraged to be sparse by the presence of the ℓ_1 norm.

By noting that the loss function is independent from the vector α, we can write the penalty term explicitly as

$$J(\beta) = \min_{\alpha \in \mathbb{R}^n} \left\{ \|\beta - \alpha\|_2^2 + \rho\|\alpha\|_1 \right\}. \tag{3.5.6}$$

This function is called the Moreau envelope, and its solution is the well-known soft-thresholding operator. We will see that this penalty function has the same form of the function Ω for a particular choice of the constraint set Λ. Note that just the ℓ_1 norm is multiplied with the coefficient $\rho > 0$: this is indeed sufficient, as a coefficient for the ℓ_2 norm could be easily factored out of the minimisation problem.

Our first step is to use the variational formulation for the ℓ_1 norm, writing it as

$$\|\alpha\|_1 = \frac{1}{2} \inf_{\mu \in \mathbb{R}^n_{++}} \left\{ \alpha^T M^{-1} \alpha + \sum_{i \in \mathbb{N}_n} \mu_i \right\}$$

where $M^{-1} = \operatorname{diag}(\mu_1^{-1}, \dots, \mu_n^{-1})$. Since the term $\|\beta - \alpha\|_2^2$ is independent from μ, we can bring it inside the infimum and change the order of the optimisations, to obtain

$$J(\beta) = \inf_{\mu \in \mathbb{R}^n_{++}} \left\{ \min_{\alpha \in \mathbb{R}^n} \left\{ \|\beta - \alpha\|_2^2 + \frac{\rho}{2} \alpha^T M^{-1} \alpha \right\} + \frac{\rho}{2} \sum_{i \in \mathbb{N}_n} \mu_i \right\}. \tag{3.5.7}$$

The inner minimisation problem is quadratic, and can be solved component-wise. For a generic index i (here omitted), we need to minimise $(\beta - \alpha)^2 + \frac{\rho}{2\mu}\alpha^2$, which has derivative

[2] With the notation $\|A\|_{a,b}$, we refer to the norm $\|\cdot\|_a$ computed on the vector whose k-th component is the norm $\|\cdot\|_b$ of the k-th column of matrix A.

$-2(\beta-\alpha)+\frac{\rho}{\mu}\alpha$, so we have $\hat{\alpha}=\frac{2\mu}{2\mu+\rho}\beta$ and the value at the minimum is $\frac{\beta^2}{2\mu+\rho}\rho$. The penalty term becomes

$$J(\beta)=\frac{1}{2}\inf_{\mu\in\mathbb{R}^n_{++}}\left\{\sum_{i\in\mathbb{N}_n}\left(\frac{\beta_i^2}{\frac{2\mu_i+\rho}{2\rho}}+\rho\mu_i\right)\right\},$$

and after the change of variables $\tilde{\mu}_i=\frac{\mu_i}{\rho}+\frac{1}{2}$ we have

$$J(\beta)=\frac{1}{2}\inf_{\tilde{\mu}>\frac{1}{2}}\left\{\sum_{i\in\mathbb{N}_n}\left(\frac{\beta_i^2}{\tilde{\mu}_i}+\rho^2\tilde{\mu}_i\right)\right\}-\frac{n}{4}\rho^2.$$

We can exploit the quasi-homogeneous property discussed in § 3.2.7 to reduce this expression to the form of function Ω. Then, by the changing of variables $\lambda_i=\rho\tilde{\mu}_i$, we can write

$$J(\beta)=\rho\Omega(\beta|\Lambda)-\frac{n}{4}\rho^2,$$

where $\Lambda=\{\lambda:2\lambda>\rho\}$.

Indeed, the constraint set Λ produces the Box penalty defined in § 3.3.1, where the constraint set $B[a,b]$ has parameters $a=\frac{\rho}{2}$ and $b\to\infty$. We use the closed formula for the Box penalty with these parameters, so that we can finally rewrite the penalty term of Equation Equation (3.5.6) as

$$J(\beta)=\rho\|\beta\|_1+\sum_{i\in\mathbb{N}_n}\left(\frac{\rho}{2}-|\beta_i|\right)_+^2-\frac{n}{4}\rho^2.$$

We conclude by showing that this function is the same as the Moreau envelope. Since the variables are clearly decomposable, it is sufficient to consider the case $n=1$. Let $J_1(\beta)=(\beta-\hat{\alpha})^2+\rho|\hat{\alpha}|$ be the envelope computed at the proximal operator $\hat{\alpha}=(|\beta|-\frac{\rho}{2})_+\text{sgn}(\beta)$, and $J_2(\beta)=\rho|\beta|+\left(\frac{\rho}{2}-|\beta|\right)_+^2-\frac{\rho^2}{4}$ be the Box penalty function. It is an easy matter to compute $J_1(\beta)=J_2(\beta)=\rho|\beta|-\frac{\rho^2}{4}$ when $\frac{\rho}{2}<|\beta|$, and $J_1(\beta)=J_2(\beta)=\beta^2$ otherwise, obtaining the Huber-like penalty induced by the Box constraint.

3.5.3 Dirty model Ω/ℓ_1

We consider now a more general dirty model. We assume that the vector β can be decomposed into a vector $\beta-\alpha$ which has a structured support and a vector α which is sparse. To exploit this assumption, we penalise the first vector with the function $\Omega(\cdot|\Lambda)$, and the second one with ℓ_1. So, differently from what we did in § 3.5.3, we are now providing a structure using Ω, with an appropriate and unspecified set Λ, instead of using ℓ_2. We study the problem

$$\min_{\alpha,\beta}\left\{\|y-X\beta\|_2^2+\Omega(\beta-\alpha|\Lambda)+\rho\|\alpha\|_1:\alpha\in\mathbb{R}^n,\beta\in\mathbb{R}^n\right\},\tag{3.5.8}$$

and we want to show that it is a special case of regularisation with Ω. We begin our manipulations by expanding the ℓ_1 norm and rearranging the order of operations, thus rewriting the

penalty term as

$$J(\beta) = \inf_{\mu \in \mathbb{R}^n_{++}} \left\{ \min_{\alpha \in \mathbb{R}^n} \left\{ \Omega(\beta - \alpha | \Lambda) + \frac{\rho}{2} \alpha^T M^{-1} \alpha \right\} + \frac{\rho}{2} \sum_{i \in \mathbb{N}_n} \mu_i \right\}.$$

To carry out the computation of the inner minimisation with respect to α we expand the function Ω as well and rearrange to get the subproblem

$$\frac{1}{2} \inf_{\lambda \in \Lambda} \left\{ \min_{\alpha \in \mathbb{R}^n} \left\{ (\beta - \alpha)^T D^{-1} (\beta - \alpha) + \rho \alpha^T M^{-1} \alpha \right\} + \rho \sum_{i \in \mathbb{N}_n} \lambda_i \right\},$$

where the diagonal matrix is defined as $D^{-1} = \operatorname{diag}(\lambda_1^{-1}, \ldots, \lambda_n^{-1})$.

Again, we solve component-wise the quadratic inner minimisation. Omitting the indices, we want to minimise $(\beta - \alpha)^2 \frac{1}{\lambda} + \rho \alpha^2 \frac{1}{\mu}$. Its first derivative is $-2(\beta - \alpha)\frac{1}{\lambda} + 2\rho\frac{\alpha}{\mu}$, which is zero for $\hat{\alpha} = \frac{\mu}{\mu + \rho\lambda}\beta$ and has minimum value $\frac{\rho}{\mu + \rho\lambda}\beta^2$. Then we can rewrite the penalty term as

$$J(\beta) = \frac{1}{2} \inf_{\mu \in \mathbb{R}^n} \inf_{\lambda \in \Lambda} \left\{ \sum_{i \in \mathbb{N}_n} \left(\frac{\beta_i^2}{\mu_i + \rho\lambda_i} \rho + \rho\mu_i + \lambda_i \right) \right\}. \tag{3.5.9}$$

By the arithmetic-geometric mean inequality, $\frac{1}{2}\left(\frac{\beta^2}{\mu+\rho\lambda}\rho + \rho\mu + \lambda\right) \geq |\beta|\sqrt{\rho\frac{\rho\mu+\lambda}{\mu+\rho\lambda}}$. Moreover, if $\rho \leq 1$, then $\frac{\rho\mu+\lambda}{\mu+\rho\lambda} = \rho\frac{\rho\mu+\lambda}{\rho\mu+\rho^2\lambda} \geq \rho$, so the generic element of the sum is no smaller than $\rho|\beta_i|$. Indeed, this lower bound is achieved for $\mu_i = |\beta_i|$ and $\lambda_i = 0$, so

$$J(\beta) = \rho\|\beta\|_1$$

for $\rho \leq 1$ and any Λ containing the origin.

We propose a second way to look at this result. We assume that $\rho \leq 1$, and note that $J(\beta) = \min_{\alpha \in \mathbb{R}^n} \{\Omega(\beta - \alpha|\Lambda) + \rho\|\alpha\|_1\}$ can indeed be $\rho\|\beta\|_1$ for the value $\hat{\alpha} = \beta$. Now we prove that $J(\beta)$ cannot be less than $\rho\|\beta\|_1$, completing the argument. By the properties of Ω, we have that $\Omega(\beta - \alpha|\Lambda) + \rho\|\alpha\|_1 \geq \|\beta - \alpha\|_1 + \rho\|\alpha\|_1 \geq |\|\beta\|_1 - \|\alpha\|_1| + \rho\|\alpha\|_1$, where the second inequality is a consequence of the triangle inequality. If $\|\beta\|_1 \geq \|\alpha\|_1$, then $J(\beta) \geq \|\beta\|_1 + \|\alpha\|_1(\rho - 1) \geq \|\beta\|_1 + \|\beta\|_1(\rho - 1) = \rho\|\beta\|_1$. Similarly, if $\|\beta\|_1 \leq \|\alpha\|_1$, then $J(\beta) \geq \|\alpha\|_1(1 + \rho) - \|\beta\|_1 \geq \|\beta\|_1(1 + \rho) - \|\beta\|_1 = \rho\|\beta\|_1$.

3.5.4 Dirty model Ω/ℓ_2^2

We notice that the mixture of Ω and the squared ℓ_2 norm can be expressed once again as our infimum problem. We want to solve

$$\min_{\alpha,\beta} = \left\{ \|y - X\beta\|_2^2 + \Omega(\beta - \alpha|\Lambda) + \frac{\rho}{2}\|\alpha\|_2^2 : \alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n \right\},$$

so the explicit penalty term on β is

$$\begin{aligned} J(\beta) &= \min_{\alpha\in\mathbb{R}^n}\left\{\Omega(\beta-\alpha|\Lambda)+\frac{\rho}{2}\|\alpha\|_2^2\right\} \\ &= \frac{1}{2}\inf_{\lambda\in\Lambda}\left\{\min_{\alpha\in\mathbb{R}^n}\left\{\sum_{i\in\mathbb{N}_n}\left(\frac{(\beta_i-\alpha_i)^2}{\lambda_i}+\rho\alpha_i^2\right)\right\}+\sum_{i\in\mathbb{N}_n}\lambda_i\right\}. \end{aligned}$$

Again, the inner minimisation problem is quadratic: the first derivative of $\frac{(\beta-\alpha)^2}{\lambda}+\rho\alpha^2$ is $-\frac{2}{\lambda}(\beta-\alpha)+2\rho\alpha$, which is zero for $\hat{\alpha}=\frac{\beta}{1+\rho\lambda}$. The value at the minimum is $\frac{\beta^2}{1+\rho\lambda}\rho$, so the expression for the penalty term can be written as

$$J(\beta)=\frac{1}{2}\inf_{\lambda\in\Lambda}\left\{\sum_{i\in\mathbb{N}_n}\left(\frac{\beta_i^2}{\frac{1}{\rho}+\lambda_i}+\frac{1}{\rho}+\lambda_i\right)\right\}-\frac{n}{2\rho},$$

which becomes in Ω form as

$$J(\beta)=\Omega(\beta|\tilde{\Lambda})-\frac{n}{2\rho},$$

for $\tilde{\Lambda}=\left\{\tilde{\lambda}:\tilde{\lambda}=\frac{1}{\rho}+\lambda,\lambda\in\Lambda\right\}$.

3.5.5 Overlapping groups

The regulariser in [22], discussed in § 2.3.2, can be reformulated in a form *similar* to that of Ω function. We repeat here for reference the definition of the penalty term:

$$\|\beta\|_{\mathcal{J}}=\inf_{v_{J_1},\ldots,v_{J_K}}\left\{\sum_{J\in\mathcal{J}}\|v_J\|_2:\forall J\in\mathcal{J},v_J\in\mathbb{R}^d,\text{supp}\,(v_J)\subseteq J,\sum_{J\in\mathcal{J}}v_J=\beta\right\}, \quad (3.5.10)$$

where $\mathcal{J}=\{J_1,\ldots,J_K\}$ is a set of (possibly overlapping) groups of indices, where each group J_i is a subset of $\mathbb{N}_n$ and each index belong to at least one group, that is $\cup_i J_i=\mathbb{N}_n$.

Then we have the following proposition.

Proposition 3.5.1. *Let $T_j=\{i:j\in J_i\}$, for $j\in\mathbb{N}_n$, be the set of indices of groups containing component j. Then we can write function (3.5.10) with a variational formulation:*

$$\|\beta\|_{\mathcal{J}}=\inf_{\lambda\in\Lambda}\sqrt{\sum_{j\in\mathbb{N}_n}\frac{\beta_j^2}{\lambda_j}}, \quad (3.5.11)$$

where $\Lambda=\left\{\lambda:\lambda\in\mathbb{R}^d,\lambda_j=\sum_{i\in T_j}\mu_i,j\in\mathbb{N}_n,\mu\in\mathbb{R}^K_{++}\right\}$.

This Proposition can be proved by means of the following intermediate result.

Lemma 3.5.1. *If $\mu\in\mathbb{R}^n_{++}$ and $x\in\mathbb{R}$, then*

$$\inf_{y\in\mathbb{R}^n}\left\{\sum_{j\in\mathbb{N}_n}\frac{y_j^2}{\mu_j}:\sum_{j\in\mathbb{N}_n}y_j=x\right\}=\frac{x^2}{\sum_{j\in\mathbb{N}_n}\mu_j}. \quad (3.5.12)$$

Proof. The constraint is equivalent to $x^2 = (\sum_{j\in\mathbb{N}_n} y_j)^2 = \left(\sum_{j\in\mathbb{N}_n} \frac{y_j}{\sqrt{\mu_j}}\sqrt{\mu_j}\right)^2$. As a consequence of Cauchy-Schwarz inequality, this quantity is at most $\left(\sum_{j\in\mathbb{N}_n} \frac{y_j^2}{\mu_j}\right)\left(\sum_{j\in\mathbb{N}_n} \mu_j\right)$, so the value of the infimum cannot be smaller than $\frac{x^2}{\sum_{j\in\mathbb{N}_n}\mu_j}$. Indeed, this lower bound is attained for $\hat{y}_j = x\frac{\mu_j}{\sum_{j\in\mathbb{N}_n}\mu_j}$. ■

We are now able to prove Proposition 3.5.11.

Proof. To prove the proposition, we begin by making two preliminary steps. Firstly we apply the variational formulation to express the sum of terms $\sum_{J\in\mathcal{J}}\|v_J\|_2$ in Equation (3.5.10) as the result of an infimum problem on variable $\mu \in \mathbb{R}^K_{++}$. Secondly we expand each term $(\|v_J\|_2)^2$: if $(v_J)_j$ is the j-th component of vector v_J, then $(\|v_J\|_2)^2 = \sum_{j\in\mathbb{N}_n}(v_J)_j$. The result of these passages is

$$\sum_{J\in\mathcal{J}}\|v_J\|_2 = \frac{1}{2}\inf_{\mu\in\mathbb{R}^K_{++}}\left\{\sum_{i\in\mathbb{N}_K}\left(\sum_{j\in\mathbb{N}_n}\left(\frac{(v_{J_i})_j^2}{\mu_i}\right)+\mu_i\right)\right\}.$$

We use this formulation in the definition of problem (3.5.10), and proceed by inverting the order of the summations. Now note that, because of the constraint $\text{supp}\,(v_J) \subseteq J$, we can restrict the inner sum to only the groups that contain the particular index j of the outer sum. Moreover, for this reason we can rewrite the constraint $\sum_{J\in\mathcal{J}} v_J = \beta$ as the n constraints $\sum_{i\in T_j}(v_{J_i})_j = \beta_j$, for all $j \in \mathbb{N}_n$.

By further interchanging the order of the infimum operations, we will obtain n indipendent inner subproblems that can be solved as a direct application of Lemma 3.5.12:

$$\inf_{\{v_{J_i}\}_{i\in T_j}}\left\{\sum_{i\in T_j}\frac{(v_{J_i})_j^2}{\mu_i} : \sum_{i\in T_j}(v_{J_i})_j = \beta_j\right\} = \frac{\beta_j^2}{\sum_{i\in T_j}\mu_i}.$$

So that the original problem is now

$$\|\beta\|_{\mathcal{J}} = \frac{1}{2}\inf_{\mu\in\mathbb{R}^K_{++}}\left\{\sum_{j\in\mathbb{N}_n}\left(\frac{\beta_j^2}{\sum_{i\in T_j}\mu_i}\right)+\sum_{i\in\mathbb{N}_K}\mu_i\right\}. \tag{3.5.13}$$

As a result of the arithmetic-geometric mean inequality, the quantity cannot be lower than $\sqrt{\sum_{j\in\mathbb{N}_n}\left(\beta_j^2/\sum_{i\in T_j}\mu_i\right)\sum_{i\in\mathbb{N}_K}\mu_i}$, so we can change the expression inside the infimum. After the change of variables $\lambda_j = \sum_{i\in T_j}\mu_i/\sum_{i\in\mathbb{N}_K}\mu_i$, the claim of the Proposition follows. Note that the set Λ is not restricted by considering elements with components $\lambda_j = \sum_{i\in T_j}\mu_i$ for nonnegative components of μ, as they will be proportional to $1/\sum_{i\in\mathbb{N}_K}\mu_i$. ■

In general Equation (3.5.13) cannot be written in the same form of function Ω. However, it become possible to do so for a special case of the grouping $\mathcal{J}$. Specifically, we assume that each component j belongs to exactly m groups. This assumption is not artificial, as it fits for

example the description of Graph Lasso [22] for particular graphs, i.e. when the graph is a clique and there is a group of variables for each sub-clique.

We can use the property of § 3.2.7 to multiply the sum of variables $\sum_{i\in\mathbb{N}_K}\mu_i$ by m (and at the same time to get rid of the $\frac{1}{n}$ coefficient). This restores the number of occurrences of each variables μ_i, so by the change of variables $\lambda_j = \sum_{i\in T_j}\mu_i$ we get precisely the expression for function Ω:

$$\|\beta\|_{\mathcal{J}} = \sqrt{\frac{n}{m}}\Omega(\beta|\Lambda) \tag{3.5.14}$$

with $\Lambda = \left\{\lambda : \lambda \in \mathbb{R}^n_{++}, \lambda_j = \sum_{i\in T_j}\mu_i, j \in \mathbb{N}_n, \mu \in \mathbb{R}^K_{++}\right\}$.

Chapter 4

Numerical algorithms

In Chapter 3, we proposed the problem of penalising a loss function with the function Ω, specially designed for structured sparsity. This problem is in general hard to solve and all-purpose toolboxes that rely on common techniques are slow and cannot handle a large number of dimensions. In this chapter we address the issue of implementing the learning method (3.1.1) numerically for some particular cases. We present two algorithms that can be used for the Wedge and Tree penalties described in § 3.3.2 and § 3.3.3 and for the norm and conic constraint sets described in § 3.3.4. Using these algorithms, our technique becomes feasible in practice.

A natural approach is an algorithm that minimises in an alternating way with respect to the two blocks of variables, β and λ. As we focus on the square loss function, the minimisation with respect to β is trivial to compute. The minimisation with respect to λ is a subproblem which is not in general easy. However, for the mentioned special cases of the Wedge and the Tree penalties, we can use theoretical results from Chapter 3 to solve it efficiently. For the Wedge, the running time of the subproblem is linear in the number of dimensions. The overall alternating algorithm has good performances.

Our second proposed algorithm, NEPIO, is a proximal method based on a numerical computation of the proximity operator. As we will see, the proximity operator can be computed as the fixed point of a particular linear operator. Convergence to the fixed point can be stopped earlier to allow for an efficient computation, which is approximate but sufficient for the whole algorithm to converge. We apply this algorithm to the norm and conic constraint sets. The alternating algorithm can handle only the Wedge and the Tree penalties, which are indeed special cases of the conic constraint set. While this algorithm is faster for a small number of dimensions, NEPIO scales better and can handle a larger number of dimensions. Moreover, NEPIO can handle the norm constraint set.

We describe the alternating algorithm in Section 4.1, and NEPIO in Section 4.2.

4.1 Alternating algorithm

In this section we describe a natural blockwise coordinate descent algorithm inspired from [1]. This approach updates the minimiser by considering variables λ and β independently, minimising alternatingly with respect to both vectors. This algorithm introduces the subproblem of minimising with respect to λ, i.e. computing the value of function Ω. In general this is not easy, but for the cases of the Wedge and the Tree penalties we can appeal to Theorems 3.3.2 and 3.3.3 to do so. In order to apply these theorems, we need to be able to compute a partition of vector β that satisfies some conditions, which are discussed in § 3.3.3 and § 3.3.2. We present an efficient partitioning algorithm that can be used to solve this task for the two cases.

In § 4.1.1, we describe completely the alternating algorithm and prove that it converges. In § 4.1.2 we discuss the step which minimises with respect to β. In § 4.1.3, we discuss the algorithms for the subproblems of minimising with respect to λ.

4.1.1 Description and convergence

Since the penalty function $\Omega(\cdot|\Lambda)$ is constructed as the infimum of a family of quadratic regularisers, the optimisation problem (3.1.1) reduces to a simultaneous minimisation over the vectors β and λ. For a fixed $\lambda \in \Lambda$, the minimum over $\beta \in \mathbb{R}^n$ is a standard Tikhonov regularisation and can be solved directly in terms of a matrix inversion. For a fixed β, the minimisation over $\lambda \in \Lambda$ requires computing the penalty function (3.1.2). These observations naturally suggests an alternating minimisation algorithm, which has already been considered in special cases in [1]. To describe our algorithm we choose $\epsilon > 0$ and introduce the mapping $\phi^\epsilon : \mathbb{R}^n \to \mathbb{R}^n_{++}$, whose i-th coordinate at $\beta \in \mathbb{R}^n$ is given by

$$\phi^\epsilon_i(\beta) = \sqrt{\beta_i^2 + \epsilon}.$$

For $\beta \in (\mathbb{R}\backslash\{0\})^n$, we also let $\lambda(\beta) = \operatorname{argmin}\{\Gamma(\beta, \lambda) : \lambda \in \Lambda\}$.

The alternating minimisation algorithm is defined as follows: choose, $\lambda_0 \in \Lambda$ and, for $k \in \mathbb{N}$, define the iterates

$$\beta^k = \beta(\lambda^{k-1}) \tag{4.1.1}$$

$$\lambda^k = \lambda(\phi^\epsilon(\beta^k)). \tag{4.1.2}$$

The following theorem establishes convergence of this algorithm.

Theorem 4.1.1. *If the set Λ is admissible in the sense of Definition 3.2.1, then the iterations (4.1.1)–(4.1.2) converges to a vector $\gamma(\epsilon)$ such that*

$$\gamma(\epsilon) = \operatorname{argmin}\left\{\|y - X\beta\|_2^2 + 2\rho\Omega(\phi^\epsilon(\beta)|\Lambda) : \beta \in \mathbb{R}^n\right\}.$$

Moreover, any convergent subsequence of the sequence $\{\gamma\left(\frac{1}{\ell}\right) : \ell \in \mathbb{N}\}$ *converges to a solution of the optimisation problem* (3.1.1).

Proof. We divide the proof into several steps. To this end, we define

$$E_\epsilon(\beta, \lambda) := \|y - X\beta\|^2 + 2\rho\Gamma(\phi^\epsilon(\beta), \lambda)$$

and note that $\beta(\lambda) = \operatorname{argmin}\{E_\epsilon(\alpha, \lambda) : \alpha \in \mathbb{R}^n\}$.

Step 1. We define two sequences, $\theta_k = E_\epsilon(\beta^k, \lambda^{k-1})$ and $\nu_k = E_\epsilon(\beta^k, \lambda^k)$ and observe, for any $k \geq 2$, that

$$\nu_k \leq \theta_k \leq \nu_{k-1}. \tag{4.1.3}$$

These inequalities follow directly from the definition of the alternating algorithm, see equations (4.1.1) and (4.1.2).

Step 2. We define the compact set $B = \{\beta : \beta \in \mathbb{R}^n, \|\beta\|_1 \leq \theta_1\}$. From the first inequality in Proposition 3.2.2 and inequality (4.1.3) we conclude, for every $k \in \mathbb{N}$, that $\beta^k \in B$.

Step 3. We define the function $g : \mathbb{R}^n \to \mathbb{R}$ at $\beta \in \mathbb{R}^n$ as

$$g(\beta) = \min\left\{E_\epsilon(\alpha, \lambda(\phi^\epsilon(\beta))) : \alpha \in \mathbb{R}^n\right\}.$$

We claim that g is continuous on B. In fact, there exists a constant $\kappa > 0$ such that, for every $\gamma^1, \gamma^2 \in B$, it holds that

$$|g(\gamma^1) - g(\gamma^2)| \leq \kappa\|\lambda(\phi^\epsilon(\gamma^1)) - \lambda(\phi^\epsilon(\gamma^2))\|_\infty. \tag{4.1.4}$$

The essential ingredient in the proof of this inequality is the fact that there exists constant $\overline{a}$ and $\overline{b}$ such that, for all $\beta \in B$, $\lambda(\phi^\epsilon(\beta)) \in [\overline{a}, \overline{b}]^n$. This follows from the inequalities developed in the proof of Proposition 3.2.1.

Step 4. By step 2, there exists a subsequence $\{\beta^{k_\ell} : \ell \in \mathbb{N}\}$ which converges to $\tilde{\beta} \in B$ and, for all $\beta \in \mathbb{R}^n$ and $\lambda \in \Lambda$, it holds that

$$E_\epsilon(\tilde{\beta}, \lambda(\phi^\epsilon(\tilde{\beta}))) \leq E_\epsilon(\beta, \lambda(\phi^\epsilon(\tilde{\beta}))), \quad E_\epsilon(\tilde{\beta}, \lambda(\phi^\epsilon(\tilde{\beta}))) \leq E_\epsilon(\tilde{\beta}, \lambda). \tag{4.1.5}$$

Indeed, from step 1 we conclude that there exists $\psi \in \mathbb{R}_{++}$ such that

$$\lim_{k\to\infty} \theta_k = \lim_{k\to\infty} \nu_k = \psi.$$

Since, by Proposition 3.2.1 $\lambda(\beta)$ is continuous for $\beta \in (\mathbb{R}\backslash\{0\})^n$, we obtain that

$$\lim_{\ell\to\infty} \lambda^{k_\ell} = \lambda(\phi^\epsilon(\tilde{\beta})).$$

By the definition of the alternating algorithm, we have, for all $\beta \in \mathbb{R}^n$ and $\lambda \in \Lambda$, that

$$\theta_{k+1} = E_\epsilon(\beta^{k+1}, \lambda^k) \leq E_\epsilon(\beta, \lambda^k), \quad \nu_k = E_\epsilon(\beta^k, \lambda^k) \leq E_\epsilon(\beta^k, \lambda).$$

From this inequality we obtain, passing to limit, inequalities (4.1.5).

Step 5. The vector $(\tilde{\beta}, \lambda(\phi^\epsilon(\tilde{\beta}))$ is a stationary point. Indeed, since Λ is admissible, by step 3, $\lambda(\phi^\epsilon(\tilde{\beta}) \in \mathrm{int}(\Lambda)$. Therefore, since E_ϵ is continuously differentiable this claim follows from step 4.

Step 6. The alternating algorithm converges. This claim follows from the fact that E_ϵ is strictly convex. Hence, E_ϵ has a unique global minimum in $\mathbb{R}^n \times \Lambda$, which in virtue of inequalities (4.1.5) is attained at $(\tilde{\beta}, \lambda(\phi^\epsilon(\tilde{\beta})))$.

The last claim in the theorem follows from the fact that the set $\{\gamma(\epsilon) : \epsilon > 0\}$ is bounded and the function $\lambda(\beta)$ is continuous. ■

4.1.2 Solving the quadratic in β

At each iteration of the alternating algorithm, we minimise the objective function with respect to β. We consider, as a function of β, the quadratic

$$\frac{1}{m}\|y - X\beta\|_2^2 + \gamma\beta D^{-1}\beta, \tag{4.1.6}$$

where $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. The minimiser of (4.1.6) depends on the tuning parameter γ, and it is easily found by setting its first derivative to zero:

$$\hat{\beta}_\gamma = \left(\frac{2}{m}X^T X + \gamma D^{-1}\right)^{-1} \frac{2}{m}X^T y. \tag{4.1.7}$$

We can compute explicitly the limit as $\gamma \to 0$, that is the case of interpolation. By factoring $D^{-1} = D^{-1/2}D^{-1/2}$ and applying the inverse property for a product, we can use the definition of pseudoinverse[1] of the matrix $XD^{1/2}$. Finally we have

$$\hat{\beta}_0 = D^{1/2}\left(XD^{1/2}\right)^\dagger y. \tag{4.1.8}$$

The repeated computation, for each iteration, of (4.1.7) or (4.1.8) is expensive. We can apply a "Kernel trick" to improve performances (for background, see for instance [47]). We begin by considering the problem of the ridge regression:

$$\frac{1}{m}\|y - X\beta\|_2^2 + \gamma\beta^T\beta. \tag{4.1.9}$$

The use of the term $X\beta$ in the loss function can be interpreted as the use of the trivial feature map $\phi(x_i) = x_i$, for $i = 1, \ldots, n$. Under this assumption, the representer theorem assures that

[1] For a matrix A, its pseudoinverse is $A^\dagger = \lim_{\gamma\to 0}(A^T A + \gamma I)^{-1}A^T$.

the solution to the problem will be of the form $X^T c$ for an m-dimensional vector of coefficients c. The new function can be written as $\frac{1}{m}\|y - Gc\|_2^2 + \gamma c^T G c$, where $G = XX^T$ is the Gram matrix. The value of c which minimises this function is $\hat{c} = (G + m\gamma I)^{-1} y$, and $\hat{\beta} = X^T \hat{c}$.

When the additional term is $\beta^T D^{-1} \beta$, as in (4.1.6), we can repeat the simpler case with the change of variables $X \mapsto XD^{1/2}$ and $\beta \mapsto D^{-1/2}\beta$. We then revert the solution $\hat{\beta} = X^T \hat{c}$ to the original variables to get finally

$$\hat{\beta}_\gamma = DX^T \left(XDX + \frac{m}{2}\gamma I \right)^{-1} y. \tag{4.1.10}$$

Note that $\hat{\beta}_\gamma$ as computed in (4.1.8) requires the inversion of an $n \times n$ matrix, while the same vector as computed with (4.1.10) requires the inversion of an $m \times m$. This is very appealing, as we are interested in the case $m \ll n$.

4.1.3 Computation of special penalties

The most challenging step in the alternating algorithm is the computation of the vector λ. Fortunately, if Λ is a second order cone, problem (3.1.2) defining the penalty function $\Omega(\cdot|\Lambda)$ may be reformulated as a second order cone program (SOCP), see e.g. [10]. To see this, we introduce an additional variable $t \in \mathbb{R}^n$ and note that

$$\Omega(\beta|\Lambda) = \min \left\{ \sum_{i \in \mathbb{N}_n} t_i + \lambda_i : \|(2\beta_i, t_i - \lambda_i)\|_2 \leq t_i + \lambda_i, t_i \geq 0,\ i \in \mathbb{N}_n,\ \lambda \in \Lambda \right\}.$$

In particular, the examples discussed in Sections 3.3.2 and 3.3.3, the set Λ is formed by linear constraints and, so, problem (3.1.2) is an SOCP. We may then use available toolboxes to compute the solution of this problem. However, in special cases the computation of the penalty function may be significantly facilitated by using available analytical formulae. Here, we describe how to do this in the case of the Wedge penalty, followed by the Tree penalty.

Wedge penalty. As described in Theorem 3.3.2, it is possible to compute the vector $\lambda(\beta)$ given a partition $\mathcal{J} = \{J_\ell : \ell \in \mathbb{N}_k\}$ which satisfies two conditions presented in (3.3.3) and (3.3.4). We repeat them here for reference. The "cross over" condition is satisfied if

$$\frac{\|\beta_{|J_\ell}\|_2^2}{|J_\ell|} > \frac{\|\beta_{|J_{\ell+1}}\|_2^2}{|J_{\ell+1}|}$$

for each index $\ell < k$, while the "stay within" condition is satisfied if

$$\frac{\|\beta_{|J_\ell}\|_2^2}{|J_\ell|} \geq \frac{\|\beta_{|K}\|_2^2}{|K|},$$

where K is each possible subset of the generic set J_ℓ formed by its first $|K| < |J_\ell|$ components.

Algorithm 4.1 Iterative algorithm to compute the wedge partition

Input: $\beta \in \mathbb{R}^n$

Initialisation: $k \leftarrow 0$

`for` $t = 1$ `to` n `do`

$J_{k+1} \leftarrow \{t\}$

$k \leftarrow k + 1$

`while` $k > 1$ `and` $\frac{\|\beta_{|J_{k-1}}\|_2}{\sqrt{|J_{k-1}|}} \leq \frac{\|\beta_{|J_k}\|_2}{\sqrt{|J_k|}}$

$J_{k-1} \leftarrow J_{k-1} \cup J_k$

$k \leftarrow k - 1$

`end`

`end`

Output: $J_1, \ldots, J_k$

Note that these conditions define a unique partition $\mathcal{J}$ which depends on the vector β. Also, the number of groups in the partition is not known a priori, and its construction is not obvious. To this end, we present an efficient algorithm, which is summarised in Algorithm 4.1.

For the purpose of describing the partitioning algorithm in the case of the Wedge, we define a vector $\beta \in \mathbb{R}^n$ to be admissible if, for every $k \in \mathbb{N}_n$, it holds that $\|\beta_{|\mathbb{N}_k}\|_2/\sqrt{k} \leq \|\beta\|_2/\sqrt{n}$. The proof of the next lemma is straightforward and we do not elaborate on the details.

Lemma 4.1.1. *If $\beta \in \mathbb{R}^n$ and $\delta \in \mathbb{R}^p$ are admissible and $\|\beta\|_2/\sqrt{n} \leq \|\delta\|_2/\sqrt{p}$ then (β, δ) is admissible.*

Algorithm 4.1 processes the components of vector β in a sequential manner. Initially, the very first component forms the only set in the partition. After the generic iteration $t-1$, where the partition is composed of k sets, the index of the next component, t, is put in a new set J_{k+1}. Two cases can occur: the means of the squares of the sets are in strict descending order, or this order is violated by the last set. The latter is the only case that requires further action, so the algorithm merges the last two sets and repeats until the sets in the partition are fully ordered. Note that, since the only operation performed by the algorithm is the merge of admissible sets, Lemma 4.1.1 ensures that after each step t the current partition satisfies the "stay within" conditions. Moreover, the *while* loop ensures that after each step the current partition satisfies, for every $\ell \in \mathbb{N}_{k-1}$, the "cross over" conditions. Thus, the output of the algorithm is the partition $\mathcal{J}$ defined in Theorem 3.3.2. In the actual implementation of the algorithm, the means of squares of each set can be saved. This allows us to compute the mean of squares of a merged set as a weighted mean, which is a constant time operation. Since there are $n-1$ consecutive

terms in total, this is also the maximum number of merges that the algorithm can perform. Each merge requires exactly one additional test, so we can conclude that the running time of the algorithm is linear. In the experimental section (see Figure 5.14 in § 5.4), we will show an empirical validation of this last remark.

Tree penalty. The case of the Tree penalty is similar because, as described in Theorem 3.3.3, we can compute $\lambda(\beta)$ from a certain partition $\mathcal{J} = \{J_1, \dots J_k\}$ which depends on β alone. Summarised in Algorithm 4.2, we present an iterative algorithm to find this partition .

The two conditions satisfied by partition $\mathcal{J}$ are analogous to the "stay within" and "cross over" conditions described earlier, see (3.3.12) and (3.3.13). These conditions are a generalisation of the ones for the line graph, taking into account the more complex topology of the tree graph. For the "stay within" condition, for each group J_ℓ, instead of considering the first few components starting from the left, we now consider the first few components starting from the root and reaching each node. For the "cross over" condition, instead of pairs of consecutive sets J_ℓ and $J_{\ell+1}$, we now consider pairs of sets such that one is "under" the other, $J_1 \Downarrow J_2$, meaning that there is an arch from one node in J_1 directed to one node in J_2.

Here again, the algorithm processes the components of vector β in a sequential manner. Initially, each leaf of the tree is a singleton of the partition $\mathcal{J}$. All the other components are considered following an order which can be precomputed, and which is such that all nodes (except the leaves) will be traversed in inverse depth order, so that the root node will always be the last node. This order is fundamental for the algorithm, as it ensures that, when a node is considered, all the nodes and groups of nodes that are "under" it are stable. At the generic iteration, S will be the set of elements in the current partitions that are "under" the current set J_{NEW}. This set is then tested against the element of S with higher value, and a merge can then occur.

The complexity of this algorithm depends on the topology of the tree, i.e. on its depth and its branching factor. While, as we have seen, it can run in linear time for the Line graph, its performances slow down as the tree becomes more complex. The algorithm is still competitive in the case we have tested, where each node has four children. See Figure 5.14 in § 5.4 for an efficiency experiment.

Note that this algorithm can be parallelised easily by taking advantage of the structure of the tree graph. In our example, the four trees having as root one child of the original root can be partitioned simultaneously by four instances of the algorithm. Finally, the original root will be added to the four results using the same procedure. We did not test this technique because

Algorithm 4.2 Iterative algorithm to compute the tree partition

Input: $\beta \in \mathbb{R}^n$, tree graph G

Initialisation: $L \leftarrow$ Leaves(G), $k \leftarrow |L|$; $J_i \leftarrow \{L_i\}$ for $i = 1, \ldots, k$; order $\subseteq \mathbb{N}^{n-k}$ (see text)

`for` $t \in$ order `do`

stable $\leftarrow 0$; $J_{\text{NEW}} \leftarrow \{t\}$

`while not` stable

$S \leftarrow \{J \in \{J_1, \ldots, J_k\} : J \Downarrow J_{\text{NEW}}\}$

`if` $|S| = 0$ `then`

stable $\leftarrow 1$

`end`

$J_{\text{MAX}} \leftarrow \underset{J \in S}{\operatorname{argmax}} \frac{\|\beta_{|J}\|_2}{\sqrt{|J|}}$

`if` $\frac{\|\beta_{|J_{\text{NEW}}}\|_2}{\sqrt{|J_{\text{NEW}}|}} \leq \frac{\|\beta_{|J_{\text{MAX}}}\|_2}{\sqrt{|J_{\text{MAX}}|}}$ `then`

$J_{\text{NEW}} \leftarrow J_{\text{NEW}} \cup J_{\text{MAX}}$

$J_{\text{MAX}} \leftarrow$ NULL

$k \leftarrow k - 1$

`else`

stable $\leftarrow 1$

`end`

`end`

$k \leftarrow k + 1$

$J_k \leftarrow J_{\text{NEW}}$

`end`

Output: $J_1, \ldots, J_k$

the proposed sequential one was fast enough for our purposes.

4.2 Proximal methods

In this section, we discuss how to solve problem (3.1.1) using an accelerated first-order method that scales linearly with respect to the problem size, as we will show in the experiments in Chapter 5.

Proximal methods rely on the computation of the proximity operator of the function Γ restricted to $\mathbb{R}^n \times \Lambda$. In some cases, like the Wedge and Tree penalties, this operator can be computed exactly. In general, though, this computation is not possible or too expensive.

We consider the constraints set Λ defined in § 3.3.4. We argue that in this case the proxim-

ity operator corresponds to the fixed point of a linear map, and show that this fixed point can be computed efficiently, hence recovering the proximity operator to be used in the main algorithm.

In § 4.2.1 we describe the computation of the proximity operator as fixed point of a linear map. In § 4.2.2 we describe how to incorporate this proximity operator to an accelerated proximal method.

4.2.1 Computation of the Proximity Operator

We want to solve the optimisation problem

$$\inf \left\{ \frac{1}{2}\|X\beta - y\|_2^2 + \rho\,\Gamma(\beta, \lambda) : \beta \in \mathbb{R}^n, \lambda \in \Lambda \right\} \tag{4.2.1}$$

under the general assumption that $\Lambda = \{\lambda \in \mathbb{R}^n_{++} : A\lambda \in S\}$. Note that the loss function is here divided by 2 while in (4.1.6) it was divided by the sample size m. This is done just to simplify the exposition: it has no effects on the solution because a positive coefficient applied to the loss function is absorbed by the tuning of parameter ρ.

The proximity operator for a function $\omega : \mathbb{R}^d \to \mathbb{R}$, and computed at a point $x \in \mathbb{R}^d$, is defined as

$$\text{prox}_\omega(x) = \operatorname{argmin} \left\{ \frac{1}{2}\|y - x\|_2^2 + \omega(y) : y \in \mathbb{R}^d \right\}.$$

According to this definition, the proximity operator of Γ at $(\alpha, \mu) \in \mathbb{R}^n \times \mathbb{R}^n$ is the solution of the problem

$$\min \left\{ \frac{1}{2}\|(\beta, \lambda) - (\alpha, \mu)\|_2^2 + \rho\,\Gamma(\beta, \lambda) : \beta \in \mathbb{R}^n, \lambda \in \Lambda \right\}. \tag{4.2.2}$$

For any fixed λ, a direct computation yields that the objective function in (4.2.2) attains its minimum at

$$\beta_i(\lambda) = \frac{\alpha_i \lambda_i}{\lambda_i + \rho}, \tag{4.2.3}$$

which can be used to rewrite (4.2.2) into the simplified problem

$$\min \left\{ \frac{1}{2}\|\lambda - \mu\|^2 + \frac{\rho}{2} \sum_{i=1}^{n} \left(\frac{\alpha_i^2}{\lambda_i + \rho} + \lambda_i \right) : \lambda \in \Lambda \right\}. \tag{4.2.4}$$

This problem can still be interpreted as a proximity map computation, and we discuss how to solve it with a fixed-point algorithm.

In addition to our general assumption that $\Lambda = \{\lambda \in \mathbb{R}^n_{++} : A\lambda \in S\}$, we assume that the projection of the set S can be easily computed. This latter assumption holds for the cases of Tree-C and Grid-C constraints.

The key step to compute the proximity operator is to rewrite it as a composition of functions. To this end, we define the $(n + k) \times n$ matrix

$$B = \begin{bmatrix} I \\ A \end{bmatrix}$$

and the function $\varphi(s,t) = \varphi_1(s) + \varphi_2(t)$, for $(s,t) \in \mathbb{R}^n \times \mathbb{R}^k$, where

$$\varphi_1(s) = \frac{\rho}{2} \sum_{i \in \mathbb{N}_n} \left(\frac{\alpha_i^2}{s_i + \rho} + s_i + \delta_{\mathbb{R}_{++}}(s_i) \right),$$

and $\varphi_2(t) = \delta_S(t)$. With the notation $\delta_S(\cdot)$ we refer to the indicator function: if $C \subseteq \mathbb{R}^n$, then $\delta_C : \mathbb{R}^n \to \mathbb{R}$ is the function which is 0 if $x \in C$ and $+\infty$ otherwise. Note that the solution of problem (4.2.4) is the same as the proximity map of the linearly composite function $\varphi \circ B$ at μ, which solves the problem

$$\min \left\{ \frac{1}{2}\|\lambda - \mu\|^2 + \varphi(B\lambda) : \lambda \in \mathbb{R}^n \right\}.$$

Variable λ now must not satisfy any constraint, because they have been logically transferred inside the indicator functions. Nevertheless, this new problem does not seem easier to solve. It turns out, however, that if the proximity map of the function φ has a simple form, the following theorem adapted from [33, Theorem 3.1] can be used to accomplish this task. For ease of notation we set $d = n + k$.

Theorem 4.2.1. *Let φ be a convex function on $\mathbb{R}^d$, B a $d \times n$ matrix, $\mu \in \mathbb{R}^n$, $c > 0$, and define the mapping $H : \mathbb{R}^d \to \mathbb{R}^d$ at $v \in \mathbb{R}^d$ as*

$$H(v) = (I - \mathit{prox}_{\frac{\varphi}{c}})((I - cBB^\top)v + B\mu).$$

Then, for any fixed point $\hat{v}$ of H, it holds that

$$\mathit{prox}_{\varphi \circ B}(\mu) = \mu - cB^\top \hat{v}. \tag{4.2.5}$$

The *Picard iterates* $\{v_s : s \in \mathbb{N}\} \subseteq \mathbb{R}^d$, starting at $v_0 \in \mathbb{R}^d$, are defined by the recursive equation $v_s = H(v_{s-1})$. Since the operator $I - \mathrm{prox}_\varphi$ is *nonexpansive*[2] (see e.g. [12]), the map H is nonexpansive if $c \in \left[0, \frac{2}{||B||^2}\right]$. Because of this, the Picard iterates are not guaranteed to converge to a fixed point of H. However, a simple modification with an averaging scheme can be used to compute the fixed point.

Theorem 4.2.2. *[38] Let $H : \mathbb{R}^d \to \mathbb{R}^d$ be a nonexpansive mapping which has at least one fixed point and let $H_\kappa := \kappa I + (1-\kappa)H$. Then, for every $\kappa \in (0,1)$, the Picard iterates of H_κ converge to a fixed point of H.*

[2] A mapping $T : \mathbb{R}^d \to \mathbb{R}^d$ is called nonexpansive if $\|T(v) - T(v')\|_2 \leq \|v - v'\|_2$, for every $v, v' \in \mathbb{R}^d$.

The required proximity operator of φ is directly given, for every $(s,t) \in \mathbb{R}^n \times \mathbb{R}^k$, by

$$\text{prox}_{\varphi}(s,t) = \left(\text{prox}_{\varphi_1}(s), \text{prox}_{\varphi_2}(t)\right).$$

Both prox_{φ_1} and prox_{φ_2} can be easily computed. The latter requires computing the projection on the set S. The former requires, for each component of the vector $s \in \mathbb{R}^n$, the solution of a cubic equation as stated in the next lemma.

Lemma 4.2.1. *For every $\mu, \alpha \in \mathbb{R}$ and $r, \rho > 0$, the function $h : \mathbb{R}_+ \to \mathbb{R}$ defined at s as $h(s) := (s-\mu)^2 + r\left(\frac{\alpha^2}{s+\rho} + s\right)$ has a unique minimum on its domain, which is attained at $(x_0 - \rho)_+$, where x_0 is the largest real root of the polynomial $2x^3 + (r - 2(\mu+\rho))x^2 - r\alpha^2$.*

Proof. Setting the derivative of h equal to zero and making the change of variable $x = s + \rho$ yields the polynomial stated in the lemma. Let x_0 be the largest root of this polynomial. Since the function h is strictly convex on its domain and grows at infinity, its minimum can be attained only at one point, which is $x_0 - \rho$, if $x_0 > \rho$, and zero otherwise. ■

4.2.2 Accelerated Proximal Method

Theorem 4.2.1 motivates a proximal numerical approach to solving problem (4.2.1). Let $E(\beta) = \frac{1}{2}\|X\beta - y\|_2^2$ and assume that an upper bound L of $\|X^\top X\|$ is known[3]. Proximal first-order methods – see [12, 5, 36, 51] and references therein – can be used for nonsmooth optimisation, where the objective consists of a strongly smooth term, plus a nonsmooth part, in our case E and $\Gamma + \delta_\Lambda$, respectively. The idea is to replace E with its linear approximation around a point w_t specific to iteration t. This leads to the computation of a proximity operator, and specifically in our case to

$$u_t := (\beta_t, \lambda_t) \leftarrow \text{argmin}\left\{\frac{L}{2}\left\|(\beta,\lambda) - \left(w_t - \frac{1}{L}\nabla E(w_t)\right)\right\|_2^2 + \rho\Gamma(\beta,\lambda) : \beta \in \mathbb{R}^n, \lambda \in \Lambda\right\}.$$

Subsequently, the point w_t is updated, based on the current and previous estimates of the solution $u_t, u_{t-1}, \ldots$ and the process repeats.

The simplest update rule, which is also a commonly used one, is $w_t = u_t$. By contrast, *accelerated proximal methods* proposed by [36] use a carefully chosen w update with two levels of memory, u_t, u_{t-1}. If the proximity map can be exactly computed, such schemes exhibit a fast quadratic decay in terms of the iteration count, that is, the distance of the objective from the minimal value is $O\left(\frac{1}{T^2}\right)$ after T iterations. In the case that the proximity operator is computed *numerically*, it has been shown only very recently [53, 46] that, under some circumstances, the accelerated method still converges with the rate $O\left(\frac{1}{T^2}\right)$. The main advantages of accelerated

[3] For variants of such algorithms which adaptively learn L, see the following references.

methods are their low cost per iteration and their scalability to large problem sizes. Moreover, in applications where a thresholding operation is involved – as in Lemma 4.2.1 – the zeros in the solution are exact.

Algorithm 4.3 NEsterov PIcard-Opial algorithm (NEPIO)

Input: $u_1, w_1 \leftarrow$ arbitrary feasible values

`for` $t \leftarrow 1, 2, \ldots$

Compute a fixed point $\hat{v}^{(t)}$ of H_t by Picard-Opial

$u_{t+1} \leftarrow w_t - \frac{1}{L}\nabla E(w_t) - \frac{c}{L}B^\top \hat{v}^{(t)}$

$w_{t+1} \leftarrow \pi_{t+1}u_{t+1} - (\pi_{t+1} - 1)u_t$

`end`

Output: w

For our purposes, we use a version of accelerated methods influenced by [51]. Our final algorithm is called *NEPIO* and is summarised in Algorithm 4.3. According to Nesterov [36], the optimal update is

$$w_{t+1} \leftarrow u_{t+1} + \theta_{t+1}\left(\frac{1}{\theta_t} - 1\right)(u_{t+1} - u_t),$$

where the sequence θ_t is defined by $\theta_1 = 1$ and the recursion

$$\frac{1 - \theta_{t+1}}{\theta_{t+1}^2} = \frac{1}{\theta_t^2}\,. \tag{4.2.6}$$

We have adapted [51, Algorithm 2] (equivalent to FISTA [5]) by computing the proximity operator of $\frac{\varphi}{L} \circ B$ using the Picard-Opial process described in Section 4.2.1. We rephrased the algorithm using the sequence $\pi_t := 1 - \theta_t + \sqrt{1 - \theta_t} = 1 - \theta_t + \frac{\theta_t}{\theta_{t-1}}$ for numerical stability. At each iteration, the map H_t is defined by

$$H_t(v) := \left(I - \operatorname{prox}_{\frac{\phi}{c}}\right)\left(\left(I - \frac{c}{L}BB^\top\right)v - \frac{1}{L}B\Big(\nabla E(w_t) - Lw_t\Big)\right).$$

We also apply an Opial averaging so that the update at stage s of the proximity computation is $v_{s+1} = \kappa v_s + (1 - \kappa)H_t(v_s)$. By Theorem 4.2.1, the fixed point process combined with the assignment of u are equivalent to $u_{t+1} \leftarrow \operatorname{prox}_{\frac{\varphi}{L}\circ B}\left(w_t - \frac{1}{L}\nabla E(w_t)\right)$.

The reason for resorting to Picard-Opial is that exact computation of the proximity operator (4.2.4) is possible only in simple special cases for the set Λ. By contrast, our approach can be applied to a wide variety of constraints. Moreover, we are not aware of another proximal method for solving problems (4.2.1) or (3.1.1) and alternatives like interior point methods do not scale well with problem size. In Chapter 5, we will demonstrate empirically the scalability

of Algorithm 4.3, as well as the efficiency of both the proximity map computation and the overall method.

As noted in Section 3.4, we can compute exactly and efficiently the proximity operator in the case of the Wedge and the Tree penalties by performing the threshold $\hat{\lambda} = (\lambda(y) - \rho)_+$, where $\lambda(y)$ is computed using Algorithms 4.1 or 4.2. As can be seen in Chapter 5, Figure 5.15, the running time scales better in the number of dimensions.

Chapter 5

Numerical experiments

The goal of this chapter is threefold. The first one is to understand the usefulness of the many examples of penalties showed in Section 3.3. To this end, we designed several different sparse and structured models, so that the advantage of some of the penalties over other convex techniques (e.g. many Group Lasso variants) become apparent.

The second one is to test the algorithms to solve the problem proposed in Chapter 4. This, in particular for algorithm NEPIO (Section 4.2), is done by comparison with an all-purpose toolbox. We show that our algorithm scales well, and that it can be used for problems much larger than those handled by a generic toolbox, or by other existing techniques.

Finally, the last goal is compare the performances of our technique to greedy algorithms. Specifically, we focused on StructOMP ([20]) because, thanks to its flexibility, it can be used in situations where no other greedy algorithm can be usefully applied. In these situations, it can exploit prior knowledge comparable to that available to convex techniques.

The experiments in Section 5.1 appeared in [32], while other experiments have been included in other submitted work.

5.1 Experiments for different sets Λ

In this section we present some numerical simulations with the proposed method. For simplicity, we consider data generated noiselessly from $y = X\beta^*$, where $\beta^* \in \mathbb{R}^{100}$ is the true underlying regression vector, and X is an $m \times 100$ input matrix, m being the sample size. The elements of X are generated i.i.d. from the standard normal distribution, and the columns of X are then normalized such that their ℓ_2 norm is 1. Since we consider the noiseless case, we solve the interpolation problem $\min\{\Omega(\beta) : y = X\beta\}$, for different choices of the penalty function Ω. In practice, (3.1.1) is solved for a tiny value of the parameter, for example, $\rho = 10^{-8}$, which we found to be sufficient to ensure that the error term in (3.1.1) is negligible at the minimum. All experiments were repeated 50 times, generating each time a new matrix X. In the figures

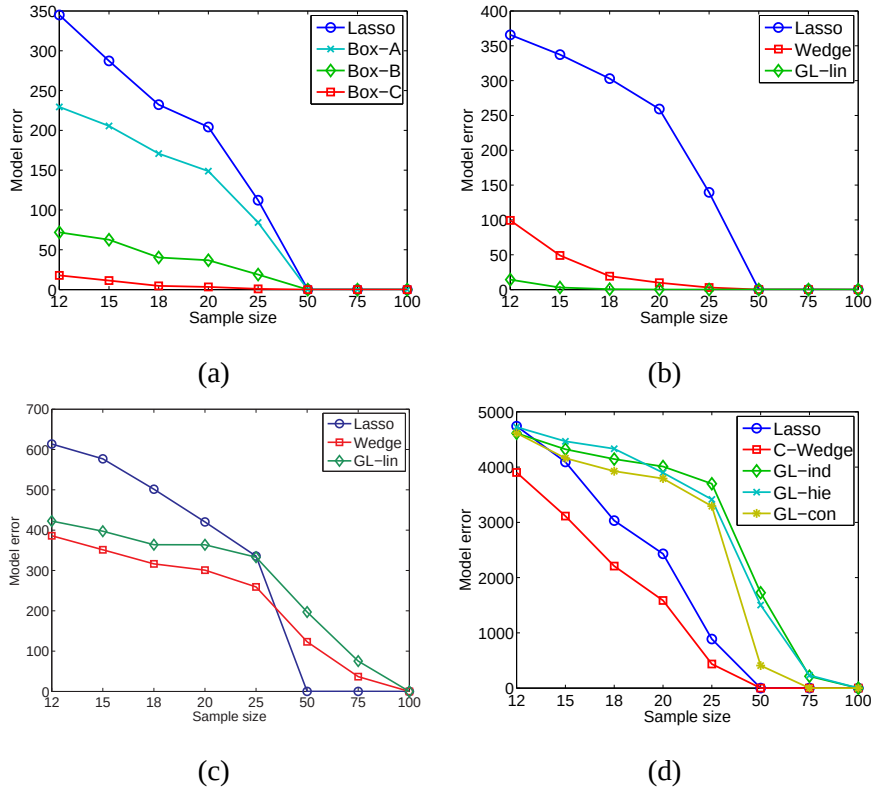


Figure 5.1: Comparison between different penalty methods: (a) Box vs. Lasso; (b,c) Wedge vs. Hierarchical group Lasso; (d) Composite wedge. See text for more information

we report the average of the model error of the vector $\hat{\beta}$ learned by each method, as a function of the sample size m. The former is defined as $\mathrm{ME}(\hat{\beta}) = \mathbb{E}[\|\hat{\beta} - \beta^*\|_2^2]$. In the following, we discuss a series of experiments, corresponding to different choices for the model vector β^* and its sparsity pattern. In all experiments, we solved the optimization problem (3.1.1) with the alternating algorithm presented in Section 4.1. Whenever possible we solved step (4.1.2) using analytical formulas and resorted to the solver CVX (*http://cvxr.com/cvx/*) in the other cases. For example, in the case of the wedge penalty, we found that the computational time of the algorithm in Figure 4.1 is $495, 603, 665, 869, 1175$ times faster than that of the solver CVX for $n = 100, 500, 1000, 2500, 5000$, respectively.

Box. In the first experiment the model is 10-sparse (it has 10 nonzero components), where each nonzero component, in a random position, is an integer uniformly sampled in the interval $[-10, 10]$. We wish to show that the more accurate the prior information about the model is, the more precise the estimate will be. We use a box penalty (see Theorem 3.3.1) constructed "around" the model, imagining that an oracle tells us that each component $|\beta_i^*|$ is bounded within an interval. We consider three boxes $B[a, b]$ of different sizes, namely $a_i = (r - |\beta_i^*|)_+$ and $b_i = (|\beta_i^*| - r)_+$ and radii $r = 5, 1$ and 0.1, which we denote as Box-A, Box-B and Box-C,

respectively. We compare these methods with the Lasso – see Figure 5.1-a. As expected, the three box penalties perform better. Moreover, as the radius of a box diminishes, the amount of information about the true model increases, and the performance improves.

Wedge. In the second experiment, we consider a regression vector, whose components are nonincreasing in absolute value and only a few are nonzero. Specifically, we choose a 10-sparse vector: $\beta_j^* = 11 - j$, if $j \in \mathbb{N}_{10}$ and zero otherwise. We compare the Lasso, which makes no use of such ordering information, with the wedge penalty $\Omega(\beta|W)$ (see Theorem 3.3.2) and the hierarchical group Lasso in [58], which both make use of such information. For the group Lasso we choose $\Omega(\beta) = \sum_{\ell \in \mathbb{N}_{100}} \|\beta_{|J_\ell}\|$, with $J_\ell = \{\ell, \ell+1, \ldots, 100\}$, $\ell \in \mathbb{N}_{100}$. These two methods are referred to as "Wedge" and "GL-lin" in Figure 5.1-b, respectively. As expected both methods improve over the Lasso, with "GL-lin" being the best of the two. We further tested the robustness of the methods, by adding two additional nonzero components with value of 10 to the vector β^* in a random position between 20 and 100. This result, reported in Figure 5.1-c, indicates that "GL-lin" is more sensitive to such a perturbation.

Composite wedge. Next we consider a more complex experiment, where the regression vector is sparse within different contiguous regions $P_1, \ldots, P_{10}$, and the ℓ_1 norm on one region is larger than the ℓ_1 norm on the next region. We choose sets $P_i = \{10(i-1)+1, \ldots, 10i\}$, $i \in \mathbb{N}_{10}$ and generate a 6-sparse vector β^* whose i-th nonzero element has value $31-i$ (decreasing) and is in a random position in P_i, for $i \in \mathbb{N}_6$. We encode this prior knowledge by choosing $\Omega(\beta|\Lambda)$ with $\Lambda = \left\{\lambda \in \mathbb{R}^{100} : \|\lambda_{P_i}\|_1 \geq \|\lambda_{P_{i+1}}\|_1,\ i \in \mathbb{N}_9\right\}$. This method constraints the sum of the sets to be nonincreasing and may be interpreted as the composition of the wedge set with an average operation across the sets P_i, which may be computed using Proposition 3.2.3 . This method, which is referred to as "C-Wedge" in Figure 5.1-d, is compared to the Lasso and to three other versions of the group Lasso. The first is a standard group Lasso with the nonoverlapping groups $J_i = P_i$, $i \in \mathbb{N}_{10}$, thus encouraging the presence of sets of zero elements, which is useful because there are 4 such sets. The second is a variation of the hierarchical group Lasso discussed above with $J_i = \cup_{j=i}^{10} P_j$, $i \in \mathbb{N}_{10}$. A problem with these approaches is that the ℓ_2 norm is applied at the level of the individual sets P_i, which does not promote sparsity within these sets. To counter this effect we can enforce contiguous nonzero patterns within each of the P_i, as proposed by [24]. That is, we consider as the groups the sets formed by all sequences of $q \in \mathbb{N}_9$ consecutive elements at the beginning or at the end of each of the sets P_i, for a total of 180 groups. These three groupings will be referred to as "GL-ind", "GL-hie'', "GL-con" in Figure 5.1-d, respectively. This result indicates the advantage of "C-Wedge" over the other methods considered. In particular, the group Lasso methods fall behind our method and

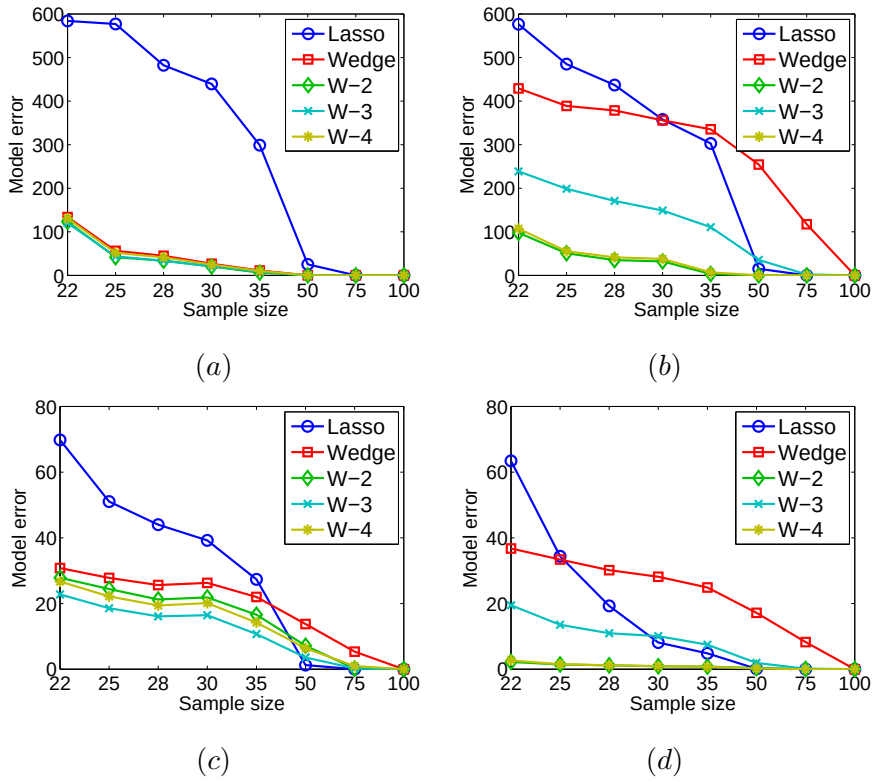


Figure 5.2: Penalty $\Omega(\beta|W^k)$, $k = 1, \ldots, 4$, used for several polynomial models: (a) degree 1, (b) degree 2, (c) degree 3; (d) degree 4.

the Lasso, with "GL-con" being slight better than "GL-ind" and "GL-hie". Notice also that all group Lasso methods gradually diminish the model error until they have a point for each dimension, while our method and the Lasso have a steeper descent, reaching zero at a number of points which is less than half the number of dimensions.

Polynomials. The constraints on the finite differences (see equation (3.3.11)) impose a structure on the sparsity of the model. To further investigate this possibility we now consider some models whose absolute value belong to the sets of constraints W^k, where $k = 1, \ldots, 4$. Specifically, we evaluate the polynomials $p_1(t) = -(t+5)$, $p_2(t) = (t+6)(t-2)$, $p_3(t) = -(t+6.5)t(t-1.5)$ and $p_4(t) = (t+6.5)(t-2.5)(t+1)t$ at 100 equally spaced (0.1) points starting from -7. We take the positive part of each component and scale it to 10, so that the results can be seen in Figure 5.3. The roots of the polynomials has been chosen so that the sparsity of the models is either 18 or 19.

We solve the interpolation problem using our method with the penalty $\Omega(\beta|W^k)$, $k = 1, \ldots, 4$, with the objective of testing the robustness of our method: the constraint set W^k should be a more meaningful choice when $|\beta^*|$ is in it, but the exact knowledge of the degree is not necessary. We see in Figures 5.2 that this is indeed the case: "W-k" outperform the Lasso

for every k, but among these methods the best one knows the degree of $|\beta^*|$.

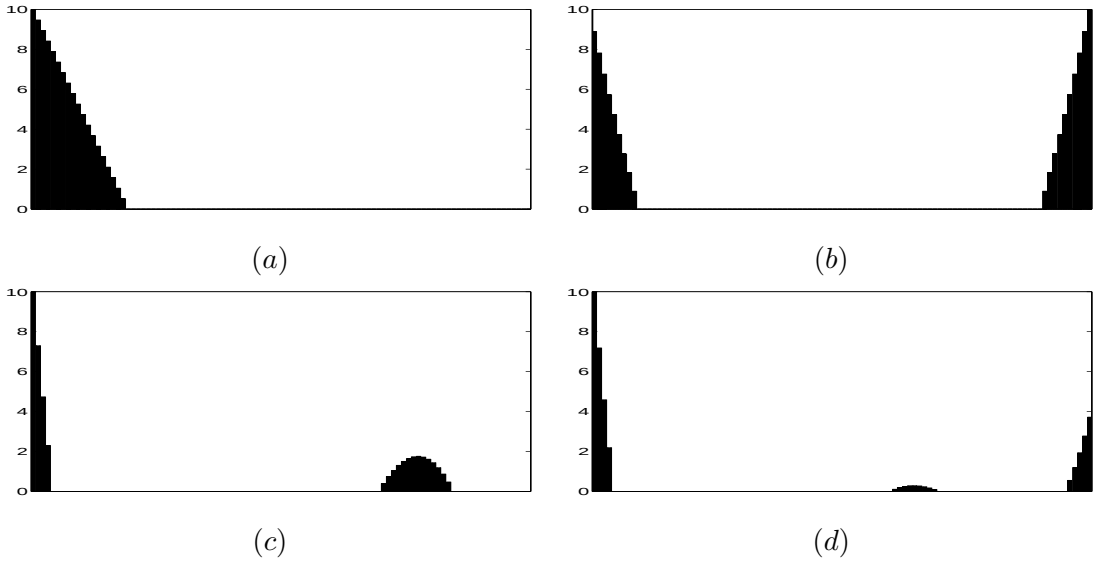


Figure 5.3: Silhouette of the polynomials by number of degree: (a) $k = 1$, (b) $k = 2$, (c) $k = 3$, (d) $k = 4$.

One important feature of these sparsity pattern is the number of contiguous regions: 1, 2, 2 and 3 respectively. As a way of testing the methods on a less artificial setting, we repeat the experiment using the same sparsity patterns, but replacing each nonzero component with a uniformly sampled random number between 1 and 2. In Figure 5.4 we can see that, even if now the models manifestly don't belong to W^k, we still have an advantage because the constraints look for a limited number of contiguous regions.

Finally, Figure 5.5 displays the regression vector found by the Lasso and the vector learned by "W-2" (left) and by the Lasso and "W-3" (right), in a single run with sample size of 20 and 35, respectively. The estimated vectors (green) are superposed to the true vector (black). Our method provides a better estimate than the Lasso in both cases.

5.2 Efficiency experiments for NEPIO

In this section, we present experiments with method (4.2.1). The goal of the experiments is to both study the computational and the statistical estimation properties of this method. One important aim of the experiments is to demonstrate that the method is statistically competitive or superior to state-of-the-art methods while being computationally efficient. The methods employed are the Lasso, StructOMP [20] and method (4.2.1) with the following choices for the constraint set Λ: *Grid-C*, $\Lambda_\alpha = \{\lambda : \|A\lambda\|_1 \leq \alpha\}$, where A is the edge map of a 1D or 2D grid and $\alpha > 0$, and *Tree-C*, $\Lambda = \{\lambda : A\lambda \geq 0\}$, where A is the edge map of a tree graph.

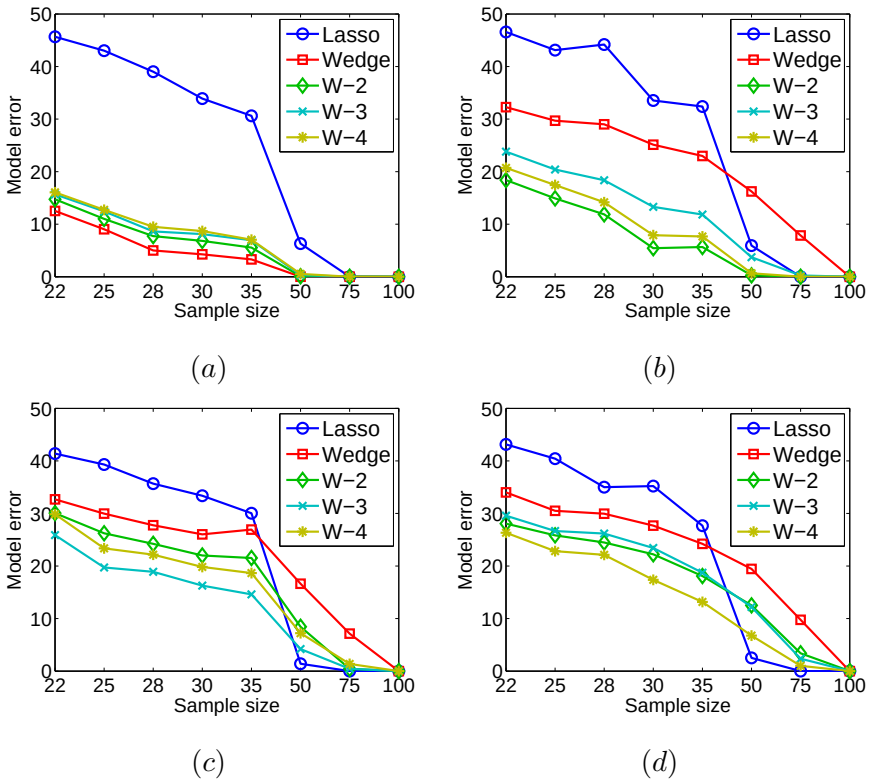


Figure 5.4: Penalty $\Omega(\beta|W^k)$, $k = 1, \ldots, 4$, used for several polynomial models with random values between the roots: (a) degree 1, (b) degree 2, (c) degree 3; (d) degree 4.

We solved the optimization problem (4.2.1) either with the toolbox CVX or with the proximal method presented in Section 4.2. When using the proximal method, we chose the parameter from Opial's Theorem $\kappa = 0.2$ and we stopped the iterations when the relative decrease in the objective value is less than 10^{-8}. For the computation of the proximity operator, we stopped the iterations of the Picard-Opial method when the relative difference between two consecutive iterates is smaller than 10^{-2}. We studied the effect of varying this tolerance in the next experiments. We used the square loss and computed the Lipschitz constant L using singular value decomposition. (Where not possible, a Frobenius estimate could be used.)

First, we investigated the computational properties of the proximal method. Our aim in these experiments was to show that our algorithm has a time complexity that scales linearly with the number of variables, while the sparsity and relative number of training examples is kept constant. We considered both the Grid and the Tree constraints and compared our algorithm to the toolbox CVX, which is an interior-point method solver. As is commonly known, interior-point methods are very fast for small problems, but do not scale well with the problem size. In the case of the Tree constraint, we also compared with a modified version of the alternating algorithm of [32]. For each problem size, we repeated the experiments 10 times

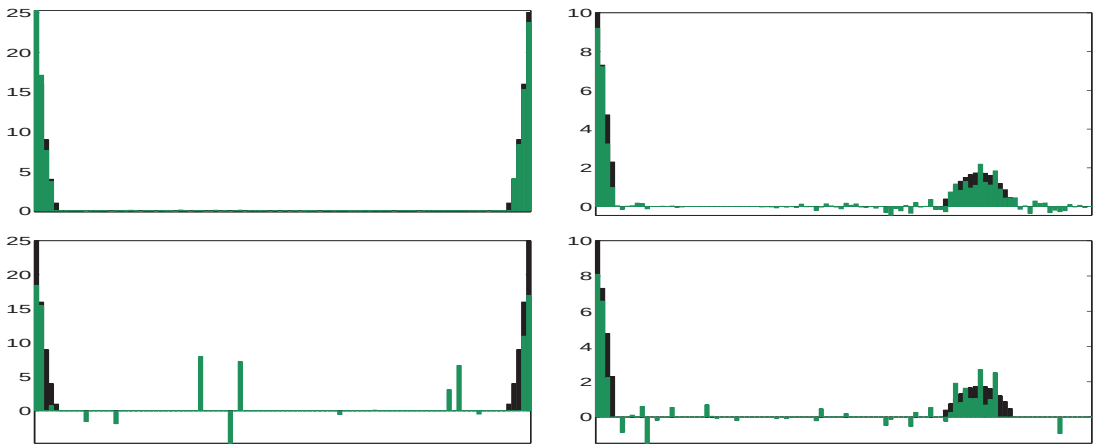


Figure 5.5: Lasso (top) vs. penalty $\Omega(\cdot|\Lambda)$ (bottom) for Convex (left) and Cubic (right); see text for more information.

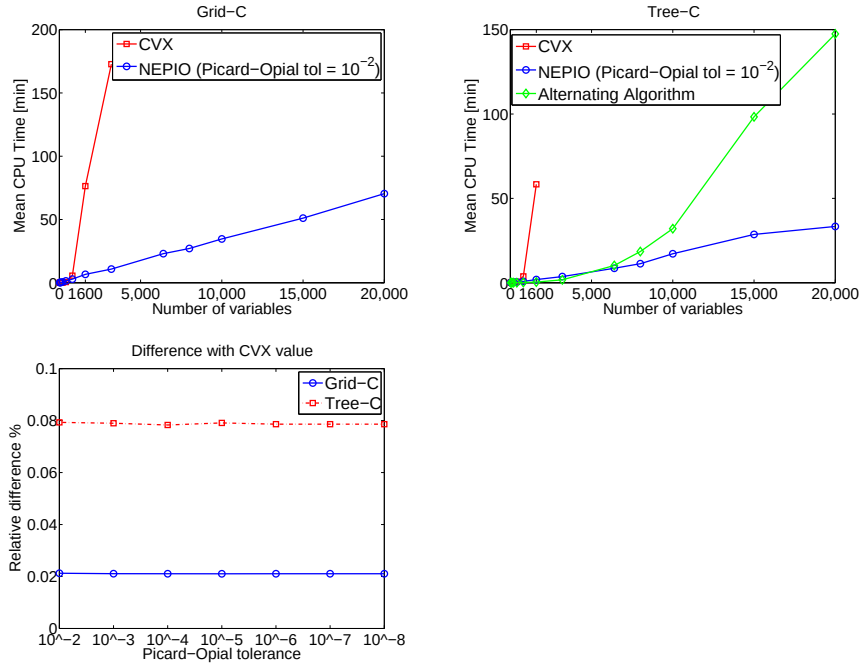


Figure 5.6: Computation time vs problem size for Grid-C (top-left) and Tree-C (top-right). Difference with the solution obtained via CVX vs Picard-Opial tolerance (bottom).

and we report the average computation time in Figure 5.6-*Top-left* and Figure 5.6-*Top-right* for Grid-C and Tree-C, respectively. This result indicates that our method is suitable for large scale experiments.

We also studied the importance of the Picard-Opial tolerance for converging to a good solution. In Figure 5.6-*Bottom*, we report the average relative distance to the solution obtained via CVX for different values of the Picard-Opial tolerance. We considered a problem with 100 variables and repeated the experiment 10 times with different sampling of training examples,

considering both the Grid and the Tree constraint. It is clear that decreasing the tolerance did not bring any advantage in terms of converging to a better solution, while it remarkably increased the computational overhead, passing from an average of $5s$ for a tolerance of 10^{-2} to $40s$ for 10^{-8} in the case of the Grid constraint.

Finally, we considered the 2D Grid-C case and observed that the number of Picard-Opial iterations needed to reach a tolerance of 10^{-2}, scales well with the number of variables n. For example when n varies between 200 and 6400, the average number of iterations varied between 20 and 40.

5.3 Tree-C and Grid-C

This section shares the same experimental protocol of Section 5.2.

One dimensional contiguous regions. In this experiment, we chose a model vector $\beta^* \in \mathbb{R}^{200}$ with 20 nonzero elements, whose values are random ± 1. We considered sparsity patterns forming one, two, three or four non-overlapping contiguous regions, which have lengths of 20, 10, 7 or 5, respectively. We generated a noiseless output from a matrix X whose elements have a standard Gaussian distribution. The estimates $\hat{\beta}$ for several models are then compared with the original. Figure 5.7 shows the model error $\frac{\| \hat{\beta} - \beta^* \|_2}{\| \beta^* \|_2}$ as the sample size changes from 22 (barely above the sparsity) up to 100 (half the dimensionality). This is the average over 50 runs, each with a different β^* and X. We observe that Grid-C outperforms both Lasso and StructOMP, whose performance deteriorates as the number of regions is increased. For one particular run with a sample size which is twice the model sparsity, Figure 5.8 shows the original vector and the estimates for different methods.

Two dimensional contiguous regions. We repeated the experiment in the case that the sparsity pattern of $\beta^* \in \mathbb{R}^{20 \times 20}$ consists of $2D$ rectangular regions. We considered either a single 5×5 region, two regions (4×4 and 3×3), three 3×3 regions and four 3×2 regions. Figure 5.9 shows the model error versus the sample size in this case. Figure 5.10 shows the original image and the images estimated by different methods for a sample size which is twice the model sparsity. Note that Grid-C is superior to both the Lasso and StructOMP and that StructOMP is outperformed by Lasso when the number of regions is more than two. This behavior is consistently confirmed by experiments in higher dimensions, not shown here for brevity.

Background subtraction. We replicated the experiment from [20, Sec. 7.3] with our method. Briefly, the underlying model β^* corresponds to the pixels of the foreground of a CCTV image, that is the portion of the image representing two standing persons. We measured the output as a random projection plus Gaussian noise. Figure 5.11-*Left* shows that, while the Grid-C

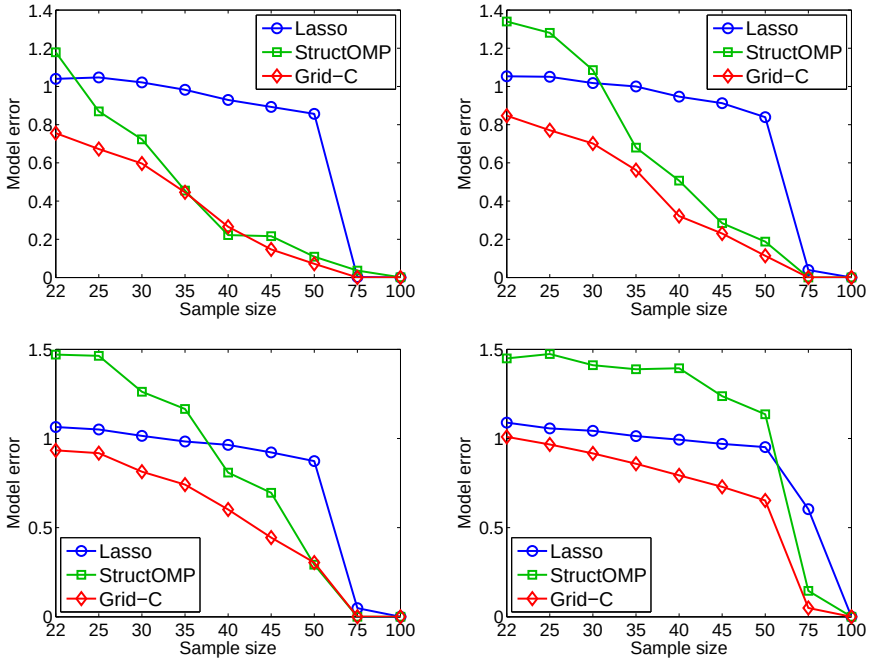


Figure 5.7: $1D$ contiguous regions: comparison between different methods for one (top-left), two (top-right), three (bottom-left) and four (bottom-right) regions.

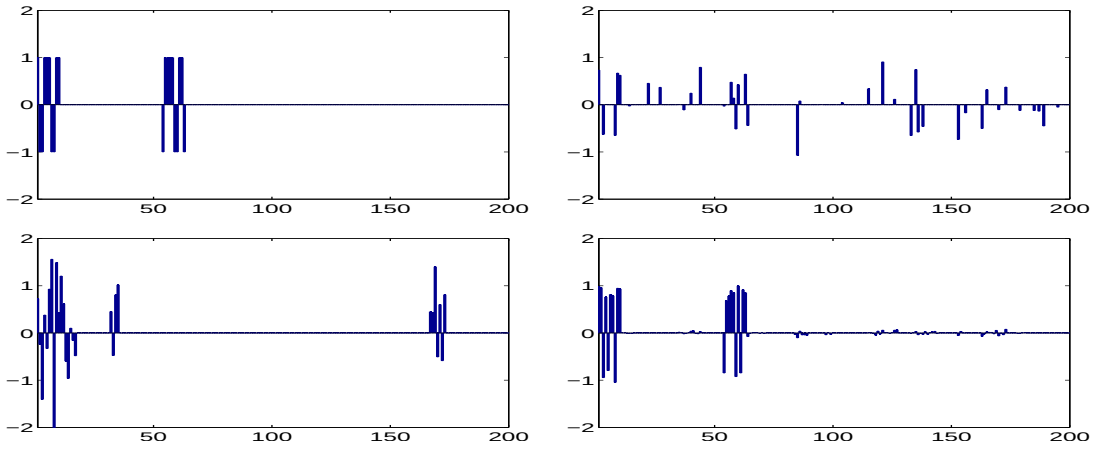


Figure 5.8: Two $1D$ contiguous regions: regression vector estimated by different models: β^* (top-left), Lasso (top-right), StructOMP (bottom-left), Grid-C (bottom-right).

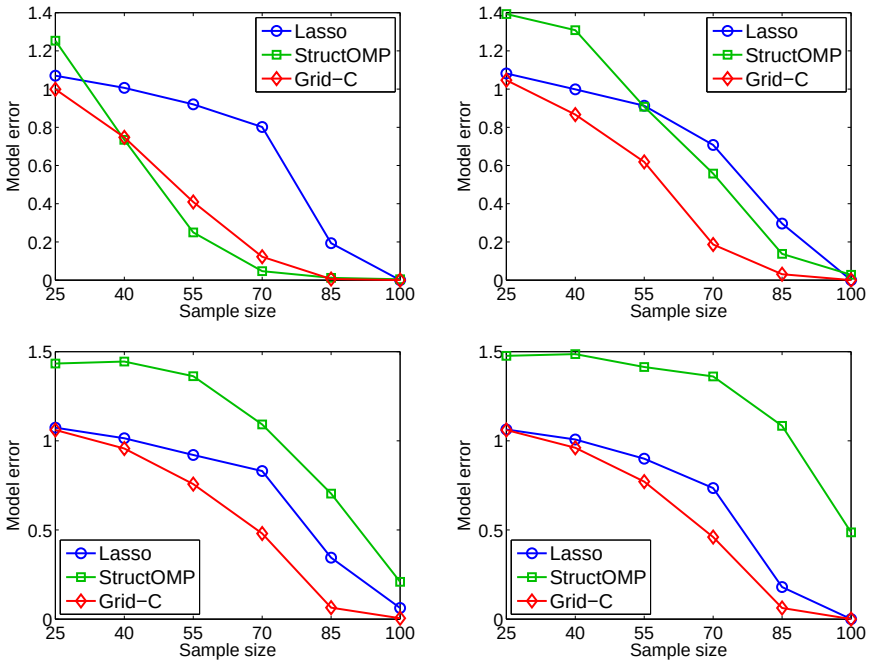


Figure 5.9: $2D$ contiguous regions: comparison between different methods for one (top-left), two (top-right), three (bottom-left) and four (bottom-right) regions.

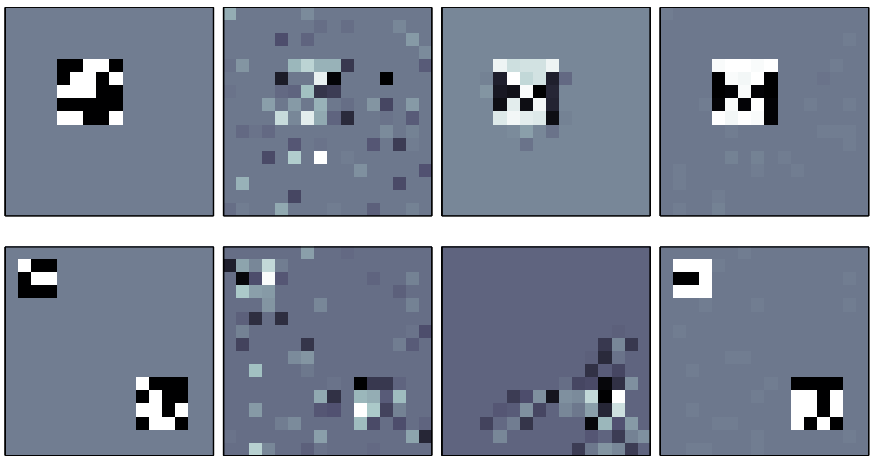


Figure 5.10: $2D$-contiguous regions: model vector (left) and vectors estimated by the Lasso, StructOMP and Grid-C (left to right), for one region (top group) and two regions (bottom group).

outperforms the Lasso, it is not as good as StructOMP. We speculate that this result is due to the non uniformity of the values of the image, which makes it harder for Grid-C to estimate the model.

Image Compressive Sensing. In this experiment, we compared the performance of Tree-C on an instance of 2D image compressive sensing, following the experimental protocol of [20]. Natural images can be well represented with a wavelet basis and their wavelet coefficients, besides being sparse, are also structured as a hierarchical tree. We computed the Haar-wavelet coefficients of a widely used *cameraman* image. We measured the output as a random projection plus Gaussian noise. StructOMP and Tree-C, both exploiting the tree structure, were used to recover the wavelet coefficients from the measurements and compared to the Lasso. The inverse wavelet transform was used to reconstruct the images with the estimated coefficients. The recovery performances of the methods are reported in Figure 5.11-*Right*, which highlights the good performance of Tree-C.

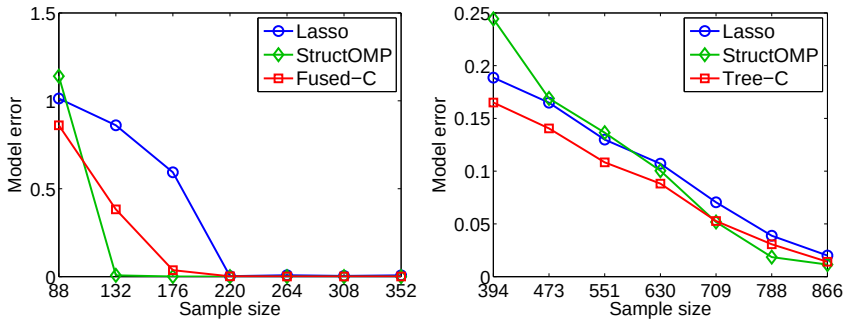


Figure 5.11: Model error for the background subtraction (left) and *cameraman* (right) experiments.

5.4 Tree-C and exact proxy

We explore the statistical properties of the tree penalty Tree-C by means of two experiments. In the first synthetic experiment, we embed the true vector $\beta^* \in \mathbb{R}^{85}$ into a tree structure where each node has exactly four children. Trees with the same branching factor can be used to order the wavelet coefficients of real images, as we will do in the second experiment. We want to show that our method, which is called Tree-C in the plots labels, is well suited to recover underlying vectors with a hierarchy of components given by the tree.

We compare Tree-C with other methods that should perform well in this case. The first one is StructOMP [20], a greedy method based on information theory. When applied to trees, this method prefers models whose components are connected through the tree graph. The second method is the hierarchical Group Lasso [58], GL-Hie. Given a careful choice of overlapping

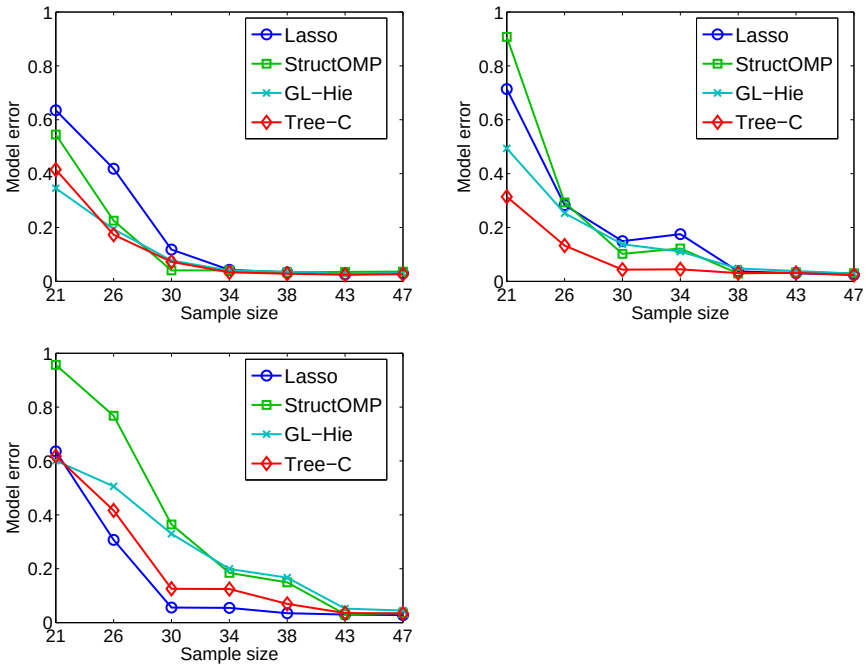


Figure 5.12: Model error for the synthetic tree experiment for the three sparsity patterns described in the text.

groups, this method favours models that respect the hierarchy. As a benchmark, we also include the solution of the Lasso. For each model, except StructOMP, we tried the values $\rho = 10^i$, $i = 3, 2, \ldots, -10, -11$ for the regularisation parameter, and selected the one which achieves the minimum model error. For StructOMP we used the complexity parameter of the model.

The sparsity of the model is 10% of the number of variables, and nonzeros elements have value 1. We consider three different sparsity patterns: in the first one, all nonzero elements are clustered at the root of the tree; in the second, half of the nonzero components are connected to the root, and half are at a middle depth; finally, in the third patterns all nonzero components are at a middle depth. As a measure of statistical performance we use the model error, which is defined as $\|\hat{\beta} - \beta^*\|_2 / \|\beta^*\|_2$ for each estimated vector $\hat{\beta}$. This quantity, averaged over 10 replicates, is shown in Figure 5.12.

As expected, the performance of the Lasso is not affected by the different sparsity patterns. For the first pattern (Figure 5.12-*Top-left*), which is entirely consistent with the tree structure, the method GL-Hie has a strong advantage. However, the results show that its performance is consistent with StructOMP and Tree-C. The results for the second pattern, (Figure 5.12-*Top-right*), which is an intermediate situation, show that Tree-C is more robust than the other methods. For the third pattern, which is completely inconsistent with the tree structure, we see that all methods are negatively affected.

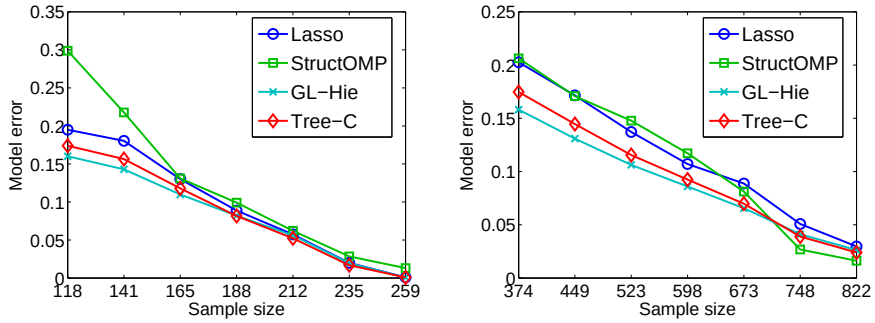


Figure 5.13: Model error for the wavelet tree experiment, 16×16 (left) and 32×32 (right).

For the second experiment, we consider an instance of a 2D image compressive sensing problem. Natural images can be well represented with a wavelet basis and their wavelet coefficients, besides being sparse, are also structured as a hierarchical tree, like the synthetic one we used in the first experiment. We follow the experimental protocol of [20] to compare the performance of Tree-C against the other methods. We computed the Haar-wavelet coefficients of the widely used *cameraman* image, scaled to 16×16 and 32×32 pixels. Despite being organized in a tree structure, only 42% and 47%, respectively, of the wavelet coefficients respect the hierarchy. We measured the output as a random projection plus Gaussian noise with zero mean and $\sigma = 0.01$. The inverse wavelet transform was used to reconstruct the images from the estimated coefficients. The recovery performances of the methods against the sample size are reported in Figure 5.13. For model selection, we restricted the values of ρ to 10^{-i}, $i = 1, 3, 5, 7$, as this proved to be enough.

We observe that for this problem all methods perform very similarly, with Tree-C and GL-Hie being slightly better. This result indicates that, even when the tree hierarchy is not strictly respected by the true regression vector, estimation with the proposed Tree-C penalty can still be used effectively.

We performed a simulation to empirically analyze the efficiency of algorithms 4.2 and 4.1. In Figure 5.14, we present the average time needed for the algorithm to compute the partition for random vector embedded in a tree of up to 25600 variables. The trees where generated with four children for each node. From the partition, it is possible to compute the proximity operator as per Equation (3.4.6).

The same experiment has been repeated for the line graph, again up to 25600 variables, and the results are shown in Figure 5.14-*right*. In this case the amount of time increases linearly in the number of dimension.

The exact computation of the proximity operator is used in the statistical experiments as the

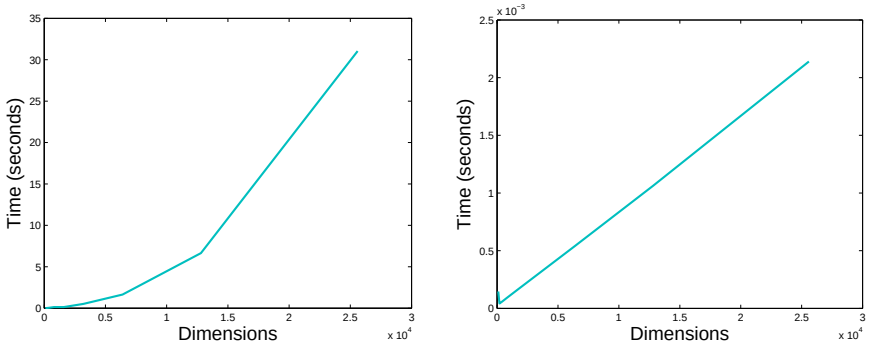


Figure 5.14: Average running time against dimensions for Algorithm 4.2 and Algorithm 4.1.

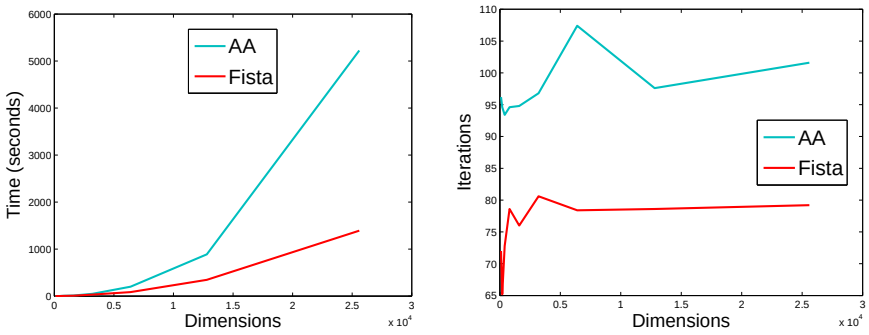


Figure 5.15: Average running time (top) and number of iterations (bottom) against dimensions for alternating algorithm (AA) and Fista.

inner step of the Fista-like algorithm, see [5]. The partition of the tree gives the minimization of the original problem, i.e. Problem (4.2.1), with respect to the variables λ. It can be used in the alternating algorithm described in 4.1. As an empirical comparison between the two algorithms, Figure 5.15-*top* shows their average running time and Figure 5.15-*bottom* their average number of iterations.

Chapter 6

Conclusions

We proposed a family of penalty functions that can be used to model structured sparsity in linear regression. We provided theoretical, algorithmic and computational information about this new class of penalty functions. Our theoretical observations highlight the generality of this framework to model structured sparsity. An important feature of our approach is that it can deal with richer model structures than current approaches while maintaining convexity of the penalty function. Our practical experience indicates that these penalties perform well numerically, improving over state of the art penalty methods for structure sparsity, suggesting that our framework is promising for applications.

The methods developed here can be extended in different directions. We mention here several possibilities. For example, for any $r > 0$, it readily follows that

$$\|\beta\|_p^p = \inf \left\{ \frac{r}{r+1} \sum_{i \in \mathbb{N}_n} \frac{\beta_i^2}{\lambda_i} + \frac{1}{r} \lambda_i^r : \lambda \in \mathbb{R}^n_{++} \right\} \tag{6.0.1}$$

where $p = 2r/(r+1)$ and $\|\beta\|_p$ is the usual ℓ^p-norm on $\mathbb{R}^n$. This formula leads us to consider the same optimization problem over a constraint set Λ. Note that if $p \to 0$ the left hand side of the above equation converges to the cardinality of the support of the vector β.

Problems associated with multi-task learning [1, 2] demand matrix analogs of the results discussed here. In this regard, we propose the following family of unitarily invariant norms on $d \times n$ matrices. Let $k = \min(d, n)$ and $\sigma(B) \in \mathbb{R}^k_+$ be the vector formed from the singular values of B. When Λ is a nonempty convex set which is invariant under permutations our point of view in this thesis suggests the penalty

$$\|B\|_\Lambda = \Omega(\sigma(B)|\Lambda).$$

The fact that this is a norm, follows from the von Neumann characterization of unitarily invariant norms. When $\Lambda = \mathbb{R}^k_{++}$ this norm reduces to the trace norm [2].

Finally, the ideas discussed in this thesis can be used in the context of kernel learning, see [3, 26, 27, 31, 43] and references therein. Let K_ℓ, $\ell \in \mathbb{N}_n$ be prescribed reproducing kernels

on a set $\mathcal{X}$, and H_ℓ the corresponding reproducing kernel Hilbert spaces with norms $\|\cdot\|_\ell$. We consider the problem

$$\min \left\{ \sum_{i \in \mathbb{N}_m} \left(y_i - \sum_{\ell \in \mathbb{N}_n} f_\ell(x_i) \right)^2 + \rho \Omega^2 \Big((\|f_\ell\|_\ell : \ell \in \mathbb{N}_n) | \Lambda \Big) : f_\ell \in H_\ell, \ell \in \mathbb{N}_n \right\}$$

and note that the choice $\Lambda = \mathbb{R}^n_{++}$ corresponds to multiple kernel learning.

All the above examples deserve a detailed analysis and we hope to provide such in future work.

We proposed new families of penalties and presented a new algorithm and results on the class of structured sparsity penalty functions proposed by [32]. These penalties can be used, among else, to learn regression vectors whose sparsity pattern is formed by few contiguous regions. We presented a proximal method for solving this class of penalty functions and derived an efficient fixed-point method for computing the proximity operator of our penalty. We reported encouraging experimental results, which highlight the advantages of the proposed penalty function over a state-of-the-art greedy method [20]. At the same time, our numerical simulations indicate that the proximal method is computationally efficient, scaling linearly with the problem size. An important problem which we wish to address in the future is to study the convergence rate of the method and determine whether the optimal rate $O(\frac{1}{T^2})$ can be attained. Finally, it would be important to derive sparse oracle inequalities for the estimators studied here.

Appendix A

Notations

β is the vector of coefficients of the model.

β^* is the underlying model, unknown and object to our research.

$\hat{\beta}$ is the estimate.

n is the dimensionality of the data. That is, $\beta \in \mathbb{R}^n$.

y is the observed vector.

m is the sample size, number of points in the training set.

X is the $m \times d$ matrix.

x a feature, column of the matrix.

L is the loss function.

P penalty function.

γ is the regularization parameter

J is a group of variables.

$\mathcal{J}$ is a set of groups.

A^T is the transpose of A.

$A^\dagger$ is the Moore-Penrose pseudoinverse of A.

δ_C it the indicator function of the set C, that is $\delta_C(x) = 0$ if $x \in C$, $\delta_C(x) = +\infty$ otherwise.

supp (β) is the support of vector β, that is the set $\{i \in \mathbb{N}_n : \beta_i \neq 0\}$.

$\mathbb{R}^n_{++}$ is the positive orthant, that is the set $\{x \in \mathbb{R}^n : x_i > 0, i \in \mathbb{N}_n\}$.

Appendix B

Proofs

In this appendix we describe in detail a result due to J.M. Danskin, which we use in the proof of Proposition 3.2.1.

Definition B.0.1. *Let f be a real-valued function defined on an open subset X of $\mathbb{R}^n$ and $u \in \mathbb{R}^n$. The directional derivative of f at $x \in X$ in the "direction" u is denoted by $(D_u f)(x)$ and is defined as*

$$(D_u f)(x) := \lim_{t \to 0} \frac{f(x+tu) - f(x)}{t}$$

if the limit exists. When the limit is taken through nonnegative values of t, we denote the corresponding right directional derivative by D_u^+.

Let Y be a compact metric space, $F : X \times Y \to \mathbb{R}$ a continuous function on its domain and define the function $f : X \to \mathbb{R}$ at $x \in X$ as

$$f(x) = \min \{F(x, y) : y \in Y\}.$$

We say that F is Danskin function if, for every $u \in \mathbb{R}^n$, the function $F'_u : X \times Y \to \mathbb{R}$ defined at $(x, y) \in X \times Y$ as $F'_u(x, y) = (D_u F(\cdot, y))(x)$ is continuous on $X \times Y$. Our notation is meant to convey the fact that the directional derivative is taken relative to the first variable of F.

Theorem B.0.1. *If X is an open subset of $\mathbb{R}^n$, Y a is compact metric space, $F : X \times Y$ is a Danskin function, $u \in \mathbb{R}^n$ and $x \in X$, then*

$$(D_u^+ f)(x) = \min \left\{F'_u(x, y) : y \in Y_x\right\}$$

where $Y_x := \{y : y \in Y,\ F(x, y) = f(x)\}$.

Proof. If $x \in X$, $y \in Y_x$ and $u \in \mathbb{R}^n$ then, for all positive t, sufficiently small, we have that

$$\frac{f(x+tu) - f(x)}{t} \leq \frac{F(x+tu, y) - F(x, y)}{t}.$$

Letting $t \to 0^+$, we get that

$$\limsup_{t\to 0^+} \frac{f(x+tu)-f(x)}{t} \leq \min\left\{F'_u(x,y) : y \in Y_x\right\}. \tag{B.0.1}$$

Next, we choose a sequence $\{t_k : k \in \mathbb{N}\}$ of positive numbers such that $\lim_{k\to\infty} t_k = 0$ and

$$\lim_{k\to\infty} \frac{f(x+t_k u)-f(x)}{t_k} = \liminf_{t\to 0^+} \frac{f(x+tu)-f(x)}{t}.$$

From the definition of the function f, there exists a $y_k \in Y$ such that $f(x+t_k u) = F(x+t_k u, y_k)$. Since Y is a compact metric space, there is a subsequence $\{y_{k_\ell} : \ell \in \mathbb{N}\}$ which converges to some $y_\infty \in Y$. It readily follows from our hypothesis that the function f is continuous on X. Indeed, we have, for every $x_1, x_2 \in X$, that

$$|f(x_1)-f(x_2)| \leq \max\{|F(x_1,y)-F(x_2,y)| : y \in Y\}.$$

Hence we conclude that $y_\infty \in Y_x$. Moreover, we have that

$$\frac{f(x+t_k u)-f(x)}{t_k} \geq \frac{F(x+t_k u, y_k)-F(x,y_k)}{t_k}.$$

By the mean value theorem, we conclude that there is positive number $\sigma_k < t_k$ such that the

$$\frac{f(x+t_k u)-f(x)}{t_k} \geq F'_u(x+\sigma_k u, y_k).$$

We let $\ell \to \infty$ and use the hypothesis that F is a Danskin function to conclude that

$$\liminf_{t\to 0^+} \frac{f(x+tu)-f(x)}{t} \geq F'_u(x,y_\infty) \geq \min\left\{F'_u(x,y) : y \in Y_x\right\}.$$

Combining this inequality with (B.0.1) proves the result. ■

We note that [6, p. 737] describes a result which is attributed to Danskin without reference. This result differs from the result presented above. The result in [6, p. 737] requires the hypothesis of convexity on the function F. The theorem above and its proof is an adaptation of Theorem 1 in [13].

We are now ready to present the proof of Proposition 3.2.1.

Proof of Proposition 3.2.1 The essential part of the proof is an application of Theorem B.0.1. To apply this result, we start with a $\beta \in (\mathbb{R}\backslash\{0\})^n$ and introduce a neighborhood of this vector defined as

$$X(\beta) = \left\{\alpha : \alpha \in \Lambda, \|\alpha-\beta\|_\infty < \frac{\beta_{\min}}{2}\right\},$$

where $\beta_{\min} = \min\{|\beta_i| : i \in \mathbb{N}_n\}$. Theorem B.0.1 also requires us to specify a compact subset $Y(\beta)$ of $\mathbb{R}^n$. We construct this set in the following way. We choose a fixed $\overline{\lambda} \in \Lambda$ and a positive

$\epsilon > 0$. From these constants we define the constants

$$\begin{aligned} c(\beta) &= \sum_{i\in\mathbb{N}_n}\left(\frac{(|\beta_i|+\beta_{\min}/2)^2}{\overline{\lambda}_i}+\overline{\lambda}_i\right), \\ a(\beta) &= \frac{\beta_{\min}^2}{4(c(\beta)+\epsilon)}, \\ b(\beta) &= \max(a(\beta), c(\beta)+\epsilon). \end{aligned}$$

With these definitions, we choose our compact set $Y(\beta)$ to be $Y(\beta) = \Lambda_{a(\beta),b(\beta)}$. To apply Theorem B.0.1, we use the fact, for any $\alpha \in X(\beta)$, that

$$\Omega(\alpha|\Lambda) = \min\{\Gamma(\alpha,\lambda) : \lambda \in Y(\beta)\}. \tag{B.0.2}$$

Let us, for the moment, assume the validity of this equation and proceed with the remainaing details of the proof. As a consequence of this equation, we conclude that there exists a vector $\lambda(\beta)$ such that $\Omega(\beta|\Lambda) = \Gamma(\beta,\lambda(\beta))$. Moreover, when $\beta \in (\mathbb{R}\backslash\{0\})^n$ the function $\Gamma_\beta : \mathbb{R}^n_{++} \to \mathbb{R}$, defined for $\lambda \in \mathbb{R}^n_{++}$, as $\Gamma_\beta(\lambda) = \Gamma(\beta,\lambda)$ is strictly convex on its domain and so, $\lambda(\beta)$ is unique.

By construction, we know, for every $\alpha \in X(\beta)$, that

$$\max\left\{\left|\lambda_i(\alpha) - \frac{a(\beta)+b(\beta)}{2}\right| : i \in \mathbb{N}_n\right\} \leq \frac{a(\beta)+b(\beta)}{2}.$$

From this inequality we shall establish that $\lambda(\beta)$ depends continuously on β. To this end, we choose any sequence $\{\beta^k : k \in \mathbb{N}\}$ which converges to β and from the above inequality we conclude that the sequence of vectors $\lambda(\beta^k)$ is bounded. However this sequence can only have one cluster point, namely $\lambda(\beta)$, because Γ is continuous. Specifically, if $\lim_{k\to\infty}\lambda(\beta^k) = \tilde{\lambda}$, then, for every $\lambda \in \Lambda$, it holds that $\Gamma(\beta^k,\lambda(\beta^k)) \leq \Gamma(\beta^k,\lambda)$ and, passing to the limit $\Gamma(\beta,\tilde{\lambda}) \leq \Gamma(\beta,\lambda)$, implying that $\tilde{\lambda} = \lambda(\beta)$.

Likewise, equation (B.0.2) yields the formula for the partial derivatives of $\Omega(\cdot|\Lambda)$. Specifically, we identify F and f in Theorem B.0.1 with Γ and $\Omega(\cdot|\Lambda)$, respectively, and note that

$$\frac{\partial\Omega}{\partial\beta_i}(\beta|\Lambda) = \min\left\{\frac{\partial\Gamma}{\partial\beta_i}(\beta,\lambda) : \lambda\in\Lambda,\ \Gamma(\beta,\lambda) = \Omega(\beta|\Lambda)\right\} = \frac{\partial\Gamma}{\partial\beta_i}(\beta,\lambda(\beta)) = 2\frac{\beta_i}{\lambda_i(\beta)}.$$

Therefore, the proof will be completed after we have established equation (B.0.2). To this end, we note that if $\lambda = (\lambda_i : i \in \mathbb{N}_n) \in \Lambda\backslash Y(\beta)$ then there exists $j \in \mathbb{N}_n$ such that either $\lambda_j < a(\beta)$ or $\lambda_j > b(\beta)$. Thus, we have, for every $\alpha \in X(\beta)$, that

$$\Gamma(\alpha,\lambda) \geq \frac{1}{2}\left(\frac{\alpha_j^2}{\lambda_j}+\lambda_j\right) \geq \frac{1}{2}\min\left(\frac{\beta_{\min}^2}{4a(\beta)}, b(\beta)\right) = \frac{c(\beta)+\epsilon}{2} \geq \Omega(\alpha|\Lambda)+\frac{\epsilon}{2}.$$

This inequality yields equation (B.0.2). ■

We end this appendix by extracting the essential features of the convergence of the alternating algorithm as described in Section 4.1. We start with two compact sets, $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$, and a strictly convex function $F : X \times Y \to \mathbb{R}$. Corresponding to F we introduce two additional functions, $f : X \to \mathbb{R}$ and $g : Y \to \mathbb{R}$ defined, for every $x \in X, y \in Y$ as

$$f(x) = \min\{F(x, y') : y' \in Y\}, \quad g(y) = \min\{F(x', y) : x' \in X\}.$$

Moreover, we introduce the mappings $\phi_1 : Y \to X$ and $\phi_2 : X \to Y$, defined, for every $x \in X$, $y \in Y$, as

$$\phi_1(y) = \operatorname{argmin}\{F(x, y) : x \in X\}, \quad \phi_2(x) = \operatorname{argmin}\{F(x, y) : y \in Y\}.$$

Lemma B.0.1. *The mappings ϕ_1 and ϕ_2 are continuous on their respective domain.*

Proof. We prove that ϕ_1 is continuous. The same argument applies to ϕ_2. Suppose that $\{y^k : k \in \mathbb{N}\}$ is a sequence in Y which converges to some point $y \in Y$. Then, since F is jointly strictly convex, the sequence $\{\phi_1(y^k) : k \in \mathbb{N}\}$ has only one cluster point in X, namely $\phi_1(y)$. Indeed, if there is a subsequence $\{\phi_1(y^{k_\ell}); \ell \in \mathbb{N}\}$ which converges to $\tilde{x}$, then by definition, we have, for every $x \in X$, $\ell \in \mathbb{N}$, that $F(\phi_1(y^{k_\ell}), y^{k_\ell}) \leq F(x, y^{k_\ell})$. From this inequality it follows that $F(\tilde{x}, y) \leq F(x, y)$. Consequently, we conclude that $\tilde{x} = \phi_1(y)$. Finally, since X is compact, we conclude that the $\lim_{k\to\infty} \phi_1(y^k) = \phi_1(y)$. ■

As an immediate consequence of the lemma, we see that f and g are continuous on their respective domains, because, for every $x \in X, y \in Y$, we have that $f(x) = F(x, \phi_2(x))$ and $g(y) = F(\phi_1(y), y)$.

We are now ready to define the alternating algorithm.

Definition B.0.2. *Choose any $y_0 \in \operatorname{int}(Y)$ and, for every $k \in \mathbb{N}$, define the iterates*

$$x^k = \phi_1(y^{k-1})$$

and

$$y^k = \phi_2(x^k).$$

Theorem B.0.2. *If $F : X \times Y \to \mathbb{R}$ satisfies the above hypotheses and it is differentiable on the interior of its domain, and there are compact subsets $X_0 \subset \operatorname{int}(X)$, $Y_0 \subseteq \operatorname{int}(Y)$ such that, for all $k \in \mathbb{N}$, $(x^k, y^k) \in X_0 \times Y_0$, then the sequence $\{(x^k, y^k) : k \in \mathbb{N}\}$ converges to the unique minimum of F on its domain.*

Proof. First, we define, for every $k \in \mathbb{N}$, the real numbers $\theta_k = F(x^k, y^{k-1})$ and $\nu_k = F(x^k, y^k)$. We observe, for all $k \geq 2$, that

$$\nu_k \leq \theta_k \leq \nu_{k-1}.$$

Therefore, there exists a constant ψ such that $\lim_{k\to\infty} \theta_k = \lim_{k\to\infty} \nu_k = \psi$. Suppose, there is a subsequence $\{x^{k_\ell} : \ell \in \mathbb{N}\}$ such that $\lim_{\ell\to\infty} x^{k_\ell} = x$. Then $\lim_{\ell\to\infty} \phi_2(x^{k_\ell}) = \phi_2(x) =: y$. Observe that $\nu_k = f(x^k)$ and $\theta_{k+1} = g(y^k)$. Hence we conclude that

$$f(x) = g(y) = \psi.$$

Since F is differentiable, (x, y) is a stationary point of F in $\mathrm{int}(X) \times \mathrm{int}(Y)$. Moreover, since F is strictly convex, it has a unique stationary point which occurs at its global minimum. ■

Appendix C

Algorithms

The algorithms summarised in this appendix are the ones described in Chapter 2: Ortogonal Matching Pursuit, model-based Cosamp as described by Baraniuk, StructOMP by Tong Zhang and Caspar by Wasserman.

Algorithm C.1 OMP, Orthogonal Matching Pursuit (adapted from [50])

Input: X, y, sparsity level s.

Initialisation: Active set $A = \emptyset$, initial residue $r^{(0)} = y$, iteration counter $t = 1$. $X^{(0)}$ is the empty matrix.

1. Find the index of the most correlated factor: $j^\star = \underset{j=1,\ldots,n}{\operatorname{argmax}} |\langle r^{(t-1)}, x_j \rangle|$.
2. Include the index into the active set: $A = A \cup j^\star$, and include the new factor in the matrix $X^{(t)} = [X^{(t-1)}, x_{j^\star}]$.
3. Solve the least squares problem: $\beta^{(t)} = \underset{\beta}{\operatorname{argmin}} \|X^{(t)}\beta - y\|_2^2$.
4. Calculate the new residual: $r^{(t)} = y - X^{(t)}\beta^{(t)}$.
5. Increment t, and go to step 1 if $t < s$.

Output: Estimate $\hat{\beta}_j = \beta_j^{(s)}$ for $j \in A$, 0 otherwise.

Algorithm C.2 CaSpaR, Clustered and Sparse Regression (from [42])

Input: X, y, distance function d, kernel function K, $\alpha \in (0,1)$, $\tau > 0$

Initialisation: Active set $A = \emptyset$.

1. Fit the linear model $\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \|X\beta - y\|_2^2 \right\}$, such that supp $(\beta) \subseteq A$.
2. Compute $W_j = \frac{1}{|A|} \sum_{\{i \in A\}} K_h(d(i,j))$, for all $j \notin A$. If this is the first iteration, then $W_j = 1$ for all j.
3. Set $j^\star = \underset{j \notin A}{\operatorname{argmax}} \{W_j |\langle X\beta - y, x_j \rangle|\}$.
4. If $|\langle X\beta - y, x_{j^\star} \rangle| < \tau$, then stop, else set $A = A \cup j^\star$ and go to Step 1.

Output: Estimate $\hat{\beta}$.

Algorithm C.3 StructOMP (from [20])

Input: X, y, $\mathcal{B} \subset 2^{\mathcal{J}}$, $s > 0$

Initialisation: Let $F^{(0)} = \emptyset$ and $\beta^{(0)} = 0$. Iteration counted $t = 1$.

1. Select $B^{(t)} \in \mathcal{B}$ to maximise the gain ratio

$$\frac{\|X^T_{B-F^{(t-1)}}(X\beta^{(t-1)} - y)\|_2^2}{\mathsf{c}(B \cup F^{(t-1)}) - \mathsf{c}(F^{(t-1)})}.$$

2. Let $F^{(t)} = B^{(t)} \cup F^{(t-1)}$.

3. Let $\beta^{(t)} = \underset{\beta}{\operatorname{argmin}}\{L(\beta) : \text{supp}\,(\beta) \subset F^{(t)}\}$.

4. If $c(\beta^{(t)}) > s$ stop, else increment t and go to Step 1.

Output: Estimate $\hat{\beta} = \beta^{(t)}$.

Algorithm C.4 Model-based CoSaMP (from [4])

Input: X, y, structured sparse approximation algorithm $\mathbb{M}$, $s > 0$

Initialisation: $\hat{\beta}_0 = 0$, $r = y$, $t = 0$.

1. Increment t.
2. Let $e = X^T r$ be the residual estimate.
3. Compute the support of the best s-sparse approximation: $\Omega = \text{supp}\left(\mathbb{M}(e)\right)$.
4. Merge the new support: $T = \Omega \cup \text{supp}\left(\hat{\beta}^{(t-1)}\right)$.
5. Form new signal estimate: $b|_T = X_T^{\dagger} y$, $b|_{T^C} = 0$.
6. Prune according to structure: $\hat{\beta}^{(t)} = \mathbb{M}(b)$.
7. Calculate the new residual: $r = y - X\hat{\beta}^{(t)}$.
8. If halting critarion is true, stop, else go to Step 1.

Output: Estimate $\hat{\beta}^{(t)}$.

Appendix D

Specialised Bregman iteration

In this appendix we present an alternative algorithm to compute the Grid-C penalty. We did not compare the efficiency of this algorithm but describe it here as a reference.

The Bregman iteration is a technique proposed by [15] to solve general optimisation problems where the penalty part contains a composition of the ℓ_1 norm. As this is the framework of the Grid-C penalty function in its Lagrangian form, we can adapt that algorithm to our case.

The implementation of the algorithm to compute our penalty requires the solution of a particular subproblem. We show how this can be found using the theory of [33], which relies on the computation of a composition of proximity map. This is approximated via a fixed point algorithm in a similar manner done for NEPIO (see Section 4.2).

In Section D.1 we revise the Bregman iteration technique and in Section D.2 we explain the implementation for the Grid-C penalty.

D.1 Generalities of Bregman iteration

The Split Bregman method, as proposed by [15], can be used to solve problems with a composition of ℓ_1 norms as regularisation part. The most general definition of the problem is

$$\min_u \{\|\Phi(u)\|_1 + H(u)\}, \tag{D.1.1}$$

where both Φ and H are convex functions.

To begin with, we consider the problem of finding the minimum of a single function $E(u)$, potentially non-differentiable, with a quadratic penalty term, that is

$$\min_u \left\{ E(u) + \frac{\lambda}{2}\|Au - b\|_2^2 \right\}. \tag{D.1.2}$$

The minimisation of $E(u)$ subject to the system of linear equations $Au = b$ can be obtained recursively using a series of increasing values for the parameter λ. This procedure, however, is not numerically stable.

In the case that A is a matrix (instead of a more general linear operator), then the solution to (D.1.2) can be found with the Bregman iteration:

$$u^{k+1} = \underset{u}{\operatorname{argmin}} \left\{ E(u) + \frac{\lambda}{2}\|Au - b^k\|_2^2 \right\}, \tag{D.1.3}$$

$$b^{k+1} = b^k + b - Au^k. \tag{D.1.4}$$

The vector b^k, at iteration k, represents the error of the linear system. This simple case from [39] and [55] hides a more complex iterations based on Bregman Distance and subgradients, which arises when A is not a matrix.

The splitting technique modifies the problem (D.1.1) introducing a new variable, allowing to cast the problem in a form similar to (D.1.2), so that the Bregman iteration can be used. Specifically, we will constraint the new variable d to take the values of $\Phi(u)$. The Lagrangian form of the new minimisation problem will be

$$\min_{u,d} \left\{ \|d\|_1 + H(u) + \frac{\lambda}{2}\|d - \Phi(u)\|_2^2 \right\}. \tag{D.1.5}$$

This problem has a form very similar to that of (D.1.2), and can be solved in a similar way:

$$(u^{k+1}, d^{k+1}) = \underset{u,d}{\operatorname{argmin}} \left\{ \|d\|_1 + H(u) + \frac{\lambda}{2}\|d - \Phi(u) - b^k\|_2^2 \right\}, \tag{D.1.6}$$

$$b^{k+1} = b^k + (\Phi(u^{k+1}) - d^{k+1}). \tag{D.1.7}$$

Furthermore, we note that this problem computes the ℓ_1 norm and the function H on different variables. One of the difficulties of the original problem (D.1.1) is precisely the fact that this is not the case: both functions are computed for the same variable.

We can perform step (D.1.6) minimising alternatively with respect to u and to d. The minimisation with respect to u is

$$u^{k+1} = \underset{u}{\operatorname{argmin}} \left\{ H(u) + \frac{\lambda}{2}\|d^k - \Phi(u) - b^k\|_2^2 \right\},$$

and so its difficulty depends on function H. The second step is

$$d^{k+1} = \underset{d}{\operatorname{argmin}} \left\{ \|d\|_1 + \frac{\lambda}{2}\|d - \Phi(u^{k+1}) - b^k\|_2^2 \right\},$$

and has the closed formula

$$d_j^{k+1} = \text{shrink}\left(\Phi(u^{k+1})_j + b_j^k, \frac{1}{\lambda} \right)$$

where shrink $(\cdot)$ is the soft thresholding operator, i.e. $\text{sgn}(x)(|x| - \lambda)_+$.

To summarise, Algorithm D.1 will provide a solution to the problem in (D.1.1). The authors of [15] suggest to set the parameter $N = 1$. No suggestions are given for the starting points of u, d and b, nor for the value λ.

Algorithm D.1 Generalised split Bregman algorithm (adapted from Osher)

Input: H, Φ, $\tau > 0$, $\lambda > 0$, integer N

Initialisation: u, d, b

```
while ‖u^k − u^{k−1}‖ > τ
    for n = 1 to N
        u^{k+1} = argmin_u { H(u) + (λ/2)‖d^k − Φ(u) − b^k‖_2^2 }
        d_j^{k+1} = shrink( Φ(u^{k+1})_j + b_j^k, 1/λ )
    end
    b^{k+1} = b^k + (Φ(u^{k+1}) − d^{k+1})
end
```

Output: solution $(\hat{u}, \hat{d})$.

D.2 Special case of Grid-C Constraints

We will use Algorithm $D.1$ to find the value of the function Ω for the Grid-C case (see § 3.3.4). The constraints set is $\Lambda_\alpha = \{\lambda : \|L\lambda\|_1 \leq \alpha\}$, where α is a positive parameter and L is the incidence matrix of a DAG. In the 1D case, matrix L will be the $n \times (n-1)$ matrix with 1 on the main diagonal, -1 on the superdiagonal and 0 otherwise. This corresponds to the constraints set $\Lambda_\alpha = \{\lambda : \sum_i |\lambda_i - \lambda_{i+1}| \leq \alpha\}$.

Our goal is to find the infimum of $\frac{1}{2}\sum_i \left(\frac{\beta_i^2+\epsilon}{u_i} + u_i\right) \equiv H(u) + \frac{\|u\|_1}{2}$, where the components of the vector β are perturbed by a slight positive amount ϵ for numerical stability. Moreover, the components of u are constrained to be nonnegative and to satisfy $\|Lu\|_1 \leq \alpha$. We can rewrite our problem as

$$\min_{u \succ 0} \left\{ H(u) + \frac{\|u\|_1}{2} + \lambda(\|Lu\|_1 - \alpha) \right\}, \tag{D.2.1}$$

for a positive Lagrangian multiplier λ.

We apply the splitting technique by introducing two new variables d and e, and by enforcing the constraints $d = u$ and $e = Lu$ via quadratic error terms. This allow us to compute function H and the ℓ_1 norm on different vectors. The new form of the problem is

$$\min_{\substack{u \succ 0 \\ d,e}} \left\{ H(u) + \frac{\|d\|_1}{2} + \lambda(\|e\|_1 - \alpha) + \frac{\mu}{2}\|d - u\|_2^2 + \frac{\mu}{2}\|e - Lu\|_2^2 \right\}, \tag{D.2.2}$$

for a positive weight μ. No real benefit is gained from weighing the two quadratic terms with different parameters. This problem is equivalent to the original problem (D.2.1), and is similar to (D.1.5), so that Algorithm D.1 can be applied.

The efficiency of the algorithm depends on how fast we can solve the step of minimisation

with respect to u, that is

$$u^{k+1} = \underset{u \succ 0}{\operatorname{argmin}} \left\{ H(u) + \frac{\mu}{2} \left(\|d^k - u - b_d^k\|_2^2 + \|e^k - Lu - b_e^k\|_2^2 \right) \right\}. \tag{D.2.3}$$

We added two further vector variables b_d and b_e to absorb the error for the new constraints, where in step (D.1.3) we only needed one variable. If we let $v = d^k - b_d^k$ and $w = e^k - b_e^k$, then the solution to (D.2.3) will be a function fpnt(u, v, w). We will see later that this function can be computed using a fixed point technique. Note that this function also depends on H, that is on β, but this will be omitted for simplicity.

The minimisation with respect to d is made again using the shrink operator. In this case the step is

$$d_j^{k+1} = \text{shrink}\left((u^{k+1})_j + (b_d^k)_j, \frac{1}{2\mu} \right). \tag{D.2.4}$$

Finally, the updates of the error variables are

$$b_d^{k+1} = b_d^k + (u^{k+1} - d^{k+1}) \tag{D.2.5}$$

$$b_e^{k+1} = b_e^k + (u^{k+1} - e^{k+1}). \tag{D.2.6}$$

Using all the updating steps together, we can now show Algorithm D.2 which provides a solution to the problem (D.2.1).

Algorithm D.2 Bregman method for function Ω

Input: β, $\tau > 0$, $\mu > 0$, integer N

Initialisation: u, d, b_e, b_d

`while` $\|u^k - u^{k-1}\| > \tau$

`for` $n = 1$ `to` N

$u^{k+1} = \text{fpnt}\left(u^k, d^k - b_d^k, e^k - b_e^k\right)$

$d_j^{k+1} = \text{shrink}\left((u^{k+1})_j + (b_d^k)_j, \frac{1}{2\mu} \right)$

`end`

$b_d^{k+1} = b_d^k + (u^{k+1} - d^{k+1})$

$b_e^{k+1} = b_e^k + (u^{k+1} - e^{k+1})$

`end`

Output: solution $\hat{u}$.

We now describe how to solve (D.2.3) using the fixed point theory that can be found in [33]. Consider the problem

$$\min_{u \succ 0} \left\{ \frac{1}{2} \sum_i \frac{z_i}{u_i} + c\|v - u\|_2^2 + c\|w - Lu\|_2^2 \right\}, \tag{D.2.7}$$

where $z_i = \beta_i^2$, $c = \frac{\mu}{2}$ for all i.

By considering the first two terms, we have the problem

$$\min\left\{\frac{1}{2}\|u-v\|_2^2 + \frac{1}{4c}\sum_i \frac{z_i}{u_i} + \delta_{\{u\geq 0\}}\right\} = \min\left\{\frac{1}{2}\|u-v\|_2^2 + h_1(u)\right\}, \tag{D.2.8}$$

that is the proximity operator of the function $h_1(u) = \omega_1(Bu)$, with $B = I$, the identity. By the theory of the fixed point, h_1 can be computed if we know the proximity operator of $\frac{\omega_1}{\lambda}$, with λ a positive constant. That is, we need

$$\text{prox}_{\frac{\omega_1}{\lambda}}(t) = \underset{\phi}{\text{argmin}}\left\{\frac{1}{2}\|t-\phi\|_2^2 + k\sum_i \frac{\beta_i^2}{\phi_i} + \delta_{\{\phi\geq 0\}}\right\}.$$

with $k = \frac{1}{4\lambda c}$. Since the variables are decomposable, we can solve the problem componentwise, which involves taking the positive root of a cubic polynomial[1].

Considering now the second two terms of (D.2.7) we have the problem

$$\min\left\{\frac{1}{2}\|u-v\|_2^2 + \frac{1}{2}\|w-Lu\|_2^2 + \delta_{\{u\geq 0\}}\right\} = \min\left\{\frac{1}{2}\|u-v\|_2^2 + h_2(u)\right\}.$$

Again, this is the composition of a proximity operator, in this case $h_2(u) = \omega_2(Bu)$, $B = I$, so we need

$$\text{prox}_{\frac{\omega_2}{\lambda}}(t) = \underset{\phi}{\text{argmin}}\left\{\frac{1}{2}\|t-\phi\|_2^2 + \frac{1}{2\lambda}\|w-L\phi\|_2^2\right\} = \left(I + \frac{L^TL}{\lambda}\right)^{-1}\left(t + \frac{L^Tw}{\lambda}\right),$$

computed finding the minimum of the quadratic form.

We define a function H which is a composition of the proximity operators we have found so far and an affine map A:

$$H(t) = I - \text{prox}_{\frac{\omega_1+\omega_2}{\lambda}}(A(t)).$$

The solution to the original problem is the fixed point of H, that is $H(r) = r$. This is obtained by applying the Picard iteration using Opial's theorem

$$r \to \frac{1}{2}I + \frac{1}{2}H(r)$$

until convergence. This is usually very fast.

In the fixed point theory, the proximity operator of the composed function $\omega \circ B$ at the point x can be found as the fixed point of H where

$$A(z) = (I - \lambda BB^T)z + Bx,$$

[1]For completeness, it is $\hat{\phi}_i = r + \frac{t_i}{3}$, with $r = \sqrt[3]{\alpha+\beta} + \sqrt[3]{\alpha-\beta}$, $\alpha = \frac{t_i^3}{27} + \frac{kz_i}{2}$, $\beta = \sqrt{\alpha^2 + \frac{p^3}{27}}$, $p = -\frac{t_i^3}{3}$.

For a point $t \in \mathbb{R}^n$, we have computed $\omega_1(Bt)$ and $\omega_2(Bt)$. So

$$H(t) = (I - \text{prox}_{\frac{\omega_1}{\lambda}}(A(t)) - \text{prox}_{\frac{\omega_2}{\lambda}}(A(t)).$$

Finally, our two proximity functions are computed on different vectors that are bound together by setting $t = (t_1, t_2)$, $B = [I; I]^T$.

With this algorithm we considered just the function Ω. If we are interested in minimising the regularised loss with this function, we have to introduce a further step when we minimise with respect to β:

$$\hat{\beta} = \left(\frac{2}{n}X^T X + \gamma \text{diag}(u_1^{-1}, \ldots, u_n^{-1})\right)^{-1} \frac{2}{n}X^T y.$$

Appendix E

Dual Problem and QCQP Formulation

The content of this section is not used in the thesis, but we include it here for completeness. We derive the dual of problem (4.2.1) when $\Lambda = \{\lambda : \lambda \in \mathbb{R}^n_{++}, A\lambda \in S\}$, where $A \neq 0$ is a prescribed $k \times n$ matrix and S is a convex set. We are particularly interested in either the case that S is a convex cone or $S = \{\|\cdot\| \leq 1\}$, where $\|\cdot\|$ is an *arbitrary* norm. These cases are described in § 3.3.4. We will show that the dual formulation, in the sense of the Fenchel duality, is, in many cases of interest, a *a quadratically constrained quadratic program* (QCQP).

E.1 Norm Constraints

We first study problem (3.1.1) in the case that

$$\Lambda = \Lambda_{A,\|\cdot\|} := \{\lambda : \lambda \in \mathbb{R}^n_{++}, \|A\lambda\| \leq 1\}.$$

Define $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $g : \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$ as

$$f(b,\xi) = \begin{cases} \frac{b^2}{\xi} + \xi & \text{if } \xi > 0 \\ +\infty & \text{if } \xi \leq 0 \end{cases}$$

and $g(\zeta,\eta) = \frac{1}{2}\|\zeta - y\|_2^2 + \delta_{\mathcal{B}}(\eta)$, where $\mathcal{B}$ is the unit ball of $\|\cdot\|$.

Note that the convex conjugate of g equals

$$g^*(p,q) = \frac{1}{2}\|p\|_2^2 + \langle p, y\rangle + \|q\|_*$$

where $\|\cdot\|_*$ denotes the dual norm of $\|\cdot\|$.

Lemma E.1.1. *The conjugate of f equals $f^* = \delta_{\mathcal{C}}$, where $\mathcal{C}$ is the parabolic region*

$$\mathcal{C} = \{(\gamma,\theta) \in \mathbb{R} \times \mathbb{R} : \gamma^2 + 4\theta \leq 4\}. \tag{E.1.1}$$

Moreover, $(b,\xi) \in \partial(f^*)(\gamma,\theta)$ if and only if

$$\begin{cases} \gamma = \frac{2b}{\xi},\ \xi > 0 \ \text{ and } \ \gamma^2 + 4\theta = 4 & \textbf{or} \\ b = \xi = 0 \ \text{ and } \ \gamma^2 + 4\theta \leq 4 & . \end{cases} \tag{E.1.2}$$

We can now obtain a dual problem of (4.2.1). Let us use x_i, a_i to denote the i-th columns of X, A, respectively.

Proposition E.1.1. *If $\Lambda = \Lambda_{A,\|\cdot\|}$, then problem* (4.2.1) *is equivalent to*

$$\min\Bigg\{\frac{1}{2}\|p-y\|_2^2 + \|q\|_* : p\in\mathbb{R}^m, q\in\mathbb{R}^k, \\ \langle x_i, p\rangle^2 + 2\rho\,\langle a_i, q\rangle \le \rho^2\ ,\ \forall i=1,\dots,n\Bigg\}. \quad \text{(E.1.3)}$$

Moreover, if $(\hat{p},\hat{q})$ is a solution of (E.1.3)*, then $(\hat{\beta},\hat{\lambda})$ is a solution of* (4.2.1) if and only if *the following equations hold*

$$\begin{cases} \hat{\beta}_i = \frac{1}{\rho}\langle x_i,\hat{p}\rangle\hat{\lambda}_i & \text{if } \langle x_i,\hat{p}\rangle^2 + 2\rho\,\langle a_i,\hat{q}\rangle = \rho^2 \\ \hat{\beta}_i = \hat{\lambda}_i = 0 & \text{if } \langle x_i,\hat{p}\rangle^2 + 2\rho\,\langle a_i,\hat{q}\rangle < \rho^2 \end{cases}, \quad \text{(E.1.4)}$$

for all $i=1,\dots,n$,

$$X\hat{\beta} = y - \hat{p}\,, \quad \text{(E.1.5)}$$

$$A\hat{\lambda} \in \operatorname{argmax}\{\langle -\hat{q},\eta\rangle : \|\eta\|\le 1\} \quad \text{(E.1.6)}$$

$$\hat{\lambda}\in\mathbb{R}^n_+\,. \quad \text{(E.1.7)}$$

Proof. We apply Fenchel's duality theorem [9, Thm. 3.3.5], noting that the Slater condition $0\in\operatorname{int}(\mathbb{R}^m\times\mathcal{B} - \mathcal{R}(X)\times A\mathbb{R}^n_{++})$ holds. Recalling the formula $(\frac{\rho}{2}f)^*(\cdot) = \frac{\rho}{2}f^*\left(\frac{2}{\rho}(\cdot)\right)$, we obtain the problem

$$\sup\Big\{-\frac{1}{2}\|p\|_2^2 + \langle p,y\rangle - \|q\|_* : p\in\mathbb{R}^m, q\in\mathbb{R}^k, \\ \left(\tfrac{2}{\rho}\langle x_i,p\rangle, \tfrac{2}{\rho}\langle a_i,q\rangle\right)\in\mathcal{C}\ ,\ \forall i=1,\dots,n\Big\},$$

which is equivalent to (E.1.3), and that the supremum is attained.

The primal-dual pair of solutions should satisfy the conditions

$$(\hat{\beta}_i,\hat{\lambda}_i)\in\partial(f^*)\left(\tfrac{2}{\rho}\langle x_i,\hat{p}\rangle, \tfrac{2}{\rho}\langle a_i,\hat{q}\rangle\right),$$

and

$$-(\hat{p},\hat{q})\in\partial g(X\hat{\beta}, A\hat{\lambda}).$$

These, combined with (E.1.2) and norm duality, yield conditions (E.1.4), (E.1.6) and (E.1.5).

■

We remark that in many cases of interest, dual problem (E.1.3) is a *quadratically constrained quadratic program* (QCQP) [10]. These include the case that $\|\cdot\|$ is a polyhedral norm, such as the ℓ_1 norm.

Recovering primal solutions from dual ones requires solving yet another optimisation problem. Thus, if $\hat{q} = 0$, the solutions satisfy $\|A\hat{\lambda}\| \leq 1$, equations (E.1.4), (E.1.5) and (E.1.7). If $\hat{q} \neq 0$, $\hat{\lambda}$ can be obtained by solving the problem

$$\min\Big\{\|A\lambda\| : \langle A^\top \hat{q}, \lambda\rangle = -\|\hat{q}\|_* \ , \ \lambda \in \mathbb{R}^n_+, \sum_{i\in J} \tfrac{1}{\rho}\lambda_i \langle x_i, \hat{p}\rangle x_i = y - \hat{p}\Big\}, \tag{E.1.8}$$

where J denotes the set of *active constraints*, that is, the indices for which $\langle x_i, \hat{p}\rangle^2 + 2\rho\,\langle a_i, \hat{q}\rangle = \rho^2$. In learning problems exhibiting sparsity, the set J has small cardinality and program (E.1.8) has a small number of variables. Moreover, in the case of polyhedral norms, (E.1.8) is a *linear program*.

E.2 Conic Constraints

Another case of interest imposes alternative constraints of a different character on λ. Namely, we consider the optimisation problem (4.2.1) when

$$\Lambda = \Lambda_{A,K} := \{\lambda : \lambda \in \mathbb{R}^n_{++}, A\lambda \in K\},$$

where K is a *convex cone*. As mentioned in [32], such cases correspond to the penalty function Ω in equation (3.1.2) being a norm.

To derive the corresponding dual problem we work as in Section E.1. In this case, however, the Slater condition is not automatically satisfied and we need an assumption on A and K. If a weaker Slater condition involving the *relative interior* of K, denoted by $\mathrm{ri}(K)$, holds then [45, Cor. 31.2.1] can be employed. To this end, we recall the concept of a *polar cone* – see, for example, [9, Sec. 3.3]. The polar cone of a set K is the set $K^- = \{\phi : \langle \phi, x\rangle \leq 0,\ \forall x \in K\}$. It is easy to see that $\delta^*_K = \delta_{K^-}$ and that $\phi \in \partial\delta_K(x), x \in K$, if and only if $\phi \in K^-, \langle \phi, x\rangle = 0$.

Proposition E.2.1. *If $\Lambda = \Lambda_{A,K}$ and there exists $\lambda \in \mathbb{R}^n_{++}$ such that $A\lambda \in \mathrm{ri}(K)$, then problem* (4.2.1) *is equivalent to*

$$\min\Bigg\{\frac{1}{2}\|p - y\|_2^2 : p \in \mathbb{R}^m, q \in -K^-, \\ \langle x_i, p\rangle^2 + 2\rho\,\langle a_i, q\rangle \leq \rho^2 \ , \ \forall i = 1, \ldots, n\Bigg\}. \tag{E.2.1}$$

Moreover, if $(\hat{p}, \hat{q})$ is a solution of problem (E.2.1), *then $(\hat{\beta}, \hat{\lambda})$ is a solution of problem* (4.2.1) if and only if *equations* (E.1.4), (E.1.5), (E.1.7) *and $A\hat{\lambda} \in K,\ \langle \hat{\lambda}, A^\top \hat{q}\rangle = 0$ hold.*

In most cases of interest, the Slater condition can be easily verified and K^- is known. For example, if $K = \mathbb{R}^n_+$, corresponding to the constraint $A\lambda \geq 0$, the cone is self-dual, meaning that $-K^- = \mathbb{R}^n_+$. In this case, (E.2.1) is a QCQP and the set of solutions $\hat{\lambda}$ is the polytope

$$\{\lambda : \lambda \in \mathbb{R}^n_+, A\lambda \in \mathbb{R}^k_+, \langle \lambda, A^\top \hat{q} \rangle = 0, \sum_{i \in J} \tfrac{1}{\rho} \lambda_i \langle x_i, \hat{p} \rangle x_i = y - \hat{p}\}.$$

Bibliography

[1] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[2] A. Argyriou, C.A. Micchelli, and M. Pontil. On spectral learning. *The Journal of Machine Learning Research*, 11:935–953, 2010.

[3] F. R. Bach, G. R. G Lanckriet, and M. I. Jordan. Multiple kernels learning, conic duality, and the smo algorithm. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[4] R.G. Baraniuk, V. Cevher, M.F. Duarte, and C. Hegde. Model-based compressive sensing. *Information Theory, IEEE Transactions on*, 56(4):1982 –2001, april 2010.

[5] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Sciences, 2 (1)*, pages 183–202, 2009.

[6] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[7] P.J. Bickel, Y. Ritov, and A.B. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Annals of Statistics*, 37:1705–1732, 2009.

[8] T. Blumensath and M. E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265 – 274, 2009.

[9] J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. CMS Books in Mathematics. Springer, 2005.

[10] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[11] F. Bunea, A.B. Tsybakov, and M.H. Wegkamp. Sparsity oracle inequalities for the Lasso. *Electronic Journal of Statistics*, 1:169–194, 2007.

[12] P.L. Combettes and V.R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation, 4 (4)*, pages 1168–1200, 2006.

[13] J.M. Danskin. The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14(4):641–664, 1966.

[14] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *ArXiv e-prints*, January 2010.

[15] T. Goldstein and S. Osher. The split bregman method for l1 regularized problems. *SIAM J. Imaging Sci. 2, 323*, 2009.

[16] A. Gramfort and M. Kowalski. Improving M/EEG source localization with an inter-condition sparse prior. In *IEEE International Symposium on Biomedical Imaging*, 2009.

[17] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. 2009.

[18] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, July 2003.

[19] R. R. Hocking. Developments in linear regression methodology: 1959-1982. *Technometrics*, 1983.

[20] J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 417–424. ACM, 2009.

[21] L. Jacob. Structured priors for supervised learning in computational biology. 2009. Ph.D. Thesis.

[22] L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *ICML*, volume 382 of *ACM International Conference Proceeding Series*. ACM, 2009.

[23] A. Jalali, P. Ravikumar, S. Sanghavi, and C. Ruan. A dirty model for multi-task learning. *Proc. Neural Info. Proc. Sys (NIPS), vol. 23*, 2010.

[24] R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. arXiv:0904.3523v3, 2010.

[25] S. Kim and E.P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. Technical report, 2009. arXiv:0909.1373.

[26] V. Koltchinskii and M. Yuan. Sparsity in multiple kernel learning. *Annals of Statistics*, 38(6):3660–3695, 2010.

[27] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[28] K. Lounici. Sup-norm convergence rate and sign concentration property of Lasso and Dantzig estimators. *Electronic Journal of Statistics*, 2:90–102, 2008.

[29] K. Lounici, M. Pontil, A.B. Tsybakov, and S. Van De Geer. Oracle Inequalities and Optimal Inference under Group Sparsity. *Arxiv preprint arXiv:1007.1771*, 2010.

[30] A. Maurer and M. Pontil. Structured sparsity and generalization. *Journal of Machine Learning Research*, 2011.

[31] C. A. Micchelli and M. Pontil. Feature space perspectives for learning the kernel. *Machine Learning*, 66:297–319, 2007.

[32] C.A. Micchelli, J.M. Morales, and M. Pontil. A family of penalty functions for structured sparsity. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1612–1623. 2010.

[33] C.A. Micchelli, L. Shen, and Y. Xu. Proximity algorithms for image models: denoising. *Inverse Problems, 27(4)*, 2011.

[34] S. Mosci, L. Rosasco, M. Santoro, A. Verri, and S. Villa. Solving Structured Sparsity Regularization with Proximal Methods. In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2010)*, pages 418–433, 2010.

[35] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301 – 321, 2009.

[36] Y. Nesterov. Gradient methods for minimizing composite objective function. *CORE*, 2007.

[37] G. Obozinski, B. Taskar, and M.I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):1–22, 2010.

[38] Z. Opial. Weak convergence of the sequence of successive approximations for nonexpansive operators. *Bulletin American Mathematical Society, 73*, pages 591–597, 1967.

[39] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Simul*, 4:460–489, 2005.

[40] A. B. Owen. A robust hybrid of lasso and ridge regression. Technical report, 2006.

[41] T. Park and G Casella. The bayesian lasso. *Journal of the American Statistical Association*, 2008.

[42] D. Percival, K. Roeder, R. Rosenfeld, and L. Wasserman. Structured, Sparse Regression With Application to HIV Drug Resistance. *ArXiv e-prints*, February 2010.

[43] T. Suzuki R. Tomioka. Regularization strategies and empirical bayesian learning for MKL. arXiv:1001.26151, 2011.

[44] F. Rapaport, E. Barillot, and J.P. Vert. Classification of arrayCGH data using fused SVM. *Bioinformatics*, 24(13):i375–i382, 2008.

[45] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[46] M. Schmidt, N. Le Roux, and F. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. *Technical Report, INRIA, HAL 00618152*, 2011.

[47] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[48] M. Szafranski, Y. Grandvalet, and P. Morizet-Mahoudeaux. Hierarchical penalization. In *In Advances in Neural Information Processing Systems 20*. MIT press, 2007.

[49] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 58(1):267–288, 1996.

[50] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[51] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *preprint*, 2008.

[52] S.A. van de Geer. High-dimensional generalized linear models and the Lasso. *Annals of Statistics*, 36(2):614, 2008.

[53] S. Villa, S. Salzo, L. Baldassarre, and Verri A. Accelerated and inexact forward-backward algorithms. *Optimization Online*, 08-2011.

[54] Z. Xiang, Y. Xi, U. Hasson, and P. Ramadge. Boosting with spatial regularization. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2107–2115. 2009.

[55] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for ℓ_1 minimization with applications to compressed sensing. *SIAM J. Imaging Sci*, 1:143–168.

[56] M. Yuan, R. Joseph, and H. Zou. Structured variable selection and estimation. *Annals of Applied Statistics*, 3(4):1738–1757, 2009.

[57] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[58] P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 37(6A):3468–3497, 2009.