A General Framework for Structured Sparsity via Proximal Optimization

Luca Baldassarre
UCL
London, UK
l.baldassarre@cs.ucl.ac.uk

Jean Morales
UCL
London, UK
j.morales@cs.ucl.ac.uk

Andreas Argyriou
TTI
Chicago, USA
argyriou@ttic.edu

Massimiliano Pontil
UCL
London, UK
m.pontil@cs.ucl.ac.uk

Abstract

We study a generalized framework for *structured sparsity*. It extends the well known methods of Lasso and Group Lasso by incorporating additional constraints on the variables as part of a convex optimization problem. This framework provides a straightforward way of favouring prescribed *sparsity patterns*, such as orderings, contiguous regions and overlapping groups, among others. Available optimization methods are limited to specific constraint sets and tend to not scale well with sample size and dimensionality. We propose a *first order proximal method*, which builds upon results on fixed points and successive approximations. The algorithm can be applied to a general class of conic and norm constraints sets and relies on a proximity operator subproblem which can be computed numerically. Experiments on different regression problems demonstrate state-of-the-art statistical performance, which improves over Lasso, Group Lasso and StructOMP. They also demonstrate the efficiency of the optimization algorithm and its scalability with the size of the problem.

1 Introduction

We study the problem of learning a sparse linear regression model. The goal is to estimate a parameter vector $\beta^* \in \mathbb{R}^n$ from a vector of 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 resulting from the presence of noise. We are interested in sparse estimation under additional conditions on the sparsity pattern of β^*. In other words, not only do we expect that β^* is sparse but also that it exhibits *structured sparsity*, namely certain configurations of its nonzero components are preferred to others. This problem arises in several applications, such as regression, image denoising, background subtraction etc. – see [9, 11] for a discussion.

Appearing in Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX. Copyright 2012 by the authors.

In this paper, we build upon the structured sparsity framework recently proposed by [12, 13]. It is a regularization method, formulated as a convex, non-smooth optimization problem over a vector of auxiliary parameters. This approach provides a constructive way to favor certain sparsity patterns of the regression vector β. Specifically, this formulation involves a penalty function given by the formula

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

This function can be interpreted as an extension of a well-known variational form for the ℓ_1 norm. The convex constraint set Λ provides a means to incorporate prior knowledge on the magnitude of the components of the regression vector. As we explain in Section 2, the sparsity pattern of β is contained in that of the auxiliary vector λ at the optimum. Hence, if the set Λ allows only for certain sparsity patterns of λ, the same property will be "transferred" to the regression vector β.

The first contribution of this paper is the introduction of a tractable class of regularizers of the above form which extend the examples described in [12, 13]. Specifically, we study in detail the cases in which the set Λ is defined by *norm* or *conic constraints*, combined with a linear map. As we shall see, these cases include formulations which can be used for learning *graph sparsity* and *hierarchical sparsity*, in the terminology of [9]. That is, the sparsity pattern of the vector β^* may consist of a few contiguous regions in one or more dimensions, or may be embedded in a tree structure. This sparsity problem may arise in several applications, ranging from functional magnetic resonance imaging [7, 25], to scene recognition in vision [8], to multi-task learning [1, 18] and to bioinformatics [20] – to mention but a few.

A main limitation of the technique described in [12, 13] is that in many cases of interest the penalty function cannot be easily computed. This makes it difficult to solve the associated regularization problem. For example, [12, 13] proposes to use block coordinate descent, but this method is feasible only for limited choices of the set Λ. The second contribution of this paper is an efficient accelerated proximal point method to solve regularized least squares with the penalty function $\Omega(\cdot|\Lambda)$ 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 [14] with an accelerated first order method equivalent to FISTA [4]. We present a numerical study of the efficiency of the proposed method and a statistical comparison of the proposed penalty functions with the greedy method of [9], the Lasso and the Group Lasso.

Recently, there has been significant research interest on structured sparsity and the literature on this subject is growing fast, see for example [1, 9, 10, 11, 26] and references therein for an indicative list of papers. In this work, we mainly focus on convex penalty methods and compare them to greedy methods [3, 9]. The latter provide a natural extension of techniques proposed in the signal processing community and, as argued in [9], allow for a significant performance improvement over more generic sparsity models such as the Lasso or the Group Lasso [26]. The former methods have until recently focused mainly on extending the Group Lasso, by considering the possibility that the groups overlap according to certain hierarchical structures [11, 27]. Very recently, general choices of convex penalty functions have been proposed [2, 12, 13]. In this paper we build upon [12, 13], providing both new instances of the penalty function and improved optimization algorithms.

The paper is organized as follows. In Section 2, we set our notation, review the method of [12, 13] and recall some basic facts from convex analysis. In Section 3, we provide some general insights on the method and introduce two new classes of sets Λ. In Section 4, we present our technique for computing the proximity operator of the penalty function and the resulting accelerated proximal method. In Section 5, we assess the efficiency and the statistical performance of the method via some numerical experiments.

2 Background

In this section, we introduce our notation, review the learning method which we study in this paper and recall some basic facts from convex analysis.

We let $\mathbb{R}_+$ and $\mathbb{R}_{++}$ be the nonnegative and positive real line, respectively. For every $\beta \in \mathbb{R}^n$ we define $|\beta| \in \mathbb{R}^n_+$ to be the vector $|\beta| = (|\beta_i|)_{i=1}^n$. For every $p \geq 1$, we define the ℓ_p norm of β as $\|\beta\|_p = (\sum_{i=1}^n |\beta_i|^p)^{\frac{1}{p}}$. We denote by $\langle \cdot, \cdot \rangle$ the standard inner product in $\mathbb{R}^n$, that is, if $x, t \in \mathbb{R}^n$, then $\langle x, t \rangle = \sum_{i=1}^n x_i t_i$. If $C \subseteq \mathbb{R}^n$, we denote by $\delta_C : \mathbb{R}^n \to \mathbb{R}$ the indicator function of the set C, that is, $\delta_C(x) = 0$ if $x \in C$ and $\delta_C(x) = +\infty$ otherwise. Finally, if $\varphi : \mathbb{R}^n \to \mathbb{R}$ is a convex function, we use $\partial\varphi(x)$ to denote the *subdifferential* of φ at $x \in \mathbb{R}^n$ [21].

We now review the structured sparsity approach of [12, 13]. Given an $m \times n$ input data matrix X and an output vector $y \in \mathbb{R}^m$, obtained from the linear regression model $y = X\beta^* + \xi$ discussed earlier, they consider the optimization problem

$$\inf \left\{ \frac{1}{2}\|X\beta - y\|_2^2 + \rho\,\Gamma(\beta, \lambda) : \beta \in \mathbb{R}^n, \lambda \in \Lambda \right\} \quad (2.1)$$

where ρ is a positive parameter, Λ is a prescribed convex subset of the positive orthant $\mathbb{R}^n_{++}$ and the function $\Gamma : \mathbb{R}^n \times \mathbb{R}^n_{++} \to \mathbb{R}$ is given by the formula

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

Note that the infimum over λ in general is not attained, however the infimum over β is always attained. Since the auxiliary vector λ appears only in the second term and our goal is to estimate β^*, we may also directly consider the regularization problem

$$\min \left\{ \frac{1}{2}\|X\beta - y\|_2^2 + \rho\,\Omega(\beta) : \beta \in \mathbb{R}^n \right\}, \quad (2.2)$$

where the penalty function takes the form

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

This problem is still convex because the function Γ is jointly convex [5]. Also, note that the function Ω is independent of the signs of the components of β.

For a generic convex set Λ, since the penalty function Ω is not easily computable, one needs to deal directly with problem (2.1). To this end, we recall here the definition of the proximity operator [15].

Definition 2.1. *Let φ be a real-valued convex function on $\mathbb{R}^d$. The proximity operator of φ is defined, for every $t \in \mathbb{R}^d$ by* $\operatorname{prox}_\varphi(t) := \operatorname{argmin}\left\{ \frac{1}{2}\|z - t\|_2^2 + \varphi(z) : z \in \mathbb{R}^d \right\}$.

The proximity operator is well-defined, because the above minimum exists and is unique.

3 The set Λ

In this section, we first provide some general insights on how the set Λ can favour certain sparsity patterns on β and, secondly, we introduce two new classes of sets Λ that can be used in many relevant applications.

We begin by noting that, for every $\lambda \in \mathbb{R}^n_{++}$, the quadratic function $\Gamma(\cdot, \lambda)$ provides an upper bound to the ℓ_1 norm, namely it holds that $\Omega(\beta) \geq \|\beta\|_1$ and the inequality is tight if and only if $|\beta| \in \Lambda$. This fact is an immediate consequence of the arithmetic-geometric inequality. In particular, we see that if we choose $\Lambda = \mathbb{R}^n_{++}$, the method (2.2) reduces to the Lasso[1]. The above observation suggests a heuristic interpretation of the method (2.2): 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 (2.2) is expected to be sparse. The penalty function therefore will encourage certain desired sparsity patterns.

The last point can be better understood by looking at problem (2.1). For every solution $(\hat{\beta}, \hat{\lambda})$, the sparsity pattern of $\hat{\beta}$ is contained in the sparsity pattern of $\hat{\lambda}$, that is, the indices associated with nonzero components of $\hat{\beta}$ are a subset of those of $\hat{\lambda}$. Indeed, if $\hat{\lambda}_i = 0$ it must hold that $\hat{\beta}_i = 0$ as well, since the objective would diverge otherwise (because of the ratio β_i^2/λ_i). Therefore, if the set Λ favors certain sparse solutions of $\hat{\lambda}$, the same sparsity pattern will be reflected on $\hat{\beta}$. Moreover, the regularization term $\sum_i \lambda_i$ favors sparse vectors λ. For example, a constraint of the form $\lambda_1 \geq \cdots \geq \lambda_n$, favors consecutive zeros at the end of λ and non-zeros everywhere else. This will lead to zeros at the end of β as well. Thus, in many cases like this, it is easy to incorporate a convex constraint on λ, whereas it may not be possible to do the same directly on β.

In this paper, we consider 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 $\Lambda = \{\lambda \in \mathbb{R}^n_{++} : A\lambda \geq 0\}$. For example, in [12, 13], 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 $\Omega(\cdot|\Lambda)$. This set can be used to encourage hierarchical sparsity. Note that, for a generic matrix A, the penalty function cannot be computed explicitly. In the next section, we present an optimization method that overcomes 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 which are constant within few connected regions.

[1]More generally, method (2.2) includes the Group Lasso method, see [12, 13].

4 Optimization Method

In this section, we discuss how to solve problem (2.1) using an accelerated first-order method that scales linearly with respect to the problem size, as we later show in the experiments. This method relies on the computation of the proximity operator of the function Γ, restricted to $\mathbb{R}^n \times \Lambda$. Since the exact computation of the proximity operator is possible only in simple special cases of sets Λ, we present here an efficient fixed-point algorithm for computing the proximity operator that can be applied to a wide variety of constraints. Finally, we discuss an accelerated proximal method that leverages our algorithm.

4.1 Computation of the Proximity Operator

According to Definition 2.1, the proximal operator of Γ at $(\alpha, \mu) \in \mathbb{R}^n \times \mathbb{R}^n$ is the solution of the problem

$$\min\left\{\frac{1}{2}\|(\beta,\lambda) - (\alpha,\mu)\|_2^2 + \rho\,\Gamma(\beta,\lambda) : \beta \in \mathbb{R}^n, \lambda \in \Lambda\right\}. \tag{4.1}$$

For fixed λ, a direct computation yields that the objective function in (4.1) attains its minimum at

$$\beta_i(\lambda) = \frac{\alpha_i \lambda_i}{\lambda_i + \rho}.$$

Using this equation we obtain the simplified problem

$$\min\left\{\frac{1}{2}\|\lambda - \mu\|^2 + \frac{\rho}{2}\sum_{i=1}^{n}\left(\frac{\alpha_i^2}{\lambda_i + \rho} + \lambda_i\right) : \lambda \in \Lambda\right\}. \tag{4.2}$$

This problem can still be interpreted as a proximity map computation. We discuss how to solve it under our general assumption $\Lambda = \{\lambda : \lambda \in \mathbb{R}^n_{++}, A\lambda \in S\}$. Moreover, we assume that the projection on the set S can be easily computed. To this end, we define the $(n+k) \times n$ matrix

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

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

$$\varphi_1(s) = \frac{\rho}{2}\sum_{i=1}^{n}\left(\frac{\alpha_i^2}{s_i + \rho} + s_i + \delta_{\mathbb{R}_{++}}(s_i)\right),$$

and $\varphi_2(t) = \delta_S(t)$. Note that the solution of problem (4.2) is the same as the proximity map of the linearly composite function $\varphi \circ B$ at μ, which solves the problem

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

At first sight this problem seems difficult to solve. However, it turns out that if the proximity map of the function φ has a simple form, the following theorem adapted from [14, Theorem 3.1] can be used to accomplish this task. For ease of notation we set $d = n + k$.

Theorem 4.1. *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*

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

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

$$\operatorname{prox}_{\varphi \circ B}(\mu) = \mu - cB^\top \hat{v}$$

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 $I - \operatorname{prox}_\varphi$ is *nonexpansive* [2] (see e.g. [6]), 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. *[19] 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.*

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

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

Both $\operatorname{prox}_{\varphi_1}$ and $\operatorname{prox}_{\varphi_2}$ 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 4.1. *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 Accelerated Proximal Method

Theorem 4.1 motivates a proximal numerical approach (Algorithm 1 below) to solving problem (2.1) and, in turn, problem (2.2). Let $E(\beta) = \frac{1}{2}\|X\beta - y\|_2^2$ and assume an upper bound L of $\|X^\top X\|$ is known.[3] Proximal first-order methods – see [4, 6, 17, 23] and references therein – can be used for nonsmooth optimization, where the objective consists of the sum of a smooth term and a non-smooth term, 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 $u_t := (\beta_t, \lambda_t) \leftarrow \operatorname{argmin}\left\{\frac{L}{2}\|(\beta, \lambda) - (w_t - \frac{1}{L}\nabla E(w_t))\|_2^2 + \rho\,\Gamma(\beta, \lambda) : \beta \in \mathbb{R}^n, \lambda \in \Lambda\right\}$. 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.

Algorithm 1
Proximal structured sparsity algorithm (NEPIO).

```
$u_1, w_1 \leftarrow$ arbitrary feasible values
for t=1,2,... do
    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 for
```

The simplest (and a commonly used) update rule is $w_t = u_t$. By contrast, *accelerated proximal methods* proposed by [17] 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 proximal operator is computed *numerically*, it has been shown only very recently [22, 24] that, under some circumstances, the accelerated method still converges with the rate $O\left(\frac{1}{T^2}\right)$. The main advantages of accelerated methods are their low cost per iteration and their scalability to large problem sizes. Moreover, in applications where a thresholding operator is involved – as in Lemma 4.1 – the zeros in the solution are exact, which may be desirable.

For our purposes, we use a version of accelerated methods influenced by [23] (described in Algorithm 1). According to Nesterov, 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 recursive equation

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

We have adapted [23, Algorithm 2] (equivalent to FISTA [4]) by computing the proximity operator of $\frac{\varphi}{L} \circ B$ using the Picard-Opial process described in Section 4.1. 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

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

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

iteration, the map H_t is defined by

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

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 4.1, the fixed point process combined with the assignment of u are equivalent to $u_{t+1} \leftarrow \mathrm{prox}_{\frac{\phi}{L}\circ B}\big(w_t - \frac{1}{L}\nabla E(w_t)\big)$.

The reason for resorting to Picard-Opial is that exact computation of the proximity operator (4.2) 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 (2.1) or (2.2) and alternatives like interior point methods do not scale well with problem size. In the next section, we will demonstrate empirically the scalability of Algorithm 1, as well as the efficiency of both the proximity map computation and the overall method.

5 Numerical Simulations

In this section, we present experiments with method (2.1). The main goal of the experiments is to study both 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" [9], "GL1", the Group Lasso variant presented in [10], and "GL2", a Group Lasso with overlapping groups. For both Group Lasso methods, we used as groups all sets of 4 contiguous variables (1D) or the sets of all neighbours of each variable (2D). Moreover, we used method (2.1) with the following choices for the constraint set Λ:

- $\Lambda = \{\lambda : \|A\lambda\|_1 \le \alpha\}$, where A is the edge map of a 1D or 2D grid – we refer to the corresponding method as "Grid-C".
- $\Lambda = \{\lambda : A\lambda \ge 0\}$, where A is the edge map of a tree graph – we refer to the corresponding method as "Tree-C".

We solved the optimization problem (2.1) either with the toolbox CVX[4] or with the proximal method presented in Section 4. When using the proximal method, we found that setting the parameter κ from Opial's Theorem to 0.2 gave the best results, even though in [14] they show that convergence of the fixed-point iterations is guaranteed also for $\kappa = 0$. Our main stopping criterion is based on the decrease in the objective value of (2.1) which must be 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 (if this were not possible, a Frobenius estimate could be used). Finally, the implementation ran on an 8GB memory quad-core Intel machine.

[4] `http://cvxr.com/cvx/`

5.1 Efficiency experiments

First, we investigated the computational properties of the proximal method (NEPIO). 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 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, but do not scale well with the problem size. We also compared to the non-accelerated version of our algorithm, similar to ISTA [4, 6]. ISTA has been shown [6] to converge in the presence of very general, but summable, errors in the computation of the proximity operator. In the case of the Tree constraint, we further compared with an adapted version of the alternating algorithm (AA) of [12, 13]. For each problem size, we repeated the experiments 10 times and we report the average computation time in Figure 1 for Grid-C and Tree-C. All methods achieve objective values that are within 1% of each other, apart from ISTA that sometimes did not converge after 10^5 iterations. The proposed method scales linearly with the problem size, making it suitable for large scale experiments.

In order to empirically assess the importance of the Picard-Opial tolerance for converging to a good solution, we considered a problem with 100 variables for both the Grid and the Tree constraints and repeated the experiments 100 times with different sampling of training examples. For each constraint, we evaluated the average distance from the solution obtained by our method with different values of the Picard-Opial tolerance to the solution obtained by CVX.

We did not observe any improvement in the solution by decreasing a fixed tolerance from 10^{-2} to 10^{-8} or by setting the tolerance to decrease as $1/T^\alpha$ with α equal to $1, 1.5$ or 2, as suggested by very recent results [22, 24]. However, decreasing the tolerance remarkably increased the computational overhead from an average of $5s$ for a fixed tolerance of 10^{-2} to $80s$ for $1/T^2$ in the case of the Grid constraint.

Finally, we considered the 2D Grid-C case and observed

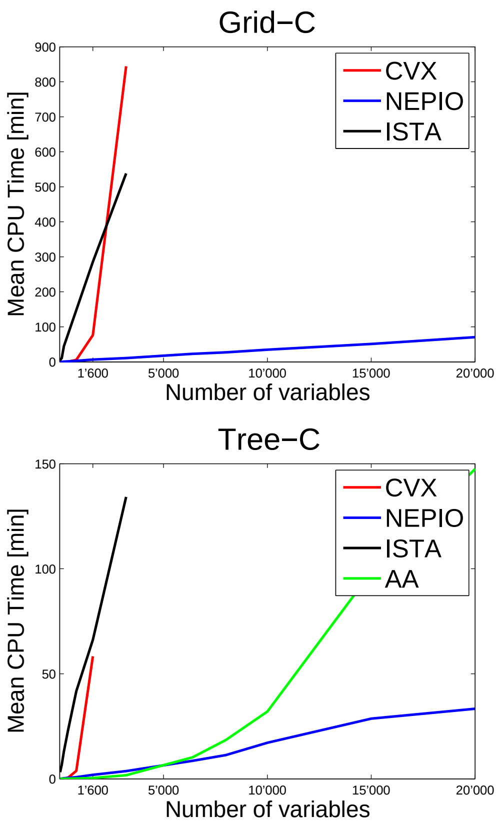

Figure 1: Computation time vs problem size for Grid-C and Tree-C for different optimization methods. For the Tree constraint, CVX was not able to deal with problem sizes bigger than 1600.

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.

In all the following statistical experiments we used a fixed tolerance of 10^{-2}.

5.2 Statistical experiments

One dimensional contiguous regions. In the first experiment we chose a model vector $\beta^* \in \mathbb{R}^{200}$ with 40 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 40, 20, 13 and 10, respectively. We generated a noiseless output from a matrix X whose elements have a standard Gaussian distribution and whose columns are normalized. The estimates $\hat{\beta}$ for several models are then compared with the original β^*. Figure 2 shows the model error $\frac{\|\hat{\beta}-\beta^*\|_2}{\|\beta^*\|_2}$ as the sample size changes from 42 (barely above the sparsity)

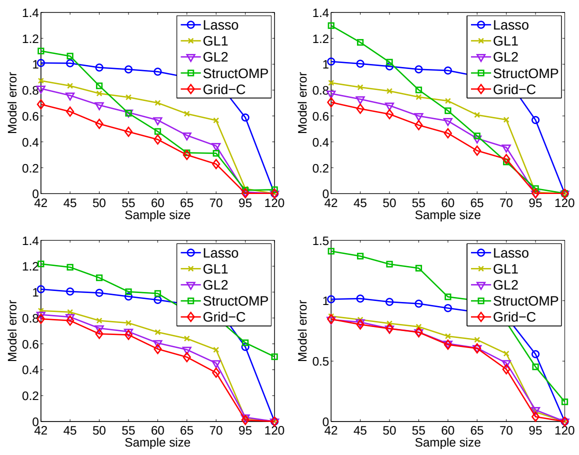

Figure 2: 1D contiguous regions: comparison between different methods for one (top-left), two (top-right), three (bottom-left) and four (bottom-right) regions.

up to 120 (more than half the dimensionality). These errors are 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. The length of the groups for the Group Lasso variants was selected among 2, 4 and 8, but only the best performers (lengths 4 in both cases) are represented here. Even then, we observe that Grid-C is comparable to both methods and sometimes shows a strong improvement over them. For one particular run with a sample size which is twice the model sparsity, Figure 3 shows the original vector and the estimates for different methods.

Two dimensional contiguous regions. We also repeated the same experiment in the case that the sparsity pattern of $\beta^* \in \mathbb{R}^{20\times 20}$ consists of 2D continuous regions. For the sparsity pattern, we consider either a single 7×7 region, two 5×5 regions, three 4×4 regions or four 4×3 regions. Figure 4 shows the model error versus the sample size in this case. Figure 5 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. Again, we observe that both Group Lasso variants for contiguous 2D regions are never better than Grid-C, and sometimes show a higher model error. For brevity, we will not include these methods in the next experiments

Background subtraction. We replicated the experiment from [9, Sec. 7.3] in which the underlying model β^* corresponds to the nonzero pixels of the foreground of a CCTV image. We measured the output as a random projection plus Gaussian noise. Figure 7-*Left* shows that, while the Grid-C outperforms the Lasso, it is not as good as StructOMP in

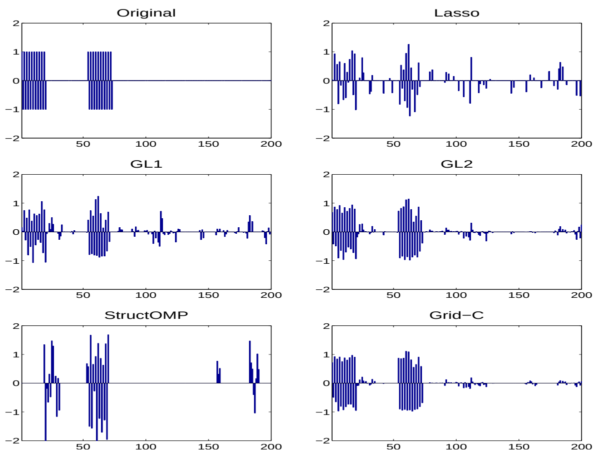

Figure 3: Two 1D contiguous regions: regression vector estimated by different models: (top-left) β^*, (top-right) Lasso, (centre-left) GL1, (centre-right) GL2, (bottom-left) StructOMP, (bottom-right) Grid-C.

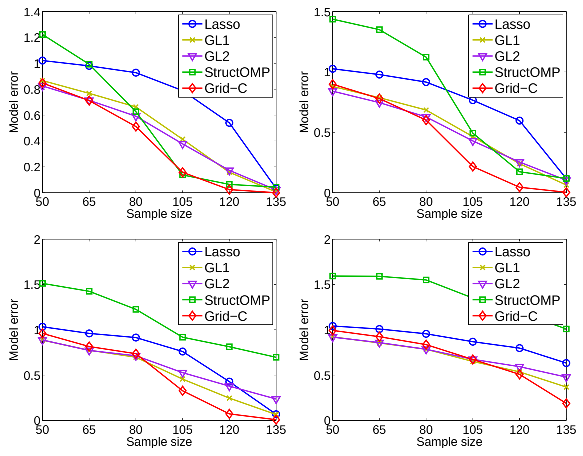

Figure 4: 2D contiguous regions: comparison between different methods for one (top-left), two (top-right), three (bottom-left) and four (bottom-right) regions.

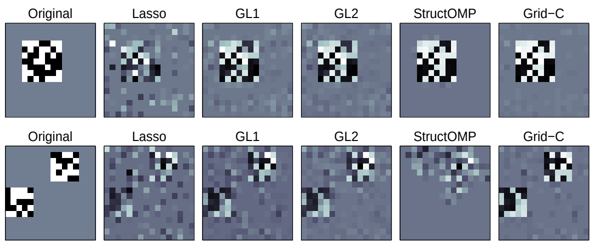

Figure 5: 2D-contiguous regions: Original regression vector and regression vector estimated by the Lasso, GL1, GL2, StructOMP and Grid-C (left to right) for one region (top) and two regions (bottom).

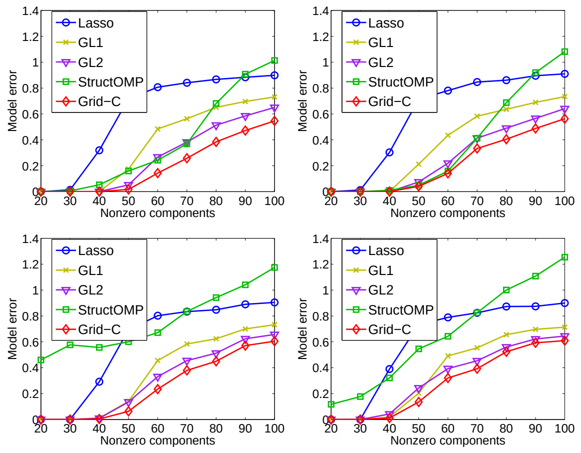

Figure 6: 1D contiguous regions: model error against sparsity for a fixed sample size (100 points). Number of contiguous regions, from left to right, top to bottom: one, two, three and four.

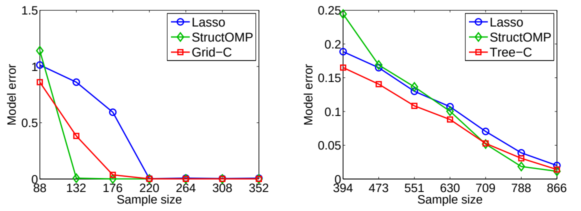

Figure 7: Average model error for the background subtraction (top) and *cameraman* (bottom) experiments.

this case. We speculate that the reason for this result is 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 the proposed method (Tree-C) on an instance of 2D image compressive sensing, following the experimental protocol of [9]. 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 sample image: *cameraman*. We generated a projection matrix X whose entries are i.i.d. draws from a standard Gaussian distribution. Zero-mean Gaussian noise with standard deviation $\sigma = 0.01$ was added to the measurements. StructOMP and Tree-C, both exploiting the tree structure, were used to recover the wavelet coefficients from the random projection measurements and compared to the Lasso. The inverse wavelet transform was used to reconstruct the images with the recovered wavelet coefficients. The recovery performances of the methods are reported in Figure 7-*Right*, which highlights the good performance of Tree-C.

6 Conclusion

We proposed new families of penalties and presented a new algorithm and results on the class of structured sparsity penalty functions proposed by [12, 13]. 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 [9], the Lasso and two Group Lasso variants. 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 applicability of the recent results regarding accelerated proximal method with inexact computation of the proximity operator [22, 24] to our case 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.

Acknowledgements

We are grateful to Charles Micchelli for many valuable discussions. This work was supported by EPSRC Grant EP/H027203/1 and Royal Society International Joint Project Grant 2012/R2.

References

[1] Argyriou, A., Evgeniou, T., and Pontil, M. Convex multi-task feature learning. *Machine Learning*, 73(3): 243–272, 2008.

[2] Bach, F. Structured sparsity-inducing norms through submodular functions. In *Advances in Neural Information Processing Systems 23*, pp. 118–126. 2010.

[3] Baraniuk, R.G., Cevher, V., Duarte, M.F., and Hegde, C. Model-based compressive sensing. *Information Theory, IEEE Transactions on*, 56(4):1982–2001, 2010.

[4] Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Sciences*, 2(1):183–202, 2009.

[5] Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

[6] Combettes, P.L. and Wajs, V.R. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200, 2006.

[7] A. Gramfort and M. Kowalski. Improving M/EEG source localization with an inter-condition sparse prior, preprint, 2009.

[8] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. 2009. In *IEEE International Symposium on Biomedical Imaging*, 2009.

[9] Huang, J., Zhang, T., and Metaxas, D. Learning with structured sparsity. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 417–424. ACM, 2009.

[10] Jacob, L., Obozinski, G., and Vert, J.-P. Group lasso with overlap and graph lasso. In *International Conference on Machine Learning (ICML 26)*, 2009.

[11] Jenatton, R., Audibert, J.-Y., and Bach, F. Structured variable selection with sparsity-inducing norms. arXiv:0904.3523v2, 2009.

[12] Micchelli, C.A., Morales, J.M., and Pontil, M. A family of penalty functions for structured sparsity. In *Advances in Neural Information Processing Systems 23*, pp. 1612–1623. 2010.

[13] Micchelli, C.A., Morales, J.M., and Pontil, M. Regularizers for structured sparsiy. *Advances in Computational Mathematics*, pp. 1-35, 2011.

[14] Micchelli, C.A., Shen, L., and Xu, Y. Proximity algorithms for image models: denoising. *Inverse Problems*, 27(4), 2011.

[15] Moreau, J.J. Fonctions convexes duales et points proximaus dans un espace Hilbertien. *Acad. Sci. Paris Sér. A Math.*, 255:2897–2899, 1962.

[16] Mosci, S., Rosasco, L., Santoro, M., Verri, A., and Villa, S. Solving Structured Sparsity Regularization with Proximal Methods. Proc. of ECML, pages 418–433, 2010.

[17] Nesterov, Y. *Gradient methods for minimizing composite objective function*. CORE, 2007.

[18] 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.

[19] Opial, Z. Weak convergence of the sequence of successive approximations for nonexpansive operators. *Bulletin American Mathematical Society*, 73: 591–597, 1967.

[20] F. Rapaport, E. Barillot, and J.P. Vert. Classification of arrayCGH data using fused SVM. *Bioinformatics*, 24(13):i375–i382, 2008.

[21] Rockafellar, R. T. *Convex Analysis*. Princeton University Press, 1970.

[22] Schmidt, M., Le Roux, N. and Bach, F. Convergence Rates of Inexact Proximal-Gradient Methods for Convex Optimization. Technical Report, INRIA, HAL 00618152, 2011

[23] Tseng, P. On accelerated proximal gradient methods for convex-concave optimization. 2008. Preprint.

[24] Villa, S., Salzo, S., Baldassarre, L. and Verri A. Accelerated and Inexact forward-backward algorithms. Optimization Online, 08-2011

[25] 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.

[26] Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B,* 68(1):49–67, 2006.

[27] Zhao, P., Rocha, G., and Yu, B. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 37(6A):3468–3497, 2009.