International Joint Conference on Neural Networks (IJCNN)
Killarney, 2015

DOI: https://doi.org/10.1109/IJCNN.2015.7280612

The Generalized Group Lasso

Carlos M. Alaíz*† and José R. Dorronsoro*
*Dpto. Ing. Informática & Instituto de Ingeniería del Conocimiento, Universidad Autónoma de Madrid, 28049 Madrid, Spain
†Dpto. Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid, 28911 Leganés (Madrid), Spain
{carlos.alaiz,jose.dorronsoro}@uam.es

***Abstract*—In this paper the Generalized Lasso model of R. Tibshirani is extended to consider multidimensional features (or groups of features) à la Group Lasso, by substituting the ℓ_1 norm of the regularizer by the $\ell_{2,1}$ norm. The resultant model is called Generalized Group Lasso (GenGL), and it contains as particular cases the already known Group Lasso and Group Fused Lasso (GFL), but also new models as the Graph-Guided Group Fused Lasso, or the trend filtering for multidimensional features. We show how to solve them efficiently combining FISTA iterations with the Proximal Operator of the corresponding regularizer, which we compute using a dual formulation. Moreover, GenGL makes possible to introduce a new approach to Group Total Variation, the regularizer of GFL, that results in a training much faster than that of previous methods.**

I. Introduction

Big data problems are characterized by either a large number of patterns or a high dimensionality, or both, and they have caused a renewed interest on simple linear models. Indeed, when the dimensionality is very high, the expressivity of a linear model could be enough to solve certain problems, with the advantages of being very efficient to apply and easy to interpret. Nevertheless, linear models often require some kind of regularization to avoid over-fitting and also to get some structure into their weight vector. The ℓ_1 norm has been the standard choice for introducing such a structure, not only to enforce pure sparsity (i.e. to select only certain features automatically), but also because more general structures can be identified and modelled by the ℓ_1 penalization of a linear transformation of the weight vector. In fact, many resulting models can be dealt in a unified way under the Generalized Lasso (GenLA) recently proposed by R. Tibshirani [1].

On the other hand, in certain situations input features have a natural group structure, i.e., the input is made up of d_g groups with d_v features each, and thus the total input dimension is $d_g d_v$. In such cases the structure should be imposed at group level. For example, instead of obtaining a sparse vector of weights, where some of the coefficients are identically zero, it would be more convenient to have sparsity at the group level, so all the variables corresponding to a certain group should be either zero or non-zero. This is why models as the Group Lasso arise, which substitute the ℓ_1 norm by the $\ell_{2,1}$ norm (the sum of the ℓ_2 norms of the groups). Another example is the Group Fused Lasso [2], which generalizes the Fused Lasso and that has in the Proximal Operator (POp) of the Group Total Variation (GTV) regularizer an important particular case with applications in image processing.

In this paper, and continuing with this analogy between the ℓ_1 and the $\ell_{2,1}$ regularizers, we propose the Generalized Group Lasso model (GenGL) as an extension of the GenLA of [1] in which a group penalty is applied to linear transformations of the group-structured weight vector. As it shall be shown, depending on the transformation matrix used, classical group models are recovered, and also new models are obtained. Moreover, we propose a method to solve the POp of the GenGL regularizer through a dual formulation of the original problem. In summary, the main contributions of this paper are:

- The definition of the GenGL model, which contains as particular instances other state-of-the-art group linear models but that allows to apply a group approach to other regularized problems such as trend filtering or graph-guided penalizations, as well as a relatively simple computational solution for certain regularizers that are sums of non-differentiable convex functions.
- A new treatment of GTV when the variable groups have a multi-axis organization, as it is the case with images, for which we introduce an appropriate transformation matrix that simplifies the solution of the problem.
- An empirical confirmation of the computational advantage of this approach with respect to the splitting of the multidimensional regularizer into 1-dimensional ones and their combination through Proximal Dykstra.

The remaining of the paper is structured as follows. Section II contains a brief summary of the theory on Proximal Methods that will be used to train the proposed linear models. While not new, this section may have an interest of its own given that we cover the different pieces that have to be put together when solving many sparse convex optimization problems. Section III reviews the GenLA and proposes the new GenGL model. Some experiments are described in Section IV to show the potential of this new model, and finally the paper ends in Section V with a discussion and conclusions, as well as pointers to further work.

II. Proximal Methods for Structured Linear Models

A. Structured Linear Models

Learning regularized models can be characterized as solving an optimization problem of the form

$$\min_w \ \{f(w)\} = \min_w \ \{f_{\mathrm{L}}(w) + \lambda f_{\mathrm{R}}(w)\} \ , \qquad (1)$$

where f is the complete objective function to be minimized, which evaluates the quality of the model specified by the parameters w. This objective can be split into f_{L}, the loss term that determines how well the model fits the data, and f_{R}, the regularization term that can be used to avoid over-fitting, impose certain structure or introduce prior knowledge, among others. The parameter λ controls the strength of the regularizer.

In the particular case of linear regression models, where the output is estimated via a linear combination of the inputs (for the sake of clarity we omit the bias term, which can be introduced just through minor modifications of f_{L}, as in general the bias is not regularized) and a mean squared error loss is used, Prob. (1) can be expressed in matrix form as:

$$\min_{w\in\mathbb{R}^d} \left\{ \|Xw - y\|_2^2 + \lambda f_{\mathrm{R}}(w) \right\} ,$$

where the following notation has been used: *i*) $X \in \mathbb{R}^{p\times d}$ is the data matrix, which contains the input patterns as rows; *ii*) $y \in \mathbb{R}^p$ is the target vector, which contains the desired output (target) for each pattern; *iii*) $w \in \mathbb{R}^d$ is the vector of weights, which characterizes the linear model; *iv*) p is the number of training patterns; and *v*) d is the dimension of the patterns. Once the model is trained, the output for a new pattern x is estimated using the optimum weights w^{o} as $\tilde{y} = x \cdot w^{\mathrm{o}}$.

A classical choice for the regularizer $f_{\mathrm{R}}(w)$ is the Tikhonov regularization, which is just the squared ℓ_2 norm of the weights, $f_{\mathrm{R}}(w) = \|w\|_2^2$. Although this term prevents overfitting, it does not impose any structure in the resultant weights, thus not providing any additional information about the problem. Another alternative approach is to use the ℓ_1 norm, $f_{\mathrm{R}}(w) = \|w\|_1$, which results in the Lasso model [3]. The advantage of this regularizer is that it imposes sparsity on the solution, which means that some of the coefficients will be identically zero, hence giving an implicit feature selection (only those features with a non-zero coefficient are considered in order to estimate the output). As described in Section III, there are several models based on the ℓ_1 norm or the $\ell_{2,1}$ one for group features that impose some desired structure on the resultant weights. Nevertheless, the main problem of such models is that the ℓ_1 penalization is non-differentiable, which prevents the corresponding optimization problems from being solved through standard gradient-based techniques. A possible alternative, Proximal Methods, is briefly described next.

B. Proximal Methods

The models based on an ℓ_1 or $\ell_{2,1}$ penalization lead to a convex but non-differentiable optimization problem. Although these problems cannot be solved using gradient-based approaches, they fit naturally under the framework of Proximal Methods (PMs; see e.g. [4]), which are a set of techniques to optimize convex but possibly non-smooth functions through the splitting of the objective in several somehow "easier" terms. These terms are then minimized independently via its Proximal Operator (POp), defined below.

For instance, in Prob. (1) the function to be minimized is $f_{\mathrm{L}}(w) + f_{\mathrm{R}}(w)$, where w are assumed to be the parameters of the model and the penalty factor λ is included in $f_{\mathrm{R}}(w)$ for the sake of notation. As shown next, the intuitive idea of the PMs is to minimize this sum through the iterative minimization of f_{L} (using a gradient descent step, as this term is smooth) and f_{R} (using its POp, as this term will be in general non-smooth).

Let $\partial h(w)$ be the subdifferential at w of a convex function h, i.e., the set of all the subgradients of h at w [5]; as both terms $f_{\mathrm{L}}(w)$ and $f_{\mathrm{R}}(w)$ are convex, w^{o} will be a minimum of $f_{\mathrm{L}}(w)+f_{\mathrm{R}}(w)$ if and only if zero belongs to the subdifferential of the sum [6]:

$$w^{\mathrm{o}} = \arg\min_{w\in\mathbb{R}^d} \{f_{\mathrm{L}}(w) + f_{\mathrm{R}}(w)\} \iff 0 \in \partial(f_{\mathrm{L}}(w^{\mathrm{o}}) + f_{\mathrm{R}}(w^{\mathrm{o}})) \ . \quad (2)$$

Moreover, the right-hand side subdifferential can be separated as $\partial f_{\mathrm{L}}(w^{\mathrm{o}})+\partial f_{\mathrm{R}}(w^{\mathrm{o}})$ by the Moreau–Rockafellar [7] theorem, and since $f_{\mathrm{L}}(w)$ is differentiable, the optimality condition can be rewritten in the following equivalent expressions:

$$\begin{aligned} 0 &\in \partial f_{\mathrm{L}}(w^{\mathrm{o}}) + \partial f_{\mathrm{R}}(w^{\mathrm{o}}) = \nabla f_{\mathrm{L}}(w^{\mathrm{o}}) + \partial f_{\mathrm{R}}(w^{\mathrm{o}}) \\ &\iff -\gamma\nabla f_{\mathrm{L}}(w^{\mathrm{o}}) \in \gamma\lambda\partial f_{\mathrm{R}}(w^{\mathrm{o}}) \\ &\iff w^{\mathrm{o}} - \gamma\nabla f_{\mathrm{L}}(w^{\mathrm{o}}) \in (I + \gamma\partial f_{\mathrm{R}})(w^{\mathrm{o}}) \ , \end{aligned}$$

which are satisfied for any $\gamma > 0$. Furthermore, for a general convex function F, ∂F is a monotone operator [6] and, while in principle $(I + \partial F)^{-1}$ would be just a set-valued function, it is actually single-valued. Thus, at an optimal w^{o} we have

$$w^{\mathrm{o}} = (I + \gamma\partial f_{\mathrm{R}})^{-1}(w^{\mathrm{o}} - \gamma\nabla f_{\mathrm{L}}(w^{\mathrm{o}})) \ . \quad (3)$$

On the other hand, if F is convex and lower semicontinuous, its Proximal Operator at w with step $\gamma > 0$ is defined as

$$z_w = \mathrm{prox}_{\gamma;F}(w) = \arg\min_{z\in\mathbb{R}^d} \left\{ \frac{1}{2}\|z - w\|_2^2 + \gamma F(z) \right\} . \quad (4)$$

Notice that the POp can also be characterized in terms of the subdifferential using the optimality condition (2):

$$\begin{aligned} z_w = \mathrm{prox}_{\gamma;F}(w) &\iff 0 \in z_w - w + \gamma\partial F(z_w) \\ &\iff z_w \in (I + \partial F)^{-1}(w) \\ &\iff z_w = (I + \partial F)^{-1}(w) \ . \end{aligned}$$

In other words, we have an equivalent definition of the POp,

$$\mathrm{prox}_{\gamma;F}(w) = (I + \partial F)^{-1}(w) \ ,$$

that justifies the iterative algorithms described next. In any case, we remark that the POp definition through Prob. (4) is still crucial as it provides a general way to compute the POps for non-trivial functions. Now, going back to (3), we have

$$w^{\mathrm{o}} = \mathrm{prox}_{\gamma;f_{\mathrm{R}}}(w^{\mathrm{o}} - \gamma\nabla f_{\mathrm{L}}(w^{\mathrm{o}})) \ ,$$

which immediately suggests an iterative algorithm of the form

$$w^{k+1} = \mathrm{prox}_{\gamma;f_{\mathrm{R}}}\left(w^k - \gamma\nabla f_{\mathrm{L}}(w^k)\right) \ .$$

The w^k converge because the POp of a convex function is firmly non-expansive [5]. This is at the heart of the well known proximal gradient method (PGM; [4]) and of its ISTA and FISTA (Fast Iterative Shrinkage–Thresholding Algorithm) extensions [8]. In particular, FISTA iterates the pair of equations:

$$\begin{aligned} w^k &= \mathrm{prox}_{\frac{1}{L};f_{\mathrm{R}}}\left(z^k - \frac{1}{L}\nabla f_{\mathrm{L}}(z^k)\right) \ , \\ z^{k+1} &= w^k + \frac{t^k - 1}{t^{k+1}}(w^k - w^{k-1}) \ , \\ &\text{where } t^{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right) \end{aligned}$$

and L is a Lipschitz constant for ∇f_{L}. The main advantage of FISTA, that we will use in our experiments, is its convergence rate, $O\big(1/k^2\big)$, in contrast with the $O(1/k)$ sublinear convergence of ISTA and PGM [8].

The characterization of the POp as the solution of Prob. (4) implies that if there is a non-trivial function f_{R} in Prob. (1), a new optimization problem has to be solved in each FISTA iteration. This introduces a double loop, leading to a general procedure of the form

```
FISTA loop:
| ...
| Proximal loop:
| | ...
| ...
```

and larger computational costs. Moreover, computing directly the POp of f_{R} may often not be possible and we have to use tools such as the Proximal Dykstra (PD) which imply further costs. We will consider such a situation in Section III-E.

III. A Group Lasso Generalization

A. Generalized Lasso

The Generalized Lasso model introduced by R. Tibshirani (GenLA; [1]) builds a linear model solving the following optimization problem:

$$\min_{w\in\mathbb{R}^d} \left\{ \frac{1}{2} \left\| Xw - y \right\|_2^2 + \lambda \left\| Dw \right\|_1 \right\} , \tag{5}$$

where D is an appropriate transformation matrix. Notice that when X is the identity matrix, the solution of Prob. (5) is just the POp of the regularizer $\|Dw\|_1$ with step λ. Moreover, and as detailed below, Prob. (5) can be straightforwardly solved for a general matrix X through PMs (for example, FISTA), once the cited POp can be computed. Depending on the transformation matrix D used, other linear models are recovered as particular cases, namely:

1) If D is the identify matrix $I \in \mathbb{R}^{d\times d}$, Prob. (5) is the problem that defines the Lasso (LA) linear model.
2) If D is the differencing matrix

$$D = D_{\mathrm{TV_{1d}}} = \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(d-1)\times d} , \tag{6}$$

then Prob. (5) corresponds to the Fused Lasso (FL) model, where the Total Variation (TV) regularization is used. This model enforces the coefficients to be piece-wise constant, as some of the differences, penalized by the ℓ_1 norm, will be identically zero.
3) Cases 1 and 2 can be combined by mixing together both transformation matrices as:

$$D = \begin{pmatrix} D_{\mathrm{TV_{1d}}} \\ I \end{pmatrix} \in \mathbb{R}^{(2d-1)\times d} . \tag{7}$$

This transformation leads to the sparse Fused Lasso model (sFL), which is defined using both the TV term and the ℓ_1 regularizer. Different regularization parameters can be applied to each one of the terms by multiplying one of the two submatrices in (7) by a constant.
4) Case 2 can be extended to several dimensions, using transformation matrices based on the graph connectivity of a multidimensional grid. The structure of the transformation matrix $D = D_{\mathrm{TV_{2d}}}$ corresponding to a 2-dimensional TV regularizer is described in detail in Section III-E. Obviously, an additional ℓ_1 term can be added to compose a sparse multidimensional FL model by concatenating an identity matrix to the multidimensional TV transformation matrix, leading to a multidimensional sFL.
5) More generally, and following the same idea of Case 4, a Graph-Guided Fused Lasso (GraphFL) model can be defined [1] in which the variables are considered as vertices of a graph, and the difference between two adjacent variables is penalized according to the corresponding edge weight. As before, this regularizer can be represented using an appropriate transformation matrix: each edge will be represented by a row with all the entries equal to zero except the two elements corresponding to the vertices that the edge is joining; these two entries will be equal to plus and minus the weight of the edge. Therefore, the dimension of the transformation matrix D becomes $l \times d$, where l is the number of edges. Again, adding an identity matrix at the end of the transformation matrix leads to a sparse Graph-Guided Fused Lasso (sGraphFL). Notice that both the 1- and multi-dimensional FL and sFL models are particular instances of the GraphFL and sGraphFL, respectively.
6) Another set of interesting transformation matrices considered in [1] corresponds to the trend filtering case, which are just multiples of the differencing matrix $D_{\mathrm{TV_{1d}}}$ given in (6). While the standard differencing matrix $D_{\mathrm{TV_{1d}}}$ imposes piece-wise constancy (enforcing the difference between adjacent coefficients to be zero), $D^2_{\mathrm{TV_{1d}}}$ imposes piece-wise linearity (enforcing the difference between differences of adjacent elements to be zero), $D^3_{\mathrm{TV_{1d}}}$ imposes the signal to be piece-wise quadratic, and so on. Therefore, the POp of the regularizer $\left\| D^n_{\mathrm{TV_{1d}}} w \right\|_1$ can be used to approximate signals using polynomials of degree n that may span signal segments of varying lengths.

B. Solving the Generalized Lasso Problem

All the models described above are particular instances of the so called GenLA, and they can be trained by solving Prob. (5) for the different transformation matrices. As Prob. (5) consists in minimizing the sum of a smooth term (the error term) and a non-smooth one (given by the ℓ_1 norm of the transformed vector), it can be solved using FISTA, provided that the POp of the non-smooth term can be computed. This POp is the solution of the following problem:

$$\min_{w\in\mathbb{R}^d} \left\{ \frac{1}{2} \left\| w - y \right\|_2^2 + \lambda \left\| Dw \right\|_1 \right\} . \tag{8}$$

To solve Prob. (8) for a general matrix $D \in \mathbb{R}^{\bar{d}\times d}$, a dual formulation based on the Fenchel Conjugate of the terms involved can be used:

$$\min_{u\in\mathbb{R}^{\bar{d}}} \left\{ \frac{1}{2} \left\| D^\top u - w \right\|_2^2 \right\} \quad \text{s.t. } \left\| u \right\|_\infty \leq \lambda . \tag{9}$$

Problem (9) can be solved through methods such as Projected Gradient or Spectral Projected Gradient (SPG) as it is a quadratic optimization problem with box constraints. Moreover, for the particular matrix of Case 2 above and its variants of Case 6, the Projected Newton method is a suitable approach for solving Prob. (9), since the Hessian of the objective function can be computed in linear time thanks to its particular structure [9]. Once Prob. (9) is solved with minimizer u^{o}, the solution w^{o} of Prob. (8) is obtained through the equality $w^{\mathrm{o}} = y - D^\top u^{\mathrm{o}}$.

An advantage of using a single general transformation matrix is that we can work directly with regularizers that "mix" the w penalties and whose POps cannot be separated. The standard approach would then be to compute individually each POp and then mix them using PD, which adds an extra loop. For instance, if the proximal is the sum of two non-separable functions, this may require a three nested loop structure:

```
FISTA loop:
| ...
| Proximal Dykstra loop:
| | ...
| | First Proximal loop:
| | | ...
| | Second Proximal loop:
| | | ...
| | ...
| ...
```

possibly quite expensive computationally. On the other hand, the formulation of Prob. (5) is more flexible, allowing to work with a suitable matrix D that combines two regularizers into a single one whose POp can be computed through a single loop. A PD mixing is no longer required and we can revert to the two loop FISTA training of Section II-B.

C. Generalized Group Lasso

In some situations the features are grouped, i.e., the feature vector $x \in \mathbb{R}^d$ is composed by d_g groups, x_n, of d_v variables each, with $d = d_g d_v$. More specifically, x is structured as:

$$x = \left(x_{1,1}, \ldots, x_{1,d_v}, \ldots, x_{d_g,1}, \ldots, x_{d_g,d_v}\right)^\top \ .$$

In such a framework, the $\ell_{2,1}$ norm (the ℓ_1 norm of the ℓ_2 norms of the groups) is more appropriate than the ℓ_1 norm, as the former induces sparsity at group level, whereas the latter does not consider any group structure. In this situation models as the Group Lasso (GL) and the Group Fused Lasso (GFL; [2]) arise as extensions of the LA and FL models. Notice that the group structure in the patterns induces a similar structure on the weights w and as a generalization of the previous group models, we can extend Prob. (5) to a group setting and solve the problem

$$\min_{w \in \mathbb{R}^d} \left\{ \frac{1}{2} \left\| Xw - y \right\|_2^2 + \lambda \left\| \bar{D} w \right\|_{2,1} \right\} \tag{10}$$

for a certain group transformation matrix $\bar{D}$ that maps $\mathbb{R}^{d_g d_v}$ into $\mathbb{R}^{\bar{d}_g d_v}$. Thus, $\bar{D}w$ has a similar group structure now in $\mathbb{R}^{\bar{d}_g \times d_v}$ and we have

$$\left\| \bar{D} w \right\|_{2,1} = \sum_{n=1}^{\bar{d}_g} \left\| \left(\bar{D} w\right)_n \right\|_2 = \sum_{n=1}^{\bar{d}_g} \sqrt{\sum_{v=1}^{d_v} \left(\bar{D} w\right)_{n,v}^2} \ .$$

We call its solution the Generalized Group Lasso model (GenGL) and, indeed, all Cases 1 to 6 defined in Section III-A can be extended to this situation, defining the matrix $\bar{D}$ as $\bar{D} = D \otimes I$, where $I \in \mathbb{R}^{d_v \times d_v}$ is the identity matrix, $D \otimes I$ denotes Kronecker's product and D can be any of the transformation matrices introduced above. In fact:

1) The LA model of Case 1 leads to the GL model.
2) The FL models of Cases 2 to 4 result in the GFL models of [2], where the TV regularizer is replaced by the Group Total Variation (GTV) regularizer, and the sparsity-inducing ℓ_1 norm term is substituted by an $\ell_{2,1}$ norm regularizer.
3) The GraphFL and sGraphFL models of Case 5 result in a new Graph-Guided Group Fused Lasso (GraphGFL) and its sparse variant, sGraphGFL.
4) The trend filtering models of Case 6 result in a new group trend filtering.

D. Solving the Generalized Group Lasso

As it can be seen, Prob. (10) is quite close to Prob. (5) and we can follow exactly the same approach as in Section III-B using as before FISTA and computing at each iteration the POp of the right-hand term, which requires to solve:

$$\min_{w \in \mathbb{R}^d} \left\{ \frac{1}{2} \left\| w - y \right\|_2^2 + \lambda \left\| \bar{D} w \right\|_{2,1} \right\} \ . \tag{11}$$

In this case, the dual version of Prob. (11) becomes:

$$\min_{u \in \mathbb{R}^{\bar{d}}} \left\{ \frac{1}{2} \left\| \bar{D}^\top u - w \right\|_2^2 \right\} \quad \text{s.t. } \left\| u \right\|_{2,\infty} \leq \lambda \ . \tag{12}$$

The gradient of Prob. (12) can be easily computed as $\bar{D}\bar{D}^\top u - \bar{D} w$, and since the projection over the $\ell_{2,\infty}$ ball is just that of each group over the corresponding ℓ_2 balls, this problem can also be solved using SPG.

Notice that as long as data are group-structured along a single axis we can also avoid here using PD on some problems taking advantage of the possibility of combining two non-separable regularizers under an appropriate single $\bar{D}$ matrix. However there are important problems that have a natural 2- or higher-dimensional group structure as, for instance, those arising when GTV regularization is applied in image processing. We consider this situation next.

E. Group Structured Total Variation

The GFL problem is a particular case of Prob. (10) where the GTV penalty $\left\| \bar{D} w \right\|_{2,1}$ is used, with the transformation matrix $\bar{D}$ equal to the 1-dimensional group differencing matrix:

$$\bar{D}_{\text{GTV}_{\text{1d}}} = D_{\text{TV}_{\text{1d}}} \otimes I = \begin{pmatrix} -I & I & & \\ & \ddots & \ddots & \\ & & -I & I \end{pmatrix} \in \mathbb{R}^{(d_g - 1) d_v \times d_g d_v} \ ,$$

where I is the d_v-dimensional identity matrix. The groups of variables in the GFL problem are assumed to have a 1-axial group structure, but it is often of interest to consider groups structured along two or more axes. For instance, a 2-axial TV penalty along the horizontal and vertical image axes is often used in gray level image processing [9] and it leads to a 2-axis GTV penalty if RGB colour images are considered. Specifically, for a 2-dimensional $d_1 \times d_2$ signal (total number of groups is thus $d_g = d_1 d_2$) and denoting the i-th row of w by $w_{[i,\cdot]}$ and its j-th column by $w_{[\cdot,j]}$, the 2-axis GTV regularizer is defined as:

$$\text{GTV}_{\text{2d}}(w) = \sum_{i=1}^{d_1} \text{GTV}_{\text{1d}}\left(w_{[i,\cdot]}\right) + \sum_{j=1}^{d_2} \text{GTV}_{\text{1d}}\left(w_{[\cdot,j]}\right) \ , \tag{13}$$

i.e., the 2-axis GTV regularizer is the sum of horizontal and vertical 1-axis GTVs, and this can be easily extended to more than two axes. In order to compute the POp of the

2- and higher-axial GTV regularizers, notice that those terms corresponding to the same axis apply to different variables. They are therefore separable and the POp of the sum can be computed just by composing the individual POps. However, when added together, the axial GTV regularizers in (13) are no longer separable and, hence, would have to be combined, in principle using PD. Of course, this imposes an extra loop in the FISTA iterations. However, PD can be avoided working with an enlarged version of the transformation matrix, which we will denote by $\bar{D}_{\text{GTV}_{2\text{d}}}$, and that is described next.

We will consider first the case of an enlarged 2-dimensional TV differencing matrix $D_{\text{TV}_{2\text{d}}}$, which is designed to satisfy $\text{TV}_{2\text{d}}(w) = \|D_{\text{TV}_{2\text{d}}} w\|_1$, where

$$\text{TV}_{2\text{d}}(w) = \sum_{i=1}^{d_1} \text{TV}_{1\text{d}}\big(w_{[i,\cdot]}\big) + \sum_{j=1}^{d_2} \text{TV}_{1\text{d}}\big(w_{[\cdot,j]}\big) \ .$$

Once such a matrix is defined, the group variant is simply $\bar{D}_{\text{GTV}_{2\text{d}}} = D_{\text{TV}_{2\text{d}}} \otimes I$. We assume that the vector $w \in \mathbb{R}^{d_1 d_2}$ is structured as:

$$w = (w_{1,1}, \ldots, w_{1,d_2}, \ldots, w_{d_1,1}, \ldots, w_{d_1,d_2})^\top \ . \tag{14}$$

This means that the variable in the grid coordinates (i, j) is in the $id_2 + j$ position of the vector. This structure represents a 2-dimensional grid of scalar variables; the group variant of this structure is obtained by substituting each variable $w_{i,j}$ by the corresponding d_v-dimensional group feature.

Going back to the construction of $D_{\text{TV}_{2\text{d}}}$, notice that the number of connections in the grid will correspond to the number of rows in $D_{\text{TV}_{2\text{d}}}$, whereas the number of columns coincides with the total dimension of the problem. Each row of the grid has d_2 vertices, and thus $d_2 - 1$ connections, and there are d_1 rows. On the other side, there are d_2 columns with $d_1 - 1$ edges each. Therefore, the total number of edges is $2d_1d_2 - (d_1 + d_2) = 2d - (d_1 + d_2)$, i.e., about twice the dimension. This means that the matrix $D_{\text{TV}_{2\text{d}}}$ has dimensions $2d - (d_1 + d_2) \times d$, which seems prohibitive. Nevertheless, $D_{\text{TV}_{2\text{d}}}$ is highly sparse, as each row contains only two entries equal to 1 and -1. Thus, the total number of non-zero entries is just $2(2d - (d_1 + d_2))$, i.e., around $4d$, similar to the d input dimension. Now we describe how to build this matrix:

1) We start with the differencing matrix that penalizes discrepancies into a single row. As each row has d_2 elements, it is just the differencing matrix of a TV term of dimension d_2, namely $D_{\text{r}} = {D_{\text{TV}_{1\text{d}}}}^{(d_2)} \in \mathbb{R}^{(d_2-1)\times d_2}$, where the superscript denotes the dimension.
2) We repeat this to penalize the difference into a single column of d_1 elements, i.e., $D_{\text{c}} = {D_{\text{TV}_{1\text{d}}}}^{(d_1)} \in \mathbb{R}^{(d_1-1)\times d_1}$.
3) In order to penalize the differences into all the rows, the matrix D_{r} is repeated d_1 times, i.e., as many times as there are rows. Moreover, since the entries corresponding to the same row are consecutive, it suffices to extend the matrix as $D'_{\text{r}} = I^{(d_1)} \otimes D_{\text{r}}$.
4) The same has to be done for the column differences. Since two consecutive entries of the same column are separated by d_2 positions (because of the structure shown in (14)), the matrix can be expanded as $D'_{\text{c}} = D_{\text{c}} \otimes I^{(d_2)}$.
5) For mixing all the penalizations (over rows and columns) into a single matrix, we have just to concatenate the previous matrices, to get $D_{\text{TV}_{2\text{d}}} = (D'^{\top}_{\text{r}}, D'^{\top}_{\text{c}})^\top$.

```
function D = DifmatrixTVd1Fast(d)
    aux = sparse(diag(-ones(d-1,1),1));
    D = sparse(-[speye(d-1) sparse(d-1,1)]-aux(1:end-1,:));
end
function D = DifmatrixGTVd2Fast(d1, d2, nv)
    Dr = DifmatrixTVd1Fast(d2); % Step 1.
    Dc = DifmatrixTVd1Fast(d1); % Step 2.
    Drp = kron(speye(d1),Dr);   % Step 3.
    Dcp = kron(Dc,speye(d2));   % Step 4.
    D = [Drp ; Dcp];            % Step 5.
    D = kron(D,speye(nv));      % Step 6.
end
```

Listing 1. Matlab implementation of the construction of $\bar{D}_{\text{GTV}_{2\text{d}}}$ (second function), based on the construction of $D_{\text{TV}_{1\text{d}}}$ (first function).

6) Finally, the group variant is $\bar{D}_{\text{GTV}_{2\text{d}}} = D_{\text{TV}_{2\text{d}}} \otimes I$.

Therefore, we have a group differencing matrix $\bar{D}_{\text{GTV}_{2\text{d}}}$ that verifies $\text{GTV}_{2\text{d}}(w) = \big\|\bar{D}_{\text{GTV}_{2\text{d}}} w\big\|_{2,1}$, so we can solve the general Prob. (10) simply by combining a FISTA and a proximal loop without having to take recourse to PD. Of particular interest in image filtering is the case when X is just the identity matrix. Solving Prob. (10) reduces then to compute the POp of $\text{GTV}_{2\text{d}}$, what we can do now with a single loop.

An implementation (with sparse matrices) of the above algorithm is included in List. 1. As an example, the following matrix is the difference matrix $D_{\text{TV}_{2\text{d}}}$ on a 3×3 grid:

$$D_{\text{TV}_{2\text{d}}} = \begin{array}{c} \begin{matrix} {\scriptstyle(1,1)} & {\scriptstyle(1,2)} & {\scriptstyle(1,3)} & {\scriptstyle(2,1)} & {\scriptstyle(2,2)} & {\scriptstyle(2,3)} & {\scriptstyle(3,1)} & {\scriptstyle(3,2)} & {\scriptstyle(3,3)} \end{matrix} \\ \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \end{pmatrix} \\ \begin{matrix} {\scriptstyle(1,1)} & {\scriptstyle(1,2)} & {\scriptstyle(1,3)} & {\scriptstyle(2,1)} & {\scriptstyle(2,2)} & {\scriptstyle(2,3)} & {\scriptstyle(3,1)} & {\scriptstyle(3,2)} & {\scriptstyle(3,3)} \end{matrix} \end{array} .$$

IV. Experiments

In this section we will first illustrate the application of GenGL to capture a known group structure and that of GraphGFL to identify an underlying but unknown one. This will be done over two synthetic problems and then we will show the computational advantage of GenGL over GTV and other methods when applied to image denoising.

A. Synthetic Example: Identifying Weight Structure

In this experiment we will compare the efficiency of training a 1-dimensional sGFL model as the mixture of the $\ell_{2,1}$ and GTV regularizers through PD or using a GenGL set-up with the matrix of Case 3 (in its group variant).

Following the framework of [2], we define a synthetic problem whose underlying solution is structured. We begin by fixing the weights, which are clustered in 4 consecutive blocks of 25 groups each so that the weights of each block are equal in their 3 components; these piece-wise constant weights are then perturbed using Gaussian noise with a standard deviation of 0.1. We then generate the independent random input features of the problem as 100 groups of 3 variables; all follow a standard Gaussian. Finally, we multiply these features by the perturbed weights to obtain the targets which are, in turn, also perturbed

with Gaussian noise of deviation 0.1. We will generate this way 4 different samples with sizes p equal to 600, 300, 100 and 50 patterns. It is worth noting that since the number of free parameters is $d = 300$, the problems are ill-posed in the latter two cases. The linear models compared are RLS (Regularized Least Squares based on Tikhonov regularization), LA, GL, sFL, sGFL and GenGL (implementing sGFL). As a quality measure, the distance between the weights provided by each model and the true structured (unperturbed) weights is used, both using the ℓ_1 and the ℓ_2 norms. We repeat the experiment 100 times, varying the random patterns using the same distribution but keeping in all cases the same fixed structured weights. The regularization parameters are optimized with respect to the ℓ_1 distance over an initial, independent sample, and then kept fixed. The main goal of this experiment is to compare the time required to train the sGFL model using the classical and the GenGL approaches, and also to check if the different models are able to recover the underlying structured weights.[1]

Figure 1 summarizes the results. The first picture compares the time of sGFL and GenGL, clearly showing the latter to be faster, as it avoids using the PD algorithm required to mix the $\ell_{2,1}$ and GTV regularizers. The last two pictures show the ℓ_1 and ℓ_2 distances for the 4 sizes and all the models; obviously, GenGL obtains exactly the same solution (up to rounding effects) than sGFL and thus both lines overlap, and these models outperform all the others as further detailed in [2].

B. Synthetic Example: Identifying Structured Features

We will work now with an unstructured vector of random weights generated using a standard Gaussian. The patterns, however, will have random features organized in 8 groups g_n of 3 variables. These input features will be distributed in 3 blocks, $\{g_1, g_3, g_4\}$, $\{g_2, g_6\}$ and $\{g_5, g_7, g_8\}$. Inside each block the features will have an 80% correlation in the 3 components of each group, but they will be independent from the features of the other blocks. We will generate this way 1 000 patterns using a multivariate Gaussian and, finally, obtain the corresponding outputs by multiplying structured patterns and weights, and then adding some Gaussian noise with deviation 0.02.

The goal here is to check whether GenGL can identify the underlying blocks by assigning the same weights to all of their components when we solve a GraphGFL problem over a fully connected graph. We stress that the real relationship between the features is not known when building the model, which simply connects every pair of features, without any previous knowledge. The transformation matrix $\bar{D}$ will thus penalize the difference between the weights that connect any two groups of features. More precisely, we have $\bar{D} = D \otimes I$, with $I \in \mathbb{R}^{3\times 3}$ the identity matrix, and $D \in \mathbb{R}^{28\times 8}$ the differencing matrix between every pair of groups; it thus must have 28 rows, i.e., all the possible pairings of 8 elements.

The results are summarized in Fig. 2, which includes the regularization path of the weights obtained by the GraphGFL model. We depict the optimal weights for 50 different values of the regularization parameter λ between 10^0 and $10^{1.5}$ in logarithmic scale. The 8 colours represent the different groups

[1]This second objective is included for completeness, since such a comparison was already done in [2], where sGFL, which ties with GenGL, statistically outperformed the other models in the proposed task.

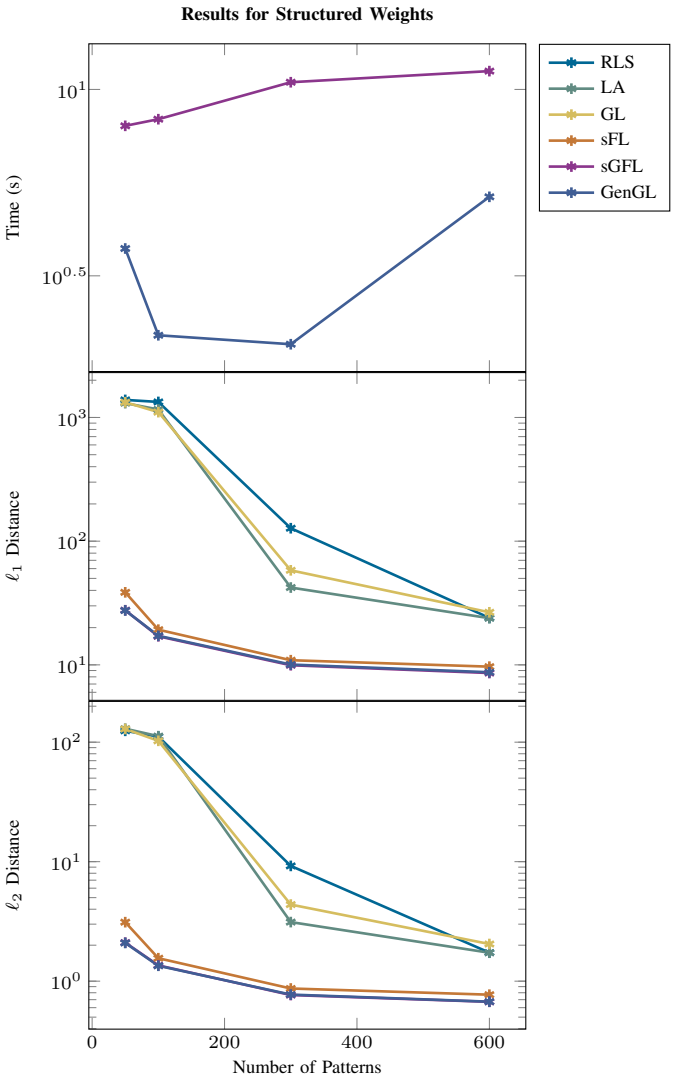
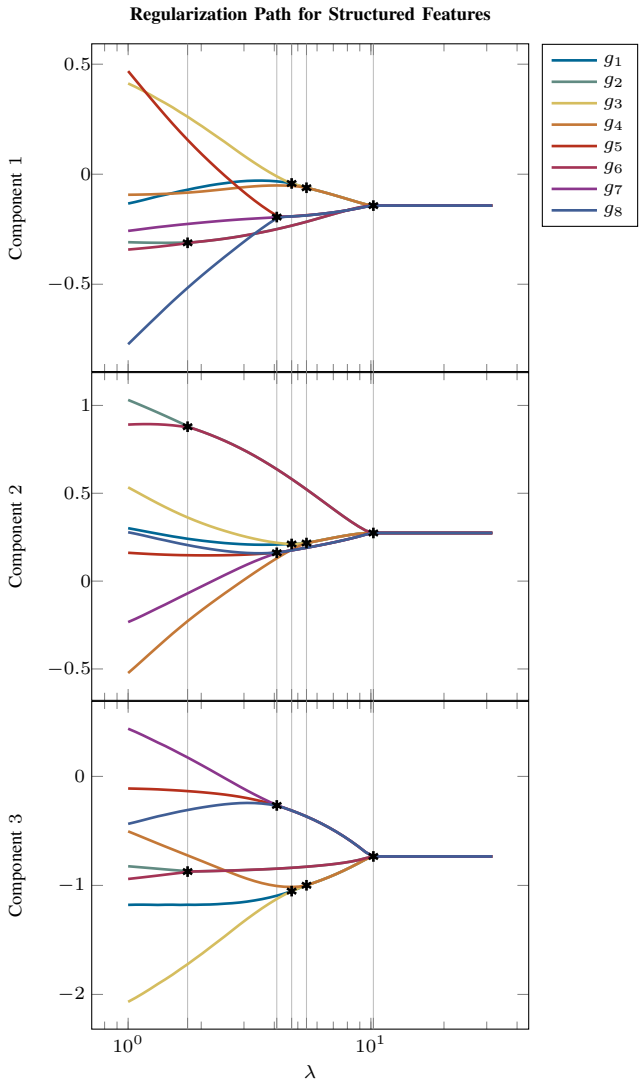


Fig. 1. Results for the structured weights synthetic example.

of features, and the 3 pictures correspond to each one of the 3 different group components of each feature. Moreover, the change points in which the groups "align", i.e., the weights take the same value for the 3 group components, are marked with an asterisk; they are also summarized at the bottom of Fig. 2, showing the successive blocks found. As expected, when λ is small the model prioritizes the training error, so the coefficients are all different and do not show any structure. Nevertheless, when λ increases the structure becomes stronger: the coefficients start to collapse and different blocks appear when their values align. In particular, for λ values between $10^{0.73}$ and $10^{1.01}$ the coefficients reveal the true underlying feature blocks. Finally, larger λ values force the differences between each pair to be identically zero and all the groups collapse in a single one.

C. Image Denoising

In this last example we will compare the efficiency of computing the POp of a 2-dimensional GTV regularizer combining the 1-dimensional GTV regularizer using PD, and using the POp of the GenGL regularizer of Case 4, in an image denoising



Change points:

$10^{0.00}$: {{g_1}{g_2}{g_3}{g_4}{g_5}{g_6}{g_7}{g_8}} ;
$10^{0.24}$: {{g_1}{g_2, g_6}{g_3}{g_4}{g_5}{g_7}{g_8}} ;
$10^{0.61}$: {{g_1}{g_2, g_6}{g_3}{g_4}{g_5, g_7, g_8}} ;
$10^{0.67}$: {{g_1, g_3}{g_2, g_6}{g_4}{g_5, g_7, g_8}} ;
$10^{0.73}$: {{g_1, g_3, g_4}{g_2, g_6}{g_5, g_7, g_8}} ;
$10^{1.01}$: {{$g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8$}} .

Fig. 2. Results for the structured features synthetic example. The three pictures show the regularization path for each component; the last expressions summarize the change points, and the clusters achieved at each of them.

problem. Although in terms of quality both approaches should be identical (as they are solving the same problem), in terms of efficiency the GTV approach needs to use PD to mix the POps of the GTV regularizers along rows and columns, each of which requires to solve a constrained problem using SPG. On the contrary, GenGL solves a unique constrained problem, although with many more variables, but as shown below, it outperforms the GTV approach for all image sizes considered.

In particular, the experiment considers denoising 5 images perturbed with different noises: *i*) *Peppers*, with additive Gaussian noise; *ii*) *House*, with speckle (multiplicative uniform) noise; *iii*) *Lena*, with Poisson noise; *iv*) *Mandrill*, with salt and pepper noise; and *v*) *Squares*, with both additive Gaussian and Poisson noises. We follow the framework of [2] using the POp of the 2-dimensional GTV regularizer that was shown to outperform a classical approach using a TV regularizer over each colour-channel independently. The regularization parameter λ for each image is selected over an initial noisy sample, and then the results are averaged over 25 perturbations with the same noise distribution. This is repeated for 4 sizes of the image, that corresponds to reductions to the 5%, 10%, 20% and 50% of the original size. We use the Improved Signal to Noise Ratio (ISNR; [10]) as the denoising quality measure; the higher its value, the cleaner the final image is.

Figure 3 shows a comparative of the execution time and the ISNR as a function of the number of variables (the number of pixels times the 3 colour channels). The different colours denote different images (and noises), whereas the type of the line represents the denoising method. As expected, in terms of the ISNR both GTV and GenGL obtain the same result (with small perturbations corresponding to the convergence of the algorithms), and both of them outperform clearly the classical TV denoising (a detailed comparison between TV and GTV denoising, including examples of the filtered images, can be found in [2]). On the other hand, when looking at execution times, GenGL is much faster than GTV. It is worth noting that both are implemented in Matlab, and the code is comparable as they use the same SPG implementation for solving the inner constrained problem; hence the advantage of GenGL comes from avoiding the PD algorithm. The time of TV is omitted as it was implemented in C. The training times required by GTV and GenGL are further compared in Fig. 4, which contains convergence plots for both methods. Execution times (averaged over 25 repetitions) are in the x-axis and in the y-axis normalized objective function values, i.e. the criterion value at the current solution minus that at the optimum, given as a percentage of the optimum criterion value. The oscillations of the GenGL approach come from the SPG algorithm used; the plot for GTV shows criterion values after each Dykstra iteration, which is why they appear to be smoother. In any case GenGL is about two orders of magnitude faster.

V. Discussion and Conclusions

In this paper, the Generalized Group Lasso (GenGL) framework for linear models is proposed as an extension to the case of multidimensional (group) features of the Generalized Lasso of [1]. In particular, the ℓ_1 norm is replaced by the $\ell_{2,1}$ norm. GenGL includes as particular cases the Group Lasso (GL) and Group Fused Lasso (GFL) and it also allows to define new models for multidimensional trend filtering or the Graph-Guided Group Fused Lasso. An additional advantage of this formulation is that the Proximal Operator (POp) of the regularizer of all the different models can be computed solving just one optimization problem, whereas in earlier approaches for the sparse GFL model (which includes a Group Total Variation term and a GL term), or for a multidimensional GFL model, two or more individual POps had to be computed and then combined using Proximal Dykstra in order to estimate the POp of the sum of all the regularizers. Consequently, the GenGL approach can be much faster in some problems.

On the other hand, second order methods for the GFL have been derived in [11]. It is also extremely interesting to try

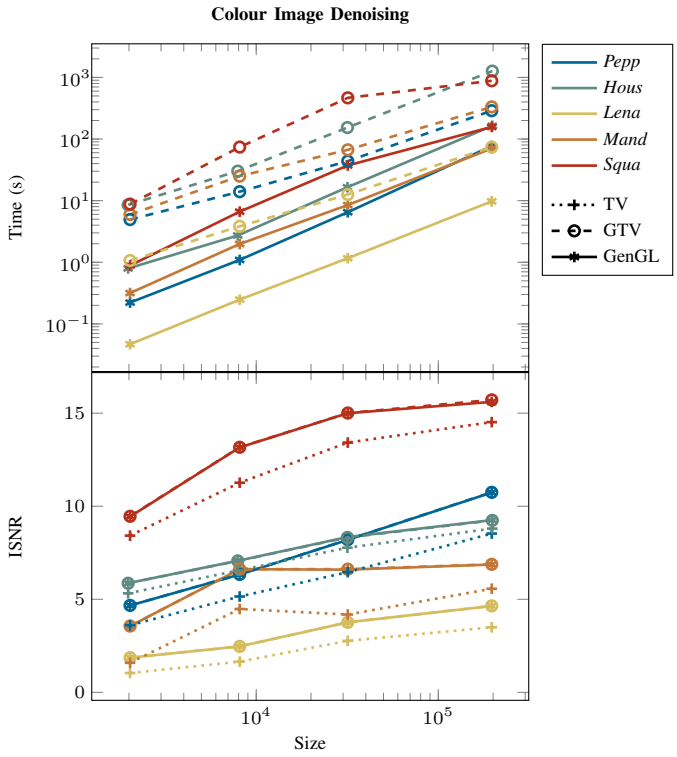


Fig. 3. Time and ISNR as a function of the size; the results are given for the different datasets (colours) and denoising methods (lines and markers).

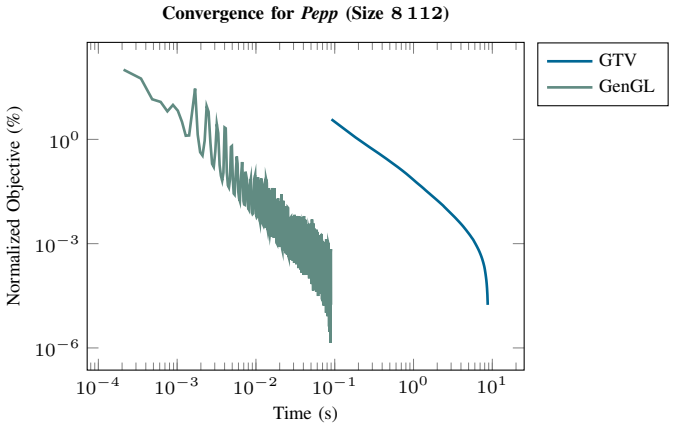


Fig. 4. Convergence of the normalized objective as a function of time, for the image *Pepp* with the second size.

to apply this Newton approach to the GenGL model, as the difference between the optimization problems of both models only resides in the transformation matrix, which will also be sparse in the GenGL model for most of the cases of interest. We point out that it is shown in [1] that in some cases the FISTA loop can be removed for GenLA through a variable change that reduces the general Prob. (5) just to the POp of the regularizer [1]. This can be applied to GenGL, although such a procedure will require working with non-sparse matrices. Hence, the gain of having a single loop may be negated by having to work with a full matrix of a large dimension. Nevertheless, this approach should also be analysed in detail.

As further work, this framework could be applied to real-world problems. An interesting case is the analysis and forecasting exploitation of Numerical Weather Prediction patterns that are given over a grid with several weather variables per grid point, having thus a natural 2-axis group structure. Another area of application is the trend filtering of multi-dimensional signals or of colour images, assuming instead of constancy between adjacent pixels, linear or higher order polynomial relations. Finally, all these models can reveal some inherent pattern structure in a given problem, which could then be exploited by building another model not on the original patterns but over the identified structure. This hierarchical framework is another possible extension of this work.

Acknowledgment

With partial support from Spain's grants TIN2013-42351-P and S2013/ICE-2845 CASI-CAM-CM, and of the UAM–ADIC Chair for Data Science and Machine Learning. The authors gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM.

References

[1] R. J. Tibshirani, *The solution path of the generalized lasso*. Stanford University, 2011.

[2] C. M. Alaíz, A. Barbero, and J. R. Dorronsoro, "Enforcing group structure through the group fused lasso," in *Artificial Neural Networks*, ser. Springer Series in Bio-/Neuroinformatics, P. Koprinkova-Hristova, V. Mladenov, and N. K. Kasabov, Eds. Heidelberg, Germany: Springer-Verlag GmbH, October 2014, vol. 4, pp. 349–371.

[3] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *J. Roy. Statist. Soc. Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.

[4] P. Combettes and J. Pesquet, "Proximal splitting methods in signal processing." Springer, 2011, pp. 185–212.

[5] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[6] H. Bauschke and P. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

[7] R. T. Rockafellar, *Convex Analysis (Princeton Landmarks in Mathematics and Physics)*. Princeton University Press, 1996.

[8] A. Beck and M. Teboulle, "A fast iterative shrinkage–thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[9] A. Barbero and S. Sra, "Fast newton–type methods for total variation regularization," in *Proceedings of the 28th International Conference on Machine Learning (ICML–11)*, New York, NY, USA, June 2011, pp. 313–320.

[10] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *Image Processing, IEEE Transactions on*, vol. 19, no. 9, pp. 2345–2356, September 2010.

[11] M. Wytock, S. Sra, and J. Z. Kolter, "Fast newton methods for the group fused lasso," in *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 2014.