Fast Newton methods for the group fused lasso

Matt Wytock
Machine Learning Dept.
Carnegie Mellon University
Pittsburgh, PA

Suvrit Sra
Machine Learning Dept.
Carnegie Mellon University
Pittsburgh, PA

J. Zico Kolter
Machine Learning Dept.
Carnegie Mellon University
Pittsburgh, PA

Abstract

We present a new algorithmic approach to the group fused lasso, a convex model that approximates a multi-dimensional signal via an approximately piecewise-constant signal. This model has found many applications in multiple change point detection, signal compression, and total variation denoising, though existing algorithms typically using first-order or alternating minimization schemes. In this paper we instead develop a specialized projected Newton method, combined with a primal active set approach, which we show to be substantially faster that existing methods. Furthermore, we present two applications that use this algorithm as a fast subroutine for a more complex outer loop: segmenting linear regression models for time series data, and color image denoising. We show that on these problems the proposed method performs very well, solving the problems faster than state-of-the-art methods and to higher accuracy.

1 Introduction

Given a multivariate signal $y_1, y_2, \ldots, y_T$, with $y_t \in \mathbb{R}^n$, the (weighted) group fused lasso (GFL) estimator (Bleakley and Vert, 2011; Alaíz et al., 2013) attempts to find a roughly "piecewise-constant" approximation to this signal. It determines this approximation by solving the optimization problem

$$\underset{x_1,x_2,\ldots,x_T}{\text{minimize}} \quad \frac{1}{2}\sum_{t=1}^{T} w_i\|x_t - y_t\|_2^2 + \sum_{t=1}^{T-1} \lambda_t \|x_t - x_{t+1}\|_2 \quad (1)$$

where $x_1, x_2, \ldots, x_T$ are the optimization variables, $w \in \mathbb{R}_+^T$ are weights for each time point, $\lambda \in \mathbb{R}_+^{T-1}$ are regularization parameters, and $\|\cdot\|_2$ denotes the Euclidean norm. Intuitively, the ℓ_2 norm on the *difference* between consecutive points encourages sparsity in this difference: each difference $x_t - x_{t+1}$ will typically be either full or identically zero at the solution, i.e., the signal x will be approximately piecewise-constant. This approach generalizes the 1D total variation norm (Tibshirani et al., 2005; Barbero and Sra, 2011), which considers only univariate signals. Owing to the piecewise-constant nature of the approximate signals formed by the group fused lasso, the approach has found applications in signal compression, multiple change-point detection, and total variation denoising. Though several algorithms have been proposed to solve (1), to the best of our knowledge these have involved, at their foundation, first-order methods such as projected gradient, block coordinate descent, or splitting methods. Although such algorithms can sometimes obtain reasonable performance, they often fail to quickly find accurate solutions, especially when one wants to solve (1) to high precision as a "subroutine" (or prox-operator) in a larger algorithm (Barbero and Sra, 2011).

In this paper, we develop a fast algorithm for solving the optimization problem (1), based upon a projected Newton approach. Our method can solve group fused lasso problems to high numerical precision, often several orders of magnitude faster than existing state-of-the-art approaches. At its heart, our method involves dualizing the optimization problem (1) *twice*, in a particular manner, to eliminate the non-differentiable ℓ_2 norm and replace it by simple nonnegativity constraints; we solve the reformulated problem to high accuracy via a projected Newton approach. In order to fully exploit the sparsity of large-scale instances, we combine the above ideas with a primal active-set method that iteratively solves reduced-size problems to find the final set of non-zero differences for the original GFL problem.

Although our fast fused group lasso method is valuable in its own right, its real power comes when used as a proximal subroutine in a more complex algorithm, an operation that often needs to be solved thousands of times. With this motivation in mind, we apply our approach to two applications: segmenting linear regression models, and color total variation image denoising.

We demonstrate the power of our approach in experiments with real and synthetic data, both for the basic group fused lasso and these applications, and show substantial improvement over the state of the art.

2 A fast Newton method for the GFL

We begin by adopting slightly more compact notation, and rewrite (1) (the *primal* problem) as

$$\underset{X}{\text{minimize}} \quad \tfrac{1}{2}\|(X-Y)W^{1/2}\|_F^2 + \|XD\Lambda\|_{1,2} \qquad \text{(P)}$$

where $X, Y \in \mathbb{R}^{n\times T}$ denote the matrices

$$X = \left[\begin{array}{ccc} x_1 & \cdots & x_T \end{array}\right], \quad Y = \left[\begin{array}{ccc} y_1 & \cdots & y_T \end{array}\right]; \quad (2)$$

$W := \operatorname{diag}(w)$ and $\Lambda := \operatorname{diag}(\lambda)$; $\|\cdot\|_F$ denotes the Frobenius norm; $\|\cdot\|_{1,2}$ denotes the mixed $\ell_{1,2}$-norm

$$\|A\|_{1,2} := \sum\nolimits_i \|a_i\|_2, \qquad (3)$$

where a_i is the ith column of A; and $D \in \mathbb{R}^{T,T-1}$ denotes the first order differencing operator

$$D = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ -1 & 1 & 0 & \cdots \\ 0 & -1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \qquad (4)$$

so that XD takes the difference of the columns of X.

2.1 Dual problems

To solve (P), it is useful to look at its dual and (for our algorithm) a modified dual of this dual. To derive these problems, we transform (P) slightly by introducing the constraint $V = XD$, and corresponding dual variables $U \in \mathbb{R}^{n\times T-1}$. The Lagrangian is then given by

$$\begin{aligned}\mathcal{L}_P(X,U,V) := \tfrac{1}{2}\|(X-Y)W^{1/2}\|_F^2 + \|V\Lambda\|_{1,2} \\ + \operatorname{tr} U^T(V - XD).\end{aligned} \qquad (5)$$

Minimizing (5) analytically over X and V gives

$$X^\star = Y - UD^TW^{-1}, \;\; V^\star = 0 \text{ iff } \|u_t\|_2 \le \lambda_t \qquad (6)$$

where u_t is the t-th column of U; this leads to the dual

$$\begin{aligned}\underset{U}{\text{maximize}} \quad & -\tfrac{1}{2}\|UD^TW^{-1/2}\|_F^2 + \operatorname{tr} UD^TY^T \\ \text{subject to} \quad & \|u_t\|_2 \le \lambda_t, \qquad t = 1,\ldots,T-1.\end{aligned} \qquad \text{(D)}$$

Indeed, several past algorithmic approaches have solved (D) directly using projected gradient methods, see e.g., (Alaíz et al., 2013).

The basis of our algorithm is to form the dual of (D), but in a manner that leads to a different problem than the original primal. In particular, noting that the constraint $\|u_t\|_2 \le \lambda_t$ is equivalent to the constraint that $\|u_t\|_2^2 \le \lambda_t^2$, we can remove the non-differentiable ℓ_2 norm, and form the Lagrangian

$$\begin{aligned}\mathcal{L}_D(U,z) = -\tfrac{1}{2}\|UD^TW^{-1/2}\|_F^2 + \operatorname{tr} UD^TY^T \\ + \sum\nolimits_{t=1}^{T-1} z_t(\|u_t\|_2^2 - \lambda_t^2).\end{aligned} \qquad (7)$$

Minimizing over U analytically yields

$$U^\star = YD(D^TW^{-1}D + Z)^{-1}, \qquad (8)$$

where $Z := \operatorname{diag}(z)$, and leads to the dual problem (the dual of the dual of (P))

$$\min_{z\ge 0} \quad \tfrac{1}{2}YD(D^TW^{-1}D + Z)^{-1}D^TY^T + \tfrac{1}{2}(\lambda^2)^Tz, \qquad \text{(DD)}$$

where λ^2 denotes squaring λ elementwise. This procedure, taking the dual of the dual of the original optimization problem, has transformed the original, non-smooth problem into a smooth optimization problem subject to a non-negativity constraint, a setting for which there are several efficient algorithms. Although (DD) is not easily solved via a standard form semidefinite program—it involves a matrix fractional term, for which the standard semidefinite programming form is computationally unattractive—it can be solved efficiently by a number of methods for smooth, bound-constrained optimization. However, as we will see below, the Hessian for this problem is typically poorly conditioned, so the choice of algorithm for minimizing (DD) has a large impact in practice. Furthermore, because the z dual variables are non-zero only for the change points of the original X variables, we expect that for many regimes we will have very few non-zero z values. These points motivate the use of projected Newton methods (Bertsekas, 1982), which perform Newton updates on the variables not bound ($z \neq 0$).

2.2 A projected Newton method for (DD)

Denote the objective of (DD) as $f(z)$; the gradient and Hessian of f are given by

$$\begin{aligned}\nabla_z f(z) &= -\tfrac{1}{2}(U^2)^T1 + \tfrac{1}{2}\lambda^2, \\ \nabla_z^2 f(z) &= U^TU \circ (D^TW^{-1}D + Z)^{-1},\end{aligned} \qquad (9)$$

where as above $U = YD(D^TW^{-1}D + Z)^{-1}$, U^2 denotes elementwise squaring of U, and $\circ$ denotes the elementwise (Hadamard) product. The projected Newton method proceeds as follows: at each iteration, we construct the set of *bound* variables

$$\mathcal{I} := \{i : z_i = 0 \text{ and } (\nabla_z f(z))_i > 0\}. \qquad (10)$$

We then perform a Newton update only on those variables that are *not* bound ($\bar{\mathcal{I}}$, referred to as the *free set*), and project back onto the feasible set

$$z_{\bar{\mathcal{I}}} \leftarrow \left[z_{\bar{\mathcal{I}}} - \alpha(\nabla_z^2 f(z))_{\bar{\mathcal{I}},\bar{\mathcal{I}}}^{-1}(\nabla_z f(z))_{\bar{\mathcal{I}}}\right]_+, \qquad (11)$$

Algorithm 1 Projected Newton for GFL

input signal $Y \in \mathbb{R}^{n \times T}$; weights $w \in \mathbb{R}_+^T$; regularization parameters $\lambda \in \mathbb{R}_+^{T-1}$; tolerance ϵ
output: optimized signal $X \in \mathbb{R}^{n \times T}$
initialization: $z \leftarrow 0$
repeat

1. Form dual variables and gradient

$$U \leftarrow YD(DW^{-1}D + Z)^{-1}$$
$$\nabla_z f(z) \leftarrow -\tfrac{1}{2}(U^2)^T 1 + \tfrac{1}{2}\lambda^2$$

2. Compute active constraints

$$\mathcal{I} \leftarrow \{i : z_i = 0 \text{ and } (\nabla_z f(z))_i > 0\}$$

3. Compute reduced Hessian and Newton direction

$$H \leftarrow U_{\bar{\mathcal{I}}}^T U_{\bar{\mathcal{I}}} \circ (D^T W^{-1} D + Z)^{-1}_{\bar{\mathcal{I}},\bar{\mathcal{I}}}$$
$$\Delta z_{\bar{\mathcal{I}}} \leftarrow -H^{-1}(\nabla_z f(z))_{\bar{\mathcal{I}}}$$

4. Update variables

$$z_{\bar{\mathcal{I}}} \leftarrow [z_{\bar{\mathcal{I}}} + \alpha \Delta z_{\bar{\mathcal{I}}}]_+$$

where α is chosen by line search
until $\|(\nabla_z f(z))_{\bar{\mathcal{I}}}\|_2 \leq \epsilon$

where α is a step size (chosen by backtracking, interpolation, or other line search), and $[\cdot]_+$ denotes projection onto the non-negative orthant. The full method is shown in Algorithm 1. Although the projected Newton method is conceptually simple, it involves inverting several (possibly $T \times T$ matrices), which is impractical if these were to be computed as general matrix operations. Fortunately, there is a great amount of structure that can be exploited in this problem.

Efficiently solving $YD(D^TW^{-1}D+Z)^{-1}$. One key operation for the weighted GFL problem is to solve linear systems of the form $D^TW^{-1}D + Z$, where W and Z are diagonal. Fortunately, the first matrix is highly structured: it is a symmetric tridiagonal matrix

$$D^TW^{-1}D = \begin{bmatrix} \frac{1}{w_1}+\frac{1}{w_2} & -\frac{1}{w_2} & 0 & \cdots \\ -\frac{1}{w_2} & \frac{1}{w_2}+\frac{1}{w_3} & -\frac{1}{w_3} & \cdots \\ 0 & -\frac{1}{w_3} & \frac{1}{w_3}+\frac{1}{w_4} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{12}$$

and adding Z to it only affects the diagonal. LAPACK has customized routines for solving problems of this form: `dpttrf` (which computes the LDL^T factorization of the matrix) and `dptts2` (which computes the solution to $LDL^TX = B$ via backsubstitution). For our work, we modified this latter code slightly to solve systems with the unknown on the left hand size, as is required for our setting; this lends a slight speedup by exploiting the memory locality of column-based matrices. The methods factor $T-1 \times T-1$ matrix in $O(T)$ time, and solve n left hand sides in time $O(Tn)$.

Computing entries of $(D^TW^{-1}D+Z)^{-1}$. The projected Newton method also requires more than just solving equations of the form above: to compute the Hessian, we must actually also compute entries of the inverse $(D^TW^{-1}D + Z)^{-1}$ — we need to compute the entries with rows and columns in $\bar{\mathcal{I}}$. Naively, this would require solving $k = |\bar{\mathcal{I}}|$ left hand sides, corresponding to the unit bases for the entries in $\bar{\mathcal{I}}$; even using the fast solver above, this takes time $O(Tk)$. To speed up this operation, we instead use a fast method for computing the actual entries of the inverse of this tridiagonal, using an approach based upon (Usmani, 1994); this ultimately lets us compute the k^2 entries in $O(k^2)$ time, which can be much faster for small free sets.

Specifically, let $a \in \mathbb{R}^{T-1}$ and $b \in \mathbb{R}^{T-2}$ denote the diagonal and the negative off-diagonal entries of $D^TW^{-1}D+Z$ respectively (that is, $a_i = \frac{1}{w_i} + \frac{1}{w_{i+1}} + z_i$ and $b_i = \frac{1}{w_{i+1}}$), we can compute individual entries of $(D^TW^{-1}D+Z)^{-1}$ as follows (the following adapts the algorithm in (Usmani, 1994), but has enough simplifications for our case that we state it explicitly here). Define $\theta, \phi \in \mathbb{R}^T$ via the recursions

$$\begin{aligned} \theta_{i+1} &= a_i\theta_i - b_{i-1}^2\theta_{i-1},\ i = 2,\ldots,T-1 \\ \theta_1 &= 1,\ \theta_2 = a_1, \\ \phi_i &= a_i\phi_{i+1} - b_i^2\phi_{i+2},\ i = T-2,\ldots,1 \\ \phi_T &= 1,\ \phi_{T-1} = a_{T-1}. \end{aligned} \tag{13}$$

Then, the (i,j) entry of $(D^TW^{-1}D + Z)^{-1}$ for $j \leq i$ is given by

$$(D^TW^{-1}D + Z)^{-1}_{ij} = \frac{1}{\theta_T}\Big(\prod_{k=i}^{j-1} b_i\Big)\theta_i\phi_{j+1}. \tag{14}$$

Finally, we can compute all the needed running products $\prod_{k=i}^{j-1} b_i$ by computing a single cumulative sum of the logs of the b_i terms $c_i = \sum_{j=1}^{i} \log b_i$ and then using the equality $\prod_{k=i}^{j-1} b_i = \exp(c_j - c_i)$.

2.3 A primal active set approach

Using the two optimizations mentioned above, the projected Newton method can very quickly find a solution accurate to numerical precision for medium sized problems (T and n on the order of thousands). However, for problems with substantially larger T, which are precisely those we are most interested in for many GFL applications, the approach above begins to break down. There are two reasons for this: 1) The size of the free

set $k = |\mathcal{I}|$, though often small at the final solution, can be significantly larger at intermediate iterations; since the Newton method ultimately does involve an $O(k^3)$ time to invert the Hessian restricted to the free set, this can quickly render the algorithm impractical. 2) Even with small free sets, the basic $O(Tn)$ cost required for a single pass over the data at each Newton iteration starts to dominate, especially since a significant number of iterations to find the correct free set may be required (only after finding the correct free set does one obtain quadratic convergence rates).

To overcome these problems, we consider a further layer to the algorithm, which wraps our fast projected Newton solver inside a primal active-set method. The basic intuition is that, at the optimal solution to the original GFL problem, there will typically be very few change points in the solution $X^\star$ (these correspond exactly to those z variables that are non-zero). If we knew these changes points ahead of time, we could solve a substantially reduced (weighted) GFL problem that was equivalent to the original problem. Specifically, let $\mathcal{J} \subseteq \{1, \ldots, T-1\}$ denote the optimal set of change point locations for the primal problem. By the relationship of dual problems, this will be identical to the set of free variables $\bar{\mathcal{I}}$ at the optimal solution, but since we treat these differently in the algorithmic design we use different notation. Then the original problem

$$\underset{X}{\text{minimize}} \ \|(X-Y)W^{1/2}\|_F^2 + \|XD\Lambda\|_{1,2}, \qquad (15)$$

where $X \in \mathbb{R}^{n \times T}$, is equivalent to the reduced problem

$$\underset{X'}{\text{minimize}} \ \|(X'-Y')W'^{1/2}\|_F^2 + \|X'D'\Lambda_{\mathcal{J},\mathcal{J}}\|_{1,2} \quad (16)$$

with optimization variable $X' \in \mathbb{R}^{n \times k+1}$ for $k = |\mathcal{J}|$, where $D' \in \mathbb{R}^{k+1 \times k}$ denotes the same first order differences matrix but now over only $k+1$-sized vectors, and where Y' and $W' = \text{diag}(w')$ are defined by

$$w'_i = \sum_{j \in \mathcal{J}'_i} w_j, \quad y'_i = \frac{1}{w'_i} \sum_{j \in \mathcal{J}'_i} w_j y_j, \qquad (17)$$

where we define $\mathcal{J}'_i = \{\mathcal{J}_{i-1} + 1, \ldots, \mathcal{J}_i\}$ for $i = 1, \ldots, k+1$ (i.e., $\mathcal{J}'_i$ denotes the list of indices within the ith segment, there being $k+1$ segments for k change points). Furthermore, all these terms can be computed in time $O(nk)$ via cumulative sums similar to the cumulative sum used for b above (which take $O(Tn)$ to compute once, but which thereafter only require $O(kn)$ to form the reduced problem).

To see this equivalence, note first that since X only changes at the points $|\mathcal{J}|$, it immediately holds that $\|XD\Lambda\|_{1,2} = \|X'D'\Lambda_{\mathcal{J},\mathcal{J}}\|_{1,2}$. To show that the other term in the objective is also equivalent, we have that

$$\begin{aligned}
&\|(X-Y)W^{1/2}\|_F^2 \\
&= \sum_{i=1}^{k+1} \|(x'_i 1^T - Y_{\mathcal{J}'_i}) W^{1/2}_{\mathcal{J}'_i,\mathcal{J}'_i}\|_F^2 \\
&= \sum_{i=1}^{k+1} \left((w_{\mathcal{J}'}^T 1) x'^T_i x'_i - 2x'^T Y_{\mathcal{J}'_i} w_{\mathcal{J}'_i} + \|Y_{\mathcal{J}'_i} W^{1/2}_{\mathcal{J}'_i,\mathcal{J}'_i}\|_F^2 \right) \\
&= \sum_{i=1}^{k+1} w'_i \|x'_i - y'_i\|_2^2 + c.
\end{aligned}$$

This equivalence motivates a primal active set method where we iteratively guess the active set $\mathcal{J}$ (with some fixed limit on its allowable size), use the projected Newton algorithm to solve the reduced problem, and then use the updated solution to re-estimate the active set. This is essentially equivalent to a common "block pivoting" strategy for non-negative least squares (Portugal et al., 1994) or ℓ_1 methods (Lee et al., 2007), and has been shown to be very efficient in practice (Kim and Park, 2010). The full algorithm, which we refer to as Active Set Projected Newton (ASPN, pronounced "aspen"), is shown in Algorithm 2. In total, the algorithm is extremely competitive compared to past approaches to GFL, as we show in Section 4, often outperforming the existing state of the art by orders of magnitude.

3 Applications

Although the ASPN algorithm for the group fused lasso is a useful algorithm in its own right, part of the appeal of a fast solver for this type of problem is the possibility of using it as a "subroutine" within solvers for more complex problems. In this section we derive such algorithms for two instances: segmentation of time-varying linear regression models and multi-channel total variance image denoising. Both models have been considered in the literature previously, and the method presented here offers a way of solving these optimization problems to a relatively high degree of accuracy using simple methods.

3.1 Linear model segmentation

In this setting, we observe a sequence of input/output pairs ($a_t \in \mathbb{R}^n$, $y_t \in \mathbb{R}$) over time and the goal is to find model parameters x_t such that $y_t \approx a_t^T x_t$ (it is more common to denote the input itself as x_t and model parameters θ_t, but the notation here is more in keeping with the rest of this paper). Naturally, if x_t is allowed to vary arbitrarily, we can always find (an infinite number of) x_t's that fit the output perfectly, but if we constrain the sum of norms $\|x_t - x_{t-1}\|_2$, then we will instead look for piecewise constant segments in the parameter space; this model was apparently first proposed in Ohlsson et al. (2010).

Algorithm 2 Active Set Projected Newton (ASPN)

input signal $Y \in \mathbb{R}^{n\times T}$; weights $w \in \mathbb{R}^T_+$; regularization parameters $\lambda \in \mathbb{R}^{T-1}_+$; maximum active set size $k_{\max}$; tolerance ϵ
output: optimized signal $X \in \mathbb{R}^{n\times T}$
initialization: $z \leftarrow 0$
repeat

1. Form dual variables and gradient

$$U \leftarrow YD(DW^{-1}D + Z)^{-1}$$
$$\nabla_z f(z) \leftarrow -\frac{1}{2}(U^2)^T 1 + \frac{1}{2}\lambda^2$$

2. Compute active set, containing all non-zero z_i's and additional element with negative gradients, up to size $k_{\max}$

$$\mathcal{J}^0 \leftarrow \{i : z_i > 0\}$$
$$\mathcal{J}^1 \leftarrow \{i : z_i = 0, \nabla_z f(z) < 0\}$$
$$\mathcal{J} \leftarrow \mathcal{J}^0 \cup \mathcal{J}^1_{1:k_{\max}-|\mathcal{J}^0|}$$

3. Form reduced problem (Y', w') for $\mathcal{J}$ using (17) and solve using projected Newton

$$z_{\mathcal{J}} \leftarrow \text{Projected-Newton}(Y', w', \lambda_{\mathcal{J}})$$

until $\|(\nabla_z f(z))_{\mathcal{J}}\|_2 \le \epsilon$

This model may be cast as the optimization problem

$$\underset{X}{\text{minimize}} \|A \operatorname{vec} X - y\|_2^2 + \|XD\Lambda\|_{1,2}, \tag{18}$$

where $X \in \mathbb{R}^{n\times T}$ is the same optimization variable as previously, $y \in \mathbb{R}^T$ denotes the vector of outputs, vec denotes the vectorization of a matrix (stacking its columns into a single column vector), and $A \in \mathbb{R}^{T\times Tn}$ is the block diagonal matrix

$$A = \begin{bmatrix} a_1^T & 0 & 0 & \cdots \\ 0 & a_2^T & 0 & \cdots \\ 0 & 0 & a_3^T & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{19}$$

While this problem looks very similar to the ordinary GFL setting, the introduction of the additional matrix A renders it substantially more complex. While it is possible to adapt the Newton methods above to solve the problem directly, much of the special problem structure is lost, and it requires, for examples, forming $Tn \times Tn$ block tridiagonal matrices, which is substantially more computationally intensive, especially for large n (the methods scale like $O(n^3)$). While optimization may still be possible with such approaches, we instead adopt a different approach that builds on the alternating direction method of multipliers (ADMM), an algorithm that has attracted great attention recently (e.g. Boyd et al. (2011)). Briefly, ADMM solves problems of the form

$$\underset{x,z}{\text{minimize}}\, f(x) + g(z), \;\; \text{subject to}\, Ax + Bz = c, \tag{20}$$

via a sequence of alternating minimizations over x and z and dual variable updates.

The "standard" ADMM algorithm. The simplest way to apply ADMM to (18), considered for the pure group fused lasso e.g., in Wahlberg et al. (2012), is to introduce variables $Z = XD$, and formulate the problem as

$$\begin{aligned} &\underset{X,Z}{\text{minimize}} \;\; \|A \operatorname{vec} X - Y\|_2^2 + \|Z\Lambda\|_{1,2} \\ &\text{subject to} \;\; XD = Z. \end{aligned} \tag{21}$$

After some derivations, this leads to the updates

$$\begin{aligned} X^{k+1} &\leftarrow \underset{X}{\operatorname{argmin}} \|A \operatorname{vec} X - y\|_2^2 + \tfrac{\rho}{2}\|XD - Z^k + U^k\|_F^2 \\ Z^{k+1} &\leftarrow \underset{Z}{\operatorname{argmin}} \|Z\Lambda\|_{1,2} + \tfrac{\rho}{2}\|X^{k+1}D - Z + U^k\|_F^2 \\ U^{k+1} &\leftarrow U^k + X^{k+1}D - Z^{k+1}, \end{aligned} \tag{22}$$

where ρ acts effectively like a stepsize for the problem. This set of updates is particularly appealing because minimization over X and Z can both be computed in closed form: the minimization over X is unconstrained quadratic optimization, and has the solution

$$\left(A^T A + \rho F^T F\right)^{-1} \left(A^T y + \rho F^T \operatorname{vec}(Z^k - U^k)\right) \tag{23}$$

where $F = (D^T \otimes I)$. Furthermore, these updates can be computed very efficiently, since $\left(A^T A + \rho F^T F\right)$ is block tridiagonal, and since this matrix does not change at each iteration, we can precompute its (sparse) Cholesky decomposition once, and use it for all iterations; using these optimizations, the X update takes time $O(Tn^2)$. Similarly, the Z update is a proximal operator that can be solved by *soft thresholding* the columns of $X^{k+1}D + U^k$ (an $O(Tn)$ operation). Although these elements make the algorithm appealing, they hide a subtle issue: the matrix $(A^T A + \rho F^T F X)$ is poorly conditioned (owing to the poor conditioning of $D^T D$), even for large ρ. Because of this, ADMM needs a large number of iterations for converging to a reasonable solution; even if each iteration is quite efficient, the overall algorithm can still be impractical.

ADMM using the GFL proximal operator. Alternatively, we can derive a different ADMM algorithm by considering instead the formulation

$$\begin{aligned} &\underset{X,Z}{\text{minimize}} \;\; \|A \operatorname{vec} X - Y\|_2^2 + \|ZD\Lambda\|_{1,2} \\ &\text{subject to} \;\; X = Z, \end{aligned} \tag{24}$$

which leads to the iterative updates

$$\begin{aligned} X^{k+1} &\leftarrow \operatorname*{argmin}_X \|A \operatorname{vec} X - y\|_2^2 + \tfrac{\rho}{2}\|X - Z^k + U^k\|_F^2 \\ Z^{k+1} &\leftarrow \operatorname*{argmin}_Z \|ZD\Lambda\|_{1,2} + \tfrac{\rho}{2}\|X^{k+1} - Z + U^k\|_F^2 \\ U^{k+1} &\leftarrow U^k + X^{k+1} - Z^{k+1}. \end{aligned} \tag{25}$$

The X update can still be computed in closed form

$$\operatorname{vec} X^{k+1} = \left(A^T A + \rho I\right)^{-1} \left(A^T y + \rho \operatorname{vec}(Z^k - U^k)\right),$$

which is even simpler to compute than in the previous case, since $A^T A + \rho I$ is block diagonal with blocks $a_i a_i^T + \rho I$, which can be solved for in $O(n)$ time; thus the entire X update takes times $O(Tn)$. The downside is that the Z update, of course, can no longer be solved with soft-thresholding. But the Z update here is precisely in the form of the group fused lasso; thus, we can use ASPN directly to perform the Z update. The main advantage here is that the matrix $A^T A + \rho I$ is much better conditioned, which translates into many fewer iterations of ADMM. Indeed, as we show below, this approach can be many orders of magnitude faster than straight ADMM, which is already a very competitive algorithm for solving these problems.

3.2 Color total variation denoising

Next, we consider color total variation denoising example. Given an $m \times n$ RGB image represented as a third order tensor, $Y \in \mathbb{R}^{3\times m\times n}$, total variation image denoising (Rudin et al., 1992; Blomgren and Chan, 1998) attempts to find an approximation $X \in \mathbb{R}^{3\times m\times n}$ such that differences between pixels in X favor being zero. It does this by solving the optimization problem

$$\begin{aligned} \operatorname*{minimize}_X \tfrac{1}{2}\|X - Y\|_F^2 + \lambda \sum_{i=1}^{m}\sum_{j=1}^{n-1} \|X_{:,i,j} - X_{:,i,j+1}\|_2 \\ + \lambda \sum_{i=1}^{m-1}\sum_{j=1}^{n} \|X_{:,i,j} - X_{:,i+1,j}\|_2, \end{aligned} \tag{26}$$

corresponding to an ℓ_2 norm penalty on the difference between all adjacent pixels, where each pixel $X_{:,i,j}$ is represented as a 3 dimensional vector. We can write this as a sum of $m + n$ group fused lasso problems

$$\begin{aligned} \operatorname*{minimize}_X \sum_{i=1}^{m} \left(\|X_{:,i,:} - Y_{:,i,:}\|_F^2 + \lambda\|X_{:,i,:}D\|_{1,2}\right) \\ + \sum_{j=1}^{n} \left(\|X_{:,:,j} - Y_{:,:,j}\|_F^2 + \lambda\|X_{:,:,j}D\|_{1,2}\right), \end{aligned}$$

where $X_{:,i,:} \in \mathbb{R}^{3\times n}$ denotes the slice of a single row of the image and $X_{:,:,j} \in \mathbb{R}^{3\times n}$ denotes the slice of a single column.

Unfortunately, this optimization problem cannot be solved directly via the group fused lasso, as the difference penalties on the rows and columns for the same matrix X render the problem quite different from the basic GFL. We can, however, adopt an approach similar to the one above, and create separate variables corresponding to the row and column slices, plus a constraint that they be equal; formally, we solve

$$\begin{aligned} \operatorname*{minimize}_{X,Z} \sum_{i=1}^{m} \left(\|X_{:,i,:} - Y_{:,i,:}\|_F^2 + \lambda\|X_{:,i,:}D\|_{1,2}\right) \\ + \sum_{j=1}^{n} \left(\|Z_{:,:,j} - Y_{:,:,j}\|_F^2 + \lambda\|Z_{:,:,j}D\|_{1,2}\right) \\ \text{subject to } X = Z. \end{aligned} \tag{27}$$

The major advantage of this approach is that it decomposes the problem into $m + n$ *independent* GFL tasks, plus a meta-algorithm that adjusts each sub-problem to make the rows and columns agree. Several such algorithms are possible, including ADMM; we present here a slightly simpler scheme known as the "proximal Dykstra" method (Combettes and Pesquet, 2011), which has been previously applied to the case of (single channel, i.e., black and white) total variation denoising (Barbero and Sra, 2011). Starting with $X^0 = Y$, $P^0 = 0$, $Q^0 = 0$, the algorithm iterates as follows:

$$\begin{aligned} Z^{k+1}_{:,:,j} &\leftarrow \mathrm{GFL}(X^k_{:,:,j} + P^k_{:,:,j}, \lambda),\ j = 1, \ldots, n \\ P^{k+1} &\leftarrow P^k + X^k - Z^{k+1} \\ X^{k+1}_{:,i,:} &\leftarrow \mathrm{GFL}(Z^{k+1}_{:,i,:} + Q^k_{:,i,:}, \lambda),\ i = 1, \ldots, m \\ Q^{k+1} &\leftarrow Q^k + Z^{k+1} - X^{k+1}. \end{aligned} \tag{28}$$

Typically, very few iterations (on the order of 10) of this outer loop are need to converge to high accuracy. Furthermore, because each of the m or n GFL problems solved in the first and third steps are independent, they can be trivially parallelized.

4 Experimental results

We present experimental results for our approaches, both on the basic group fused lasso problem, where we compare to several other potential approaches, and on the two applications of linear model segmentation and color total variation denoising. C++ and MATLAB code implementing our methods is available at `http://www.cs.cmu.edu/~mwytock/gfl/`.

4.1 Group fused lasso

Here we evaluate the ASPN algorithm versus several alternatives to solving the group fused lasso problem, evaluated on both synthetic and real data. Figure 1

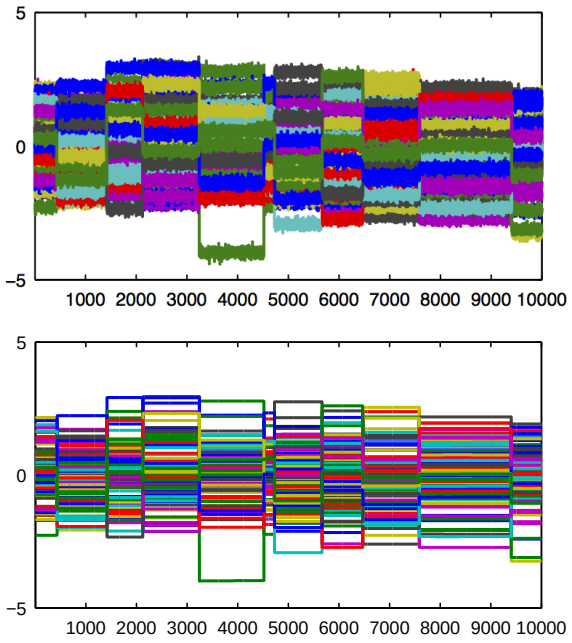

Figure 1: Above: synthetic change point data, with $T = 10000$, $n = 100$, and 10 true change points. Below: recovered signal.

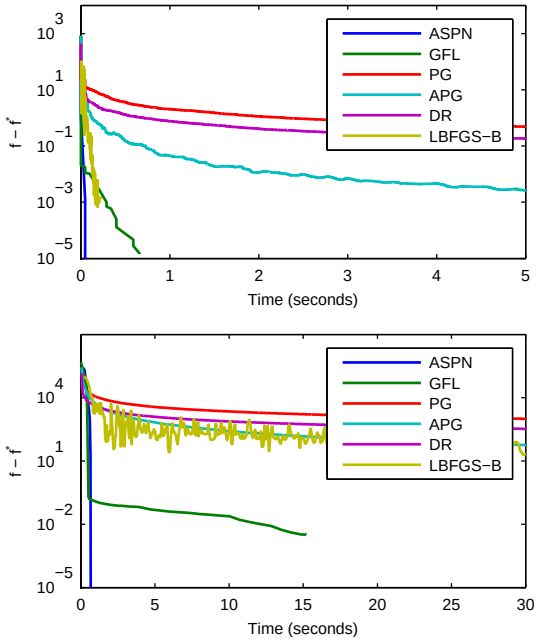

Figure 2: Above: timing results on synthetic problem with $T = 1000$, $n = 10$. Below: timing results on synthetic problem with $T = 10000$, $n = 100$.

shows a synthetic time series with $T = 10,000$, $n = 100$, and 10 discrete change points in the data; the data was generated by uniformly sampling the change points, sampling the mean of each segment from $\mathcal{N}(0, I)$, and then additional Gaussian noise. Figure 1 shows the recovered signal using the group fused lasso with $w_t = 1$, $\lambda_t = 20$. In Figure 2, we show timing results for this problem as well as a smaller problem with $T = 1000$ and $n = 10$; we compare ASPN to GFLseg (Bleakley and Vert, 2011) (which uses coordinate descent on the primal problem), an accelerated projected gradient on the dual (i.e., the FISTA algorithm) (Beck and Teboulle, 2009), Douglas-Rachford splitting (Combettes and Pesquet, 2007) (a generalization of ADMM that performs slightly better here), a projected gradient on the dual (Alaíz et al., 2013), and LBFGS-B (Byrd et al., 1995) applied to the dual of the dual. In all cases, ASPN performs as well as (often much better than) the alternatives.

Next, we evaluate how the ASPN algorithm scales as a function of the number of time points T and the number of change points at the solution, k. In Figure 3, the first set of experiments shows that when the number of change points at the solution is fixed ($k = 10$), the amount of time required for a highly accurate solution remains small even for large T, agreeing with analysis that shows the number of operations required is $O(T)$. In particular, a solution accurate to 10^{-6} is found in 4.8 seconds on a problem with $T = 10^6$ time points.

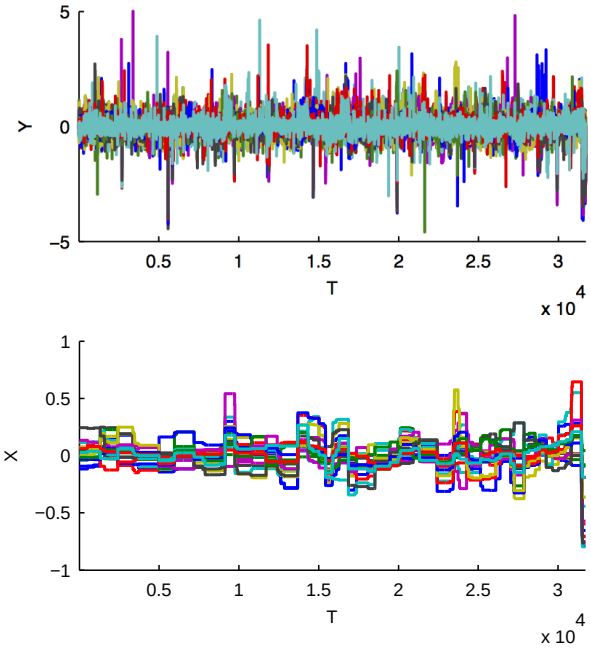

Figure 4: Above: Lung data from (Bleakley and Vert, 2011). Below: recovered signal using group fused lasso.

However, in the next set of experiments, we see that compute time grows rapidly as a function of k due to the $O(k^3)$ operations required to compute the Newton step, suggesting that the proposed method is most appropriate for problems with sparse solutions.

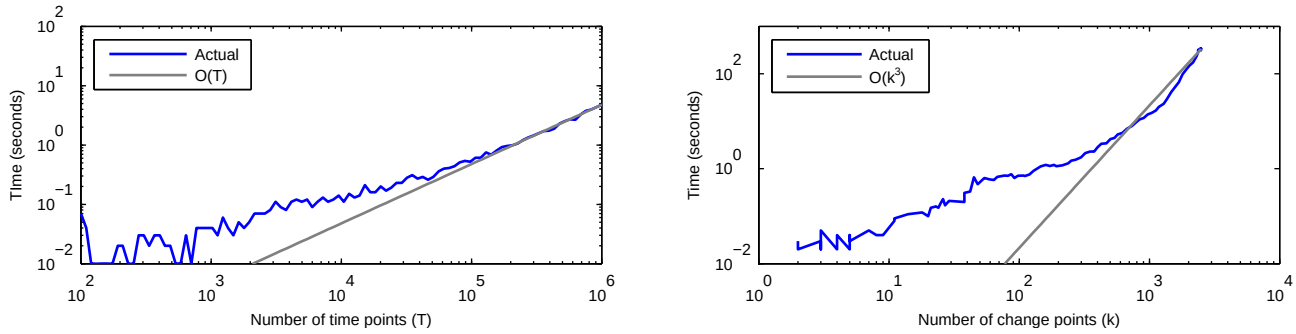

Figure 3: Left: timing results vs. number of change points at solution for synthetic problem with $T = 10000$ and $n = 10$. Right: timing results for varying T, $n = 10$, and sparse solution with 10 change points.

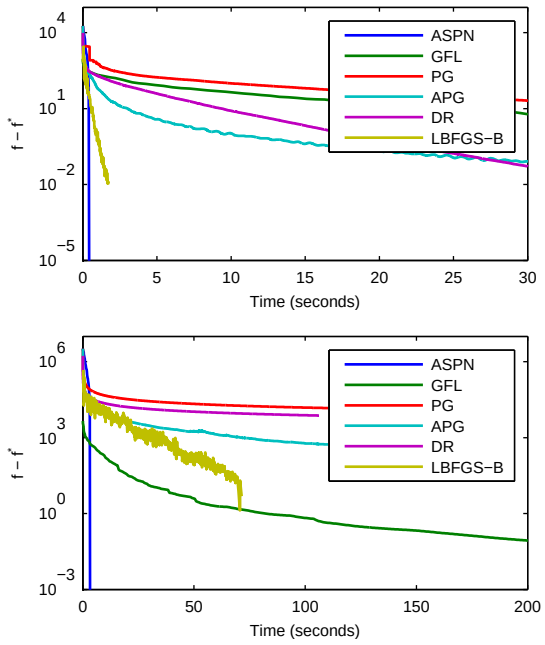

Figure 5: Above: Timing results on bladder problem, $T = 2143$, $n = 57$. Below: Timing results on lung problem, $T = 31708$, $n = 18$.

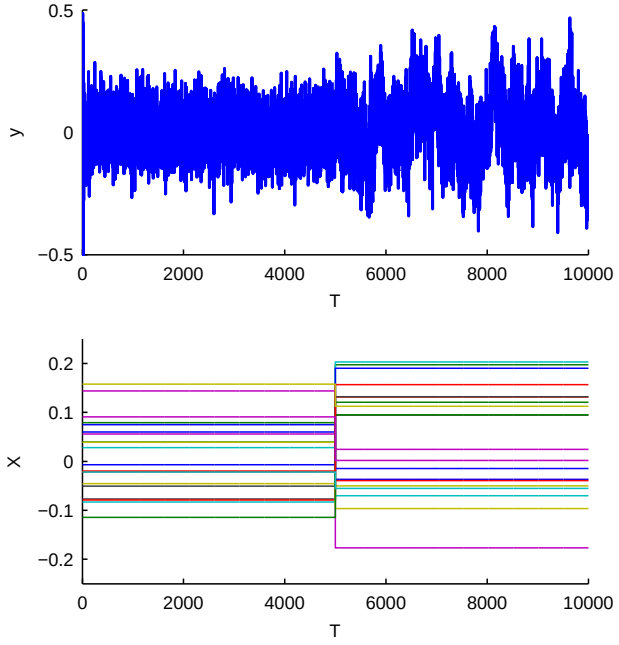

Figure 6: Above: Observed autoregressive signal z_t. Below: true autoregressive model parameters.

Finally, we evaluate the algorithm on two real time series previously used with the group fused lasso (Bleakley and Vert, 2011), from DNA profiles of bladder and lung cancer sequences. Figure 4 shows one of these two series, along with the approximation produced by the group fused lasso. Figure 5 shows timing results for the above methods again on this problem: here we observe the same overall behavior, that ASPN typically dominates the other approaches.

4.2 Linear regression segmentation

Here we apply the two different ADMM methods discussed in Section 3.1 to the task of segmenting autoregressive time series models. In particular, we observe some time series $z_1, \ldots, z_T$, and we fit a linear model to this data $z_t \approx a_t^T x_t$ where $a_t = (z_{t-1}, z_{t-2}, \ldots, z_{t-n})$. Figure 6 shows an example time series generated by this process, as well as the true underlying model that generated the data (with additional noise). This is the rough setting used in (Ohlsson et al., 2010), which was the first example we are aware of that uses such regularization techniques within an linear regression framework. Figure 7 shows the model parameters recovered using the method from Section 3.1, which here match the ground truth closely.

Of more importance, though, is the comparison between the two different ADMM approaches. Figure 8

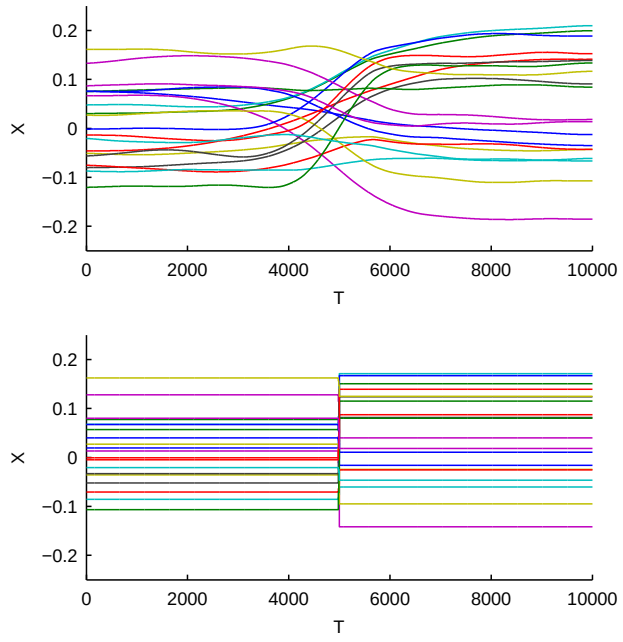

Figure 7: Above: Autoregressive parameters recovered with "simple" ADMM algorithm. Below: parameters recovered using alternative ADMM w/ ASPN.

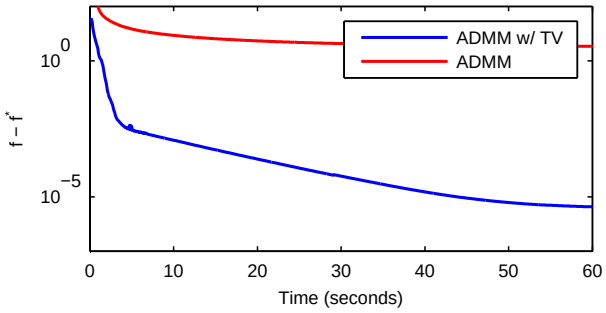

Figure 8: Convergence of simple ADMM versus alternative ADMM w/ ASPN.

shows convergence versus running time and here the "simple" ADMM approach, which encodes the difference operator in the constraints (and thus has simpler updates), converges significantly slower than our alternative. Importantly, the X axis in this figure is measured in time, and we emphasize that even though the "simple" ADMM updates are individually slightly faster (they do not involve GFL subproblems), their overall performance is much poorer. Further, as illustrated in Figure 7, the "simple" ADMM approach never actually obtains a piecewise constant X except at the optimum, which is never reached in practice.

Figure 9: Left: original image. Middle: image corrupted with Gaussian noise. Right: imaged recovered with total variation using proximal Dykstra and ASPN.

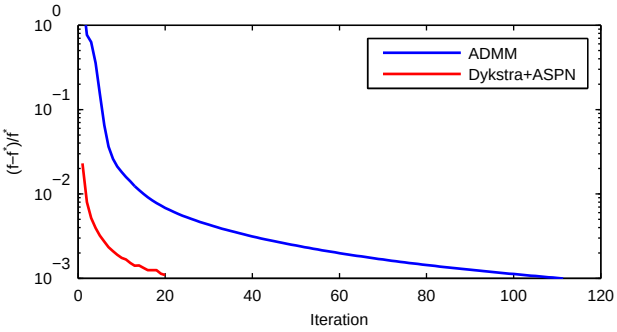

Figure 10: Comparison of proximal Dykstra method to ADMM for TV denoising of color image.

4.3 Color total variation denoising

Finally, as described in Section 3.2, we apply the proximal Dykstra algorithm, using ASPN as a fast subroutine, to color image denoising. Figure 9 shows a 256x256 image generated by combining various solid-colored shapes, corrupted with per-RGB-component noise of $\mathcal{N}(0, 0.1)$, and then recovered with total variation denoising. There has been enormous work on total variation denoising, and while a full comparison is beyond the scope of this paper, ADMM or methods such as those used by the FTVd routines in Yang et al. (2009), for instance, are considered to be some of the fastest for this problem. In Figure 10, we show the performance of our approach and ADMM versus iteration number, and as expected observe better convergence; for single-core systems, ADMM is ultimately a better solution for this problem, since each iteration of ADMM takes about 0.767 seconds in our implementation whereas 512 calls to ASPN take 20.4 seconds. However, the advantage to the ASPN approach is that all these calls can be trivially parallelized for a 256X speedup (the calls are independent and all code is CPU-bound), whereas parallelizing a generic sparse matrix solve, as needed for ADMM-based approaches, is much more challenging and thus per-iteration performance highlights the potential benefits of the ASPN approach.

Acknowledgements This work was supported in part by the National Science Foundation under award IIS-1320402, and by support from Google.

References

Alaíz, C. M., Jiménez, Á. B., and Dorronsoro, J. R. (2013). Group fused lasso. In *International Conference on Artificial Neural Networks and Machine Learning (ICANN)*, pages 66–73.

Barbero, Á. and Sra, S. (2011). Fast newton-type methods for total variation regularization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 313–320.

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

Bertsekas, D. P. (1982). Projected newton methods for optimization problems with simple constraints. *SIAM Journal on control and Optimization*, 20(2):221–246.

Bleakley, K. and Vert, J.-P. (2011). The group fused lasso for multiple change-point detection. *arXiv preprint arXiv:1106.4199*.

Blomgren, P. and Chan, T. F. (1998). Color TV: total variation methods for restoration of vector-valued images. *Image Processing, IEEE Transactions on*, 7(3):304–309.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Combettes, P. L. and Pesquet, J.-C. (2007). A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):564–574.

Combettes, P. L. and Pesquet, J.-C. (2011). Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer.

Kim, J. and Park, H. (2010). Fast active-set-type algorithms for L1-regularized linear regression. In *International Conference on Artificial Intelligence and Statistics*, pages 397–404.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007). Efficient sparse coding algorithms. In *Neural Information Processing Systems*.

Ohlsson, H., Ljung, L., and Boyd, S. (2010). Segmentation of ARX-models using sum-of-norms regularization. *Automatica*, 46(6):1107–1111.

Portugal, L. F., Judice, J. J., and Vicente, L. N. (1994). A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables. *Mathematics of Computation*, 63(208):625–643.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108.

Usmani, R. A. (1994). Inversion of a tridiagonal jacobi matrix. *Linear Algebra and Its Applications*, 212:413–414.

Wahlberg, B., Boyd, S., Annergren, M., and Wang, Y. (2012). An ADMM algorithm for a class of total variation regularized estimation problems. *arXiv preprint arXiv:1203.1828*.

Yang, J., Yin, W., Zhang, Y., and Wang, Y. (2009). A fast algorithm for edge-preserving variational multichannel image restoration. *SIAM Journal on Imaging Sciences*, 2(2):569–592.