Structured Sparsity via Alternating Direction Methods*

Zhiwei (Tony) Qin † Donald Goldfarb ‡

Department of Industrial Engineering and Operations Research
Columbia University
New York, NY 10027

December 14, 2011

Abstract

We consider a class of sparse learning problems in high dimensional feature space regularized by a structured sparsity-inducing norm that incorporates prior knowledge of the group structure of the features. Such problems often pose a considerable challenge to optimization algorithms due to the non-smoothness and non-separability of the regularization term. In this paper, we focus on two commonly adopted sparsity-inducing regularization terms, the overlapping Group Lasso penalty l_1/l_2-norm and the l_1/l_∞-norm. We propose a unified framework based on the augmented Lagrangian method, under which problems with both types of regularization and their variants can be efficiently solved. As one of the core building-blocks of this framework, we develop new algorithms using a partial-linearization/splitting technique and prove that the accelerated versions of these algorithms require $O(\frac{1}{\sqrt{\epsilon}})$ iterations to obtain an ϵ-optimal solution. We compare the performance of these algorithms against that of the alternating direction augmented Lagrangian and FISTA methods on a collection of data sets and apply them to two real-world problems to compare the relative merits of the two norms.

Keywords: structured sparsity, overlapping Group Lasso, alternating direction methods, variable splitting, augmented Lagrangian

1 Introduction

For feature learning problems in a high-dimensional space, sparsity in the feature vector is usually a desirable property. Many statistical models have been proposed in the literature to enforce sparsity, dating back to the classical Lasso model (l_1-regularization) [45, 10]. The Lasso model is particularly appealing because it can be solved by very efficient proximal gradient methods; e.g., see [12]. However, the Lasso does not take into account the structure of the features [50]. In many real applications, the features in a learning problem are often highly correlated, exhibiting a group structure. Structured sparsity has been shown to be effective in those cases. The Group Lasso model [49, 2, 41] assumes disjoint groups and enforces sparsity on the pre-defined groups of features. This model has been extended to allow for groups that are hierarchical as well as overlapping [25, 27, 3] with a wide array of applications from gene selection [27] to computer

*Research supported in part by DMS 10-16571, ONR Grant N00014-08-1-1118 and DOE Grant DE-FG02-08ER25856.

†Email: zq2107@columbia.edu.

‡Email: goldfarb@columbia.edu.

vision [23, 26]. For image denoising problems, extensions with non-integer block sizes and adaptive partitions have been proposed in [36, 37]. In this paper, we consider the following basic model of minimizing the squared-error loss with a regularization term to induce group sparsity:

$$\min_{x\in\mathbb{R}^m} L(x) + \Omega(x), \tag{1}$$

where

$$\begin{aligned} L(x) &= \frac{1}{2}\|Ax-b\|^2, \qquad A\in\mathbb{R}^{n\times m}, \\ \Omega(x) &= \begin{cases} \Omega_{l_1/l_2}(x) \equiv \lambda\sum_{s\in\mathcal{S}} w_s\|x_s\|, & \text{or} \\ \Omega_{l_1/l_\infty}(x) \equiv \lambda\sum_{s\in\mathcal{S}} w_s\|x_s\|_\infty & , \end{cases} \end{aligned} \tag{2}$$

$\mathcal{S} = \{s_1, \cdots, s_{|\mathcal{S}|}\}$ is the set of group indices with $|\mathcal{S}| = J$, and the elements (features) in the groups possibly overlap [11, 30, 25, 3]. In this model, $\lambda, w_s, \mathcal{S}$ are all pre-defined. $\|\cdot\|$ without a subscript denotes the l_2-norm. We note that the penalty term $\Omega_{l_1/l_2}(x)$ in (2) is different from the one proposed in [24], although both are called overlapping Group Lasso penalties. In particular, (1)-(2) cannot be cast into a non-overlapping group lasso problem as done in [24].

1.1 Related Work

Two proximal gradient methods have been proposed to solve a close variant of (1) with an l_1/l_2 penalty,

$$\min_{x\in\mathbb{R}^m} L(x) + \Omega_{l_1/l_2}(x) + \lambda\|x\|_1, \tag{3}$$

which has an additional l_1-regularization term on x. Chen et al. [11] replace $\Omega_{l_1/l_2}(x)$ with a smooth approximation $\Omega_\eta(x)$ by using Nesterov's smoothing technique [33] and solve the resulting problem by the Fast Iterative Shrinkage Thresholding algorithm (FISTA) [4]. The parameter η is a smoothing parameter, upon which the practical and theoretical convergence speed of the algorithm critically depends. Liu and Ye [29] also apply FISTA to solve (3), but in each iteration, they transform the computation of the proximal operator associated with the combined penalty term into an equivalent constrained smooth problem and solve it by Nesterov's accelerated gradient descent method [33]. Mairal et al. [30] apply the accelerated proximal gradient method to (1) with l_1/l_∞ penalty and propose a network flow algorithm to solve the proximal problem associated with $\Omega_{l_1/l_\infty}(x)$. Mosci et al.'s method [32] for solving the Group Lasso problem in [24] is in the same spirit as [29], but their approach uses a projected Newton method.

1.2 Our Contributions

We take a unified approach to tackle problem (1) with both l_1/l_2- and l_1/l_∞-regularizations. Our strategy is to develop efficient algorithms based on the Alternating Linearization Method with Skipping (ALM-S) [19] and FISTA for solving an equivalent constrained version of problem (1) (to be introduced in Section 2) in an augmented Lagrangian method framework. Specifically, we make the following contributions in this paper:

- We build a general framework based on the augmented Lagrangian method, under which learning problems with both l_1/l_2 and l_1/l_∞ regularizations (and their variants) can be solved. This framework allows for experimentation with its key building blocks.

- We propose new algorithms: ALM-S with partial splitting (APLM-S) and FISTA with partial linearization (FISTA-p), to serve as the key building block for this framework. We prove that APLM-S and FISTA-p have convergence rates of $O(\frac{1}{k})$ and $O(\frac{1}{k^2})$ respectively, where k is the number of iterations. Our algorithms are easy to implement and tune, and they do not require line-search, eliminating the need to evaluate the objective function at every iteration.

- We evaluate the quality and speed of the proposed algorithms and framework against state-of-the-art approaches on a rich set of synthetic test data and compare the l_1/l_2 and l_1/l_∞ models on breast cancer gene expression data [47] and a video sequence background subtraction task [30].

2 A Variable-Splitting Augmented Lagrangian Framework

In this section, we present a unified framework, based on variable splitting and the augmented Lagrangian method for solving (1) with both l_1/l_2- and l_1/l_∞-regularizations. This framework reformulates problem (1) as an equivalent linearly-constrained problem, by using the following variable-splitting procedure.

Let $y \in \mathbb{R}^{\sum_{s\in\mathcal{S}}|s|}$ be the vector obtained from the vector $x \in \mathbb{R}^m$ by repeating components of x so that no component of y belongs to more than one group. Let $M = \sum_{s\in\mathcal{S}}|s|$. The relationship between x and y is specified by the linear constraint $Cx = y$, where the (i,j)-th element of the matrix $C \in \mathbb{R}^{M\times m}$ is

$$C_{i,j} = \begin{cases} 1, & \text{if } y_i \text{ is a replicate of } x_j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

For examples of C, refer to [11]. Consequently, (1) is equivalent to

$$\begin{aligned} \min \quad & F_{obj}(x,y) \equiv \frac{1}{2}\|Ax-b\|^2 + \tilde{\Omega}(y) \\ s.t. \quad & Cx = y, \end{aligned} \tag{5}$$

where $\tilde{\Omega}(y)$ is the non-overlapping group-structured penalty term corresponding to $\Omega(y)$ defined in (2). Note that C is a highly sparse matrix, and $D = C^TC$ is a diagonal matrix with the diagonal entries equal to the number of times that each entry of x is included in some group. Problem (5) now includes two sets of variables x and y, where x appears only in the loss term $L(x)$ and y appears only in the penalty term $\tilde{\Omega}(y)$.

All the non-overlapping versions of $\Omega(\cdot)$, including the Lasso and Group Lasso, are special cases of $\Omega(\cdot)$, with $C = I$, i.e. $x = y$. Hence, (5) in this case is equivalent to applying variable-splitting on x. Problems with a composite penalty term, such as the Elastic Net, $\lambda_1\|x\|_1 + \lambda_2\|x\|^2$, can also be reformulated in a similar way by merging the smooth part of the penalty term ($\lambda_2\|x\|^2$ in the case of the Elastic Net) with the loss function $L(x)$.

To solve (5), we apply the augmented Lagrangian method [22, 38, 34, 6] to it. This method, Algorithm 2.1, minimizes the augmented Lagrangian

$$\mathcal{L}(x,y,v) = \frac{1}{2}\|Ax-b\|^2 - v^T(Cx-y) + \frac{1}{2\mu}\|Cx-y\|^2 + \tilde{\Omega}(y) \tag{6}$$

exactly for a given Lagrange multiplier v in every iteration followed by an update to v. The parameter μ in (6) controls the amount of weight that is placed on violations of the constraint $Cx = y$. Algorithm 2.1 can also be viewed as a dual ascent algorithm applied to $P(v) = \min_{x,y}\mathcal{L}(x,y,v)$ [5], where v is the dual variable, $\frac{1}{\mu}$ is the step-length, and $Cx - y$ is the gradient $\nabla_v P(v)$. This

Algorithm 2.1 AugLag

1: Choose x^0, y^0, v^0.
2: **for** $l = 0, 1, \cdots$ **do**
3: $(x^{l+1}, y^{l+1}) \leftarrow \arg\min_{x,y} \mathcal{L}(x, y, v^l)$
4: $v^{l+1} \leftarrow v^l - \frac{1}{\mu}(Cx^{l+1} - y^{l+1})$
5: Update μ according to the chosen updating scheme.
6: **end for**

algorithm does not require μ to be very small to guarantee convergence to the solution of problem (5) [34]. However, solving the problem in Line 3 of Algorithm 2.1 exactly can be very challenging in the case of structured sparsity. We instead seek an approximate minimizer of the augmented Lagrangian via the abstract subroutine ApproxAugLagMin(x, y, v). The following theorem [40] guarantees the convergence of this inexact version of Algorithm 2.1.

Theorem 2.1. *Let* $\alpha^l := \mathcal{L}(x^l, y^l, v^l) - \inf_{x\in\mathbb{R}^m, y\in\mathbb{R}^M} \mathcal{L}(x, y, v^l)$ *and* F^* *be the optimal value of problem* (5)*. Suppose problem* (5) *satisfies the modified Slater's condition, and*

$$\sum_{l=1}^{\infty} \sqrt{\alpha^l} < +\infty. \tag{7}$$

Then, the sequence $\{v^l\}$ *converges to* v^**, which satisfies*

$$\inf_{x\in\mathbb{R}^m, y\in\mathbb{R}^M} \big(F_{obj}(x, y) - (v^*)^T(Cx - y)\big) = F^*,$$

while the sequence $\{x^l, y^l\}$ *satisfies* $\lim_{l\to\infty} Cx^l - y^l = 0$ *and* $\lim_{l\to\infty} F_{obj}(x^l, y^l) = F^*$.

The condition (7) requires the augmented Lagrangian subproblem be solved with increasing accuracy. We formally state this framework in Algorithm 2.2. We index the iterations of Algorithm

Algorithm 2.2 OGLasso-AugLag

1: Choose x^0, y^0, v^0.
2: **for** $l = 0, 1, \cdots$ **do**
3: $(x^{l+1}, y^{l+1}) \leftarrow$ ApproxAugLagMin(x^l, y^l, v^l), to compute an approximate minimizer of $\mathcal{L}(x, y, v^l)$
4: $v^{l+1} \leftarrow v^l - \frac{1}{\mu}(Cx^{l+1} - y^{l+1})$
5: Update μ according to the chosen updating scheme.
6: **end for**

2.2 by l and call them 'outer iterations'. In Sections 3, we develop algorithms that implement ApproxAugLagMin(x, y, v). The iterations of these subroutine are indexed by k and are called 'inner iterations'.

3 Methods for Approximately Minimizing the Augmented Lagrangian

In this section, we use the overlapping Group Lasso penalty $\Omega(x) = \lambda \sum_{s\in\mathcal{S}} w_s \|x_s\|$ to illustrate the optimization algorithms under discussion. The case of l_1/l_∞-regularization will be discussed in Section 4. From now on, we assume without loss of generality that $w_s = 1$ for every group s.

3.1 Alternating Direction Augmented Lagrangian (ADAL) Method

The well-known Alternating Direction Augmented Lagrangian (ADAL) method [15, 17, 18, 8][1] approximately minimizes the augmented Lagrangian by minimizing (6) with respect to x and y alternatingly and then updates the Lagrange multiplier v on each iteration (e.g., see [7], Section 3.4). Specifically, the single-iteration procedure that serves as the procedure ApproxAugLagMin(x, y, v) is given below as Algorithm 3.1.

Algorithm 3.1 ADAL

1: Given x^l, y^l, and v^l.
2: $x^{l+1} \leftarrow \arg\min_x \mathcal{L}(x, y^l, v^l)$
3: $y^{l+1} \leftarrow \arg\min_y \mathcal{L}(x^{l+1}, y, v^l)$
4: **return** x^{l+1}, y^{l+1}.

The ADAL method, also known as the alternating direction method of multipliers (ADMM) and the split Bregman method, has recently been applied to problems in signal and image processing [12, 1, 20] and low-rank matrix recovery [28]. Its convergence has been established in [15]. This method can accommodate a sum of more than two functions. For example, by applying variable-splitting (e.g., see [7, 8]) to the problem $\min_x f(x) + \sum_{i=1}^K g_i(C_i x)$, it can be transformed into

$$\begin{aligned} \min_{x, y_1, \cdots, y_K} \quad & f(x) + \sum_{i=1}^K g_i(y_i) \\ s.t. \quad & y_i = C_i x, \quad i = 1, \cdots, K. \end{aligned}$$

The subproblems corresponding to y_i's can thus be solved simultaneously by the ADAL method. This so-called simultaneous direction method of multipliers (SDMM) [42] is related to Spingarn's method of partial inverses [43] and has been shown to be a special instance of a more general parallel proximal algorithm with inertia parameters [35].

Note that the problem solved in Line 3 of Algorithm 3.1,

$$y^{l+1} = \arg\min_y \mathcal{L}(x^{l+1}, y, v^l) \equiv \arg\min_y \left\{ \frac{1}{2\mu} \|d^l - y\|^2 + \tilde{\Omega}(y) \right\}, \tag{8}$$

where $d^l = Cx^{l+1} - \mu v^l$, is group-separable and hence can be solved in parallel. As in [39], each subproblem can be solved by applying the block soft-thresholding operator, $T(d_s^l, \mu\lambda) \equiv \frac{d_s^l}{\|d_s^l\|} \max(0, \|d_s^l\| - \lambda\mu), s = 1, \cdots, J$. Solving for x^{l+1} in Line 2 of Algorithm 3.1, i.e.

$$x^{l+1} = \arg\min_x \mathcal{L}(x, y^l, v^l) \equiv \arg\min_x \left\{ \frac{1}{2} \|Ax - b\|^2 - (v^l)^T Cx + \frac{1}{2\mu} \|Cx - y^l\|^2 \right\}, \tag{9}$$

involves solving the linear system

$$(A^T A + \frac{1}{\mu} D)x = A^T b + C^T v^l + \frac{1}{\mu} C^T y^l, \tag{10}$$

where the matrix on the left hand side of (10) has dimension $m \times m$. Many real-world data sets, such as gene expression data, are highly under-determined. Hence, the number of features (m) is much

[1] Recently, Mairal et al. [31] also applied ADAL with two variants based on variable-splitting to the overlapping Group Lasso problem.

larger than the number of samples (n). In such cases, one can use the Sherman-Morrison-Woodbury formula,

$$(A^TA + \frac{1}{\mu}D)^{-1} = \mu D^{-1} - \mu^2 D^{-1}A^T(I + \mu AD^{-1}A^T)^{-1}AD^{-1}, \tag{11}$$

and solve instead an $n \times n$ linear system involving the matrix $I + \mu AD^{-1}A^T$. In addition, as long as μ stays the same, one has to factorize $A^TA + \frac{1}{\mu}D$ or $I + \mu AD^{-1}A^T$ only once and store their factors for subsequent iterations.

When both n and m are very large, it might be infeasible to compute or store A^TA, not to mention its eigen-decomposition, or the Cholesky decomposition of $A^TA + \frac{1}{\mu}D$. In this case, one can solve the linear systems using the preconditioned Conjugate Gradient (PCG) method [21]. Similar comments apply to the other algorithms proposed in Sections 3.2 - 3.4 below.Alternatively, we can apply FISTA to Line 3 in Algorithm 2.2 (see Section 3.5).

3.2 ALM-S: partial split (APLM-S)

We now consider applying the Alternating Linearization Method with Skipping (ALM-S) from [19] to approximately minimize (6). In particular, we apply variable splitting (Section 2) to the variable y, to which the group-sparse regularizer $\tilde{\Omega}$ is applied, (the original ALM-S splits both variables x and y,) and re-formulate (6) as follows.

$$\begin{array}{ll} \min\limits_{x,y,\bar{y}} & \frac{1}{2}\|Ax-b\|^2 - v^T(Cx-y) + \frac{1}{2\mu}\|Cx-y\|^2 + \tilde{\Omega}(\bar{y}) \\ s.t. & y = \bar{y}. \end{array} \tag{12}$$

Note that the Lagrange multiplier v is fixed here. Defining

$$f(x,y) := \frac{1}{2}\|Ax-b\|^2 - v^T(Cx-y) + \frac{1}{2\mu}\|Cx-y\|^2, \tag{13}$$

$$g(y) = \tilde{\Omega}(y) = \lambda\sum_s \|y_s\|, \tag{14}$$

problem (12) is of the form

$$\begin{array}{ll} \min & f(x,y) + g(\bar{y}) \\ s.t. & y = \bar{y}, \end{array} \tag{15}$$

to which we now apply partial-linearization.

3.2.1 Partial linearization and convergence rate analysis

Let us define

$$F(x,y) := f(x,y) + g(y) = \mathcal{L}(x,y;v), \tag{16}$$

$$\mathcal{L}_\rho(x,y,\bar{y},\gamma) := f(x,y) + g(\bar{y}) + \gamma^T(\bar{y}-y) + \frac{1}{2\rho}\|\bar{y}-y\|^2, \tag{17}$$

where γ is the Lagrange multiplier in the augmented Lagrangian (17) corresponding to problem (15), and $\gamma_g(\bar{y})$ is a sub-gradient of g at $\bar{y}$. We now present our partial-split alternating linearization algorithm to implement ApproxAugLagMin(x,y,v) in Algorithm 2.2.

We note that in Line 6 in Algorithm 3.2,

$$x^{k+1} = \arg\min_x \mathcal{L}_\rho(x; y^{k+1}, \bar{y}^k, \gamma^k) \equiv \arg\min_x f(x; y^{k+1}) \equiv \arg\min_x f(x; \bar{y}^k). \tag{18}$$

Now, we have a variant of Lemma 2.2 in [19].

Algorithm 3.2 APLM-S

1: Given $x^0, \bar{y}^0, v$. Choose ρ, γ^0, such that $-\gamma^0 \in \partial g(\bar{y}^0)$. Define $f(x,y)$ as in (13).
2: **for** $k = 0, 1, \cdots$ until stopping criterion is satisfied **do**
3: $\quad (x^{k+1}, y^{k+1}) \leftarrow \arg\min_{x,y} \mathcal{L}_\rho(x, y, \bar{y}^k, \gamma^k)$.
4: $\quad$ **if** $F(x^{k+1}, y^{k+1}) > \mathcal{L}_\rho(x^{k+1}, y^{k+1}, \bar{y}^k, \gamma^k)$ **then**
5: $\quad\quad y^{k+1} \leftarrow \bar{y}^k$
6: $\quad\quad x^{k+1} \leftarrow \arg\min_x f(x, y^{k+1}) \equiv \arg\min_x \mathcal{L}_\rho(x; y^{k+1}, \bar{y}^k, \gamma^k)$
7: $\quad$ **end if**
8: $\quad \bar{y}^{k+1} \leftarrow p_f(x^{k+1}, y^{k+1}) \equiv \arg\min_{\bar{y}} \mathcal{L}_\rho(x^{k+1}, y^{k+1}, \bar{y}, \nabla_y f(x^{k+1}, y^{k+1}))$
9: $\quad \gamma^{k+1} \leftarrow \nabla_y f(x^{k+1}, y^{k+1}) - \frac{y^{k+1} - \bar{y}^{k+1}}{\rho}$
10: **end for**
11: **return** $(x^{K+1}, \bar{y}^{K+1})$

Lemma 3.1. *For any (x,y), if $\bar{q} := \arg\min_{\bar{y}} \mathcal{L}_\rho(x, y, \bar{y}, \nabla_y f(x,y))$ and*

$$F(x, \bar{q}) \le \mathcal{L}_\rho(x, y, \bar{q}, \nabla_y f(x,y)), \tag{19}$$

then for any $(\bar{x}, \bar{y})$,

$$2\rho(F(\bar{x}, \bar{y}) - F(x, \bar{q})) \ge \|\bar{q} - \bar{y}\|^2 - \|y - \bar{y}\|^2 + 2\rho((\bar{x} - x)^T \nabla_x f(x,y)). \tag{20}$$

Similarly, for any $\bar{y}$, if $(p,q) := \arg\min_{x,y} \mathcal{L}_\rho(x, y, \bar{y}, -\gamma_g(\bar{y}))$ and

$$F((p,q)) \le \mathcal{L}_\rho((p,q), \bar{y}, -\gamma_g(\bar{y})), \tag{21}$$

then for any (x,y),

$$2\rho(F(x,y) - F((p,q))) \ge \|(p,q)_y - y\|^2 - \|\bar{y} - y\|^2. \tag{22}$$

Proof. See Appendix A. □

Algorithm 3.2 checks condition (21) at Line 4 because the function g is non-smooth and condition (21) may not hold no matter what the value of ρ is. When this condition is violated, a skipping step occurs in which the value of y is set to the value of $\bar{y}$ in the previous iteration (Line 5) and $\mathcal{L}_\rho$ re-minimized with respect to x (Line 6) to ensure convergence. Let us define a *regular iteration* of Algorithm 3.2 to be an iteration where no skipping step occurs, i.e. Lines 5 and 6 are not executed. Likewise, we define a *skipping iteration* to be an iteration where a skipping step occurs. Now, we are ready to state the iteration complexity result for APLM-S.

Theorem 3.1. *Assume that $\nabla_y f(x,y)$ is Lipschitz continuous with Lipschitz constant $L_y(f)$, i.e. for any x, $\|\nabla_y f(x,y) - \nabla_y f(x,z)\| \le L_y(f)\|y - z\|$, for all y and z. For $\rho \le \frac{1}{L_y(f)}$, the iterates $(x^k, \bar{y}^k)$ in Algorithm 3.2 satisfy*

$$F(x^k, \bar{y}^k) - F(x^*, y^*) \le \frac{\|\bar{y}^0 - y^*\|^2}{2\rho(k + k_n)}, \quad \forall k, \tag{23}$$

where (x^*, y^*) is an optimal solution to (12), and k_n is the number of regular iterations among the first k iterations.

Proof. See Appendix B. □

Remark 3.1. *For Theorem 3.1 to hold, we need* $\rho \leq \frac{1}{L_y(f)}$. *From the definition of* $f(x,y)$ *in* (13), *it is easy to see that* $L_y(f) = \frac{1}{\mu}$ *regardless of the loss function* $L(x)$. *Hence, we set* $\rho = \mu$, *so that condition* (19) *in Lemma 3.1 is satisfied.*

In Section 3.3, we will discuss the case where the iterations entirely consist of skipping steps. We will show that this is equivalent to ISTA [4] with partial linearization as well as a variant of ADAL. In this case, the inner Lagrange multiplier γ is redundant.

3.2.2 Solving the subproblems

We now show how to solve the subproblems in Algorithm 3.2. First, observe that since $\rho = \mu$

$$\arg\min_{\bar{y}} \mathcal{L}_\rho(x, y, \bar{y}, \nabla_y f(x,y)) \equiv \arg\min_{\bar{y}} \left\{ \nabla_y f(x,y)^T \bar{y} + \frac{1}{2\mu}\|\bar{y} - y\|^2 + g(\bar{y}) \right\} \tag{24}$$

$$\equiv \arg\min_{\bar{y}} \left\{ \frac{1}{2\mu}\|d - \bar{y}\|^2 + \lambda \sum_s \|\bar{y}_s\| \right\}, \tag{25}$$

where $d = Cx - \mu v$. Hence, $\bar{y}$ can be obtained by applying the block soft-thresholding operator $T(d_s, \mu\lambda)$ as in Section 3.1. Next consider the subproblem

$$\min_{(x,y)} \mathcal{L}_\rho(x, y, \bar{y}, \gamma) \equiv \min_{(x,y)} \left\{ f(x,y) + \gamma^T(\bar{y} - y) + \frac{1}{2\mu}\|\bar{y} - y\|^2 \right\}. \tag{26}$$

It is easy to verify that solving the linear system given by the optimality conditions for (26) by block Gaussian elimination yields the system

$$\left(A^T A + \frac{1}{2\mu} D \right) x = r_x + \frac{1}{2} C^T r_y \tag{27}$$

for computing x, where $r_x = A^T b + C^T v$ and $r_y = -v + \gamma + \frac{\bar{y}}{\rho}$. Then y can be computed as $(\frac{\mu}{2})(r_y + \frac{1}{\mu} Cx)$.

As in Section 3.1, only one Cholesky factorization of $A^T A + \frac{1}{2\mu} D$ is required for each invocation of Algorithm 3.2. Hence, the amount of work involved in each iteration of Algorithm 3.2 is comparable to that of an ADAL iteration.

It is straightforward to derive an accelerated version of Algorithm 3.2, which we shall refer to as FAPLM-S, that corresponds to a partial-split version of the FALM algorithm proposed in [19] and also requires $O(\sqrt{\frac{L(f)}{\epsilon}})$ iterations to obtain an ϵ-optimal solution. In Section 3.4, we present an algorithm FISTA-p, which is a special version of FAPLM-S in which every iteration is a skipping iteration and which has a much simpler form than FAPLM-S, while having essentially the same iteration complexity.

It is also possible to apply ALM-S directly, which splits both x and y, to solve the augmented Lagrangian subproblem. Similar to (12), we reformulate (6) as

$$\begin{aligned} \min_{(x,y),(\bar{x},\bar{y})} \quad & \frac{1}{2}\|Ax - b\|^2 - v^T(Cx - y) + \frac{1}{2\mu}\|Cx - y\|^2 + \lambda \sum_s \|\bar{y}_s\| \\ s.t. \quad & x = \bar{x}, \\ & y = \bar{y}. \end{aligned} \tag{28}$$

The functions f and g are defined as in (13) and (14), except that now we write g as $g(\bar{x}, \bar{y})$ even though the variable $\bar{x}$ does not appear in the expression for g. It can be shown that $\bar{y}$ admits exactly

the same expression as in APLM-S, whereas $\bar{x}$ is obtained by a gradient step, $x - \rho\nabla_x f(x,y)$. To obtain x, we solve the linear system

$$\left(A^TA + \frac{1}{\mu+\rho}D + \frac{1}{\rho}I\right)x = r_x + \frac{\rho}{\mu+\rho}C^Tr_y, \tag{29}$$

after which y is computed by $y = \left(\frac{\mu\rho}{\mu+\rho}\right)\left(r_y + \frac{1}{\mu}Cx\right)$.

Remark 3.2. *For ALM-S, the Lipschitz constant for $\nabla f(x,y)$ $L_f = \lambda_{max}(A^TA) + \frac{1}{\mu}d_{max}$, where $d_{max} = \max_i D_{ii} \geq 1$. For the complexity results in [19] to hold, we need $\rho \leq \frac{1}{L_f}$. Since $\lambda_{max}(A^TA)$ is usually not known, it is necessary to perform a backtracking line-search on ρ to ensure that $F(x^{k+1}, y^{k+1}) \leq \mathcal{L}_\rho(x^{k+1}, y^{k+1}, \bar{x}^k, \bar{y}^k, \gamma^k)$. In practice, we adopted the following continuation scheme instead. We initially set $\rho = \rho_0 = \frac{\mu}{d_{max}}$ and decreased ρ by a factor of β after a given number of iterations until ρ reached a user-supplied minimum value ρ_{min}. This scheme prevents ρ from being too small, and hence negatively impacting computational performance. However, in both cases the left-hand-side of the system* (29) *has to be re-factorized every time ρ is updated.*

As we have seen above, the Lipschitz constant resulting from splitting both x and y is potentially much larger than $\frac{1}{\mu}$. Hence, partial-linearization reduces the Lipschitz constant and hence improves the bound on the right-hand-side of (23) and allows Algorithm 3.2 to take larger step sizes (equal to μ). Compared to ALM-S, solving for x in the skipping step (Line 6) becomes harder. Intuitively, APLM-S does a better job of 'load-balancing' by managing a better trade-off between the hardness of the subproblems and the practical convergence rate.

3.3 ISTA: partial linearization (ISTA-p)

We can also minimize the augmented Lagrangian (6), which we write as $\mathcal{L}(x,y,v) = f(x,y) + g(y)$ with $f(x,y)$ and $g(y)$ defined as in (13) and (14), using a variant of ISTA that only linearizes $f(x,y)$ with respect to the y variables. As in Section 3.2, we can set $\rho = \mu$ and guarantee the convergence properties of ISTA-p (see Corollary 3.1 below). Formally, let (x,y) be the current iterate and (x^+, y^+) be the next iterate. We compute y^+ by

$$\begin{aligned} y^+ &= \arg\min_{y'} \mathcal{L}_\rho(x,y,y',\nabla_y f(x,y)) &(30)\\ &= \arg\min_{y'} \left\{ \frac{1}{2\mu}\sum_j (\|y'_j - d_{y_j}\|^2 + \lambda\|y'_j\|) \right\}, &(31) \end{aligned}$$

where $d_y = Cx - \mu v$. Hence the solution y^+ to problem (31) is given blockwise by $T([d_y]_j, \mu\lambda), j = 1, \cdots, J$.

Now given y^+, we solve for x^+ by

$$\begin{aligned} x^+ &= \arg\min_{x'} f(x', y^+)\\ &= \arg\min_{x'} \left\{ \frac{1}{2}\|Ax' - b\|^2 - v^T(Cx' - y^+) + \frac{1}{2\mu}\|Cx' - y^+\|^2 \right\} &(32) \end{aligned}$$

The algorithm that implements subroutine ApproxAugLagMin(x,y,v) in Algorithm 2.2 by ISTA with partial linearization is stated below as Algorithm 3.3.

As we remarked in Section 3.2, Algorithm 3.3 is equivalent to Algorithm 3.2 (APLM-S) where every iteration is a skipping iteration. Hence, we have from Theorem 3.1.

Algorithm 3.3 ISTA-p (partial linearization)

Given $x^0, \bar{y}^0, v$. Choose ρ. Define $f(x, y)$ as in (13).
for $k = 0, 1, \cdots$ until stopping criterion is satisfied **do**
  $x^{k+1} \leftarrow \arg\min_x f(x; \bar{y}^k)$
  $\bar{y}^{k+1} \leftarrow \arg\min_y \mathcal{L}_\rho(x^{k+1}, \bar{y}^k, y, \nabla_y f(x^{k+1}, \bar{y}^k))$
end for
return $(x^{K+1}, \bar{y}^{K+1})$

Corollary 3.1. *Assume $\nabla_y f(\cdot, \cdot)$ is Lipschitz continuous with Lipschitz constant $L_y(f)$. For $\rho \leq \frac{1}{L_y(f)}$, the iterates $(x^k, \bar{y}^k)$ in Algorithm 3.3 satisfy*

$$F(x^k, \bar{y}^k) - F(x^*, y^*) \leq \frac{\|\bar{y}^0 - y^*\|^2}{2\rho k}, \quad \forall k, \tag{33}$$

where (x^*, y^*) is an optimal solution to (12).

It is easy to see that (31) is equivalent to (8), and that (32) is the same as (9) in ADAL.

Remark 3.3. *We have shown that with a fixed v, the ISTA-p iterations are exactly the same as the ADAL iterations. The difference between the two algorithms is that ADAL updates the (outer) Lagrange multiplier v in each iteration, while in ISTA-p, v stays the same throughout the inner iterations. We can thus view ISTA-p as a variant of ADAL with delayed updating of the Lagrange multiplier.*

The 'load-balancing' behavior discussed in Section 3.2 is more obvious for ISTA-p. As we will see in Section 3.5, if we apply ISTA (with full linearization) to minimize (6), solving for x is simply a gradient step. Here, we need to minimize $f(x, y)$ with respect to x exactly, while being able to take larger step sizes in the other subproblem, due to the smaller associated Lipschitz constant.

3.4 FISTA-p

We now present an accelerated version FISTA-p of ISTA-p. FISTA-p is a special case of FAPLM-S with a skipping step occurring in every iteration.We state the algorithm formally as Algorithm 3.4. The iteration complexity of FISTA-p (and FAPLM-S) is given by the following theorem.

Algorithm 3.4 FISTA-p (partial linearization)

Given $x^0, \bar{y}^0, v$. Choose ρ, and $z^0 = \bar{y}^0$. Define $f(x, y)$ as in (13).
for $k = 0, 1, \cdots, K$ **do**
  $x^{k+1} \leftarrow \arg\min_x f(x; z^k)$
  $\bar{y}^{k+1} \leftarrow \arg\min_y \mathcal{L}_\rho(x^{k+1}, z^k, y, \nabla_y f(x^{k+1}, z^k))$
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$
  $z^{k+1} \leftarrow \bar{y}^{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\bar{y}^{k+1} - \bar{y}^k)$
end for
return $(x^{K+1}, \bar{y}^{K+1})$

Theorem 3.2. *Assuming that $\nabla_y f(\cdot)$ is Lipschitz continuous with Lipschitz constant $L_y(f)$ and $\rho \le \frac{1}{L_y(f)}$, the sequence $\{x^k, \bar{y}^k\}$ generated by Algorithm 3.4 satisfies*

$$F(x^k, \bar{y}^k) - F(x^*, y^*) \le \frac{2\|\bar{y}^0 - y^*\|^2}{\rho(k+1)^2}, \tag{34}$$

Although we need to solve a linear system in every iteration of Algorithms 3.2, 3.3, and 3.4, the left-hand-side of the system stays constant throughout the invocation of the algorithms because, following Remark 3.1, we can always set $\rho = \mu$. Hence, no line-search is necessary, and this step essentially requires only one backward- and one forward-substitution, the complexity of which is the same as a gradient step.

3.5 ISTA/FISTA: full linearization

ISTA solves the following problem in each iteration to produce the next iterate $\begin{pmatrix} x^+ \\ y^+ \end{pmatrix}$.

$$\begin{aligned} \min_{x',y'} \quad & \frac{1}{2\rho}\left\| \begin{pmatrix} x' \\ y' \end{pmatrix} - d \right\|^2 + \lambda \sum_s \|y_s\| \\ \equiv \quad & \frac{1}{2\rho}\|x' - d_x\|^2 + \sum_j \frac{1}{2\rho}\left(\|y'_j - d_{y_j}\|^2 + \lambda\|y'_j\|\right), \end{aligned} \tag{35}$$

where $d = \begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \rho\nabla f(x,y)$, and $f(x,y)$ is defined in (13). It is easy to see that we can solve for x^+ and y^+ separately in (35). Specifically,

$$x^+ = d_x \tag{36}$$

$$y_j^+ = \frac{d_{y_j}}{\|d_{y_j}\|}\max(0, \|d_{y_j}\| - \lambda\rho), \quad j = 1, \ldots, J. \tag{37}$$

Using ISTA to solve the outer augmented Lagrangian (6) subproblem is equivalent to taking only skipping steps in ALM-S. In our experiments, we used the accelerated version of ISTA, i.e. FISTA (Algorithm 3.5) to solve (6).

FISTA (resp. ISTA) is, in fact, an inexact version of FISTA-p (resp. ISTA-p), where we minimize with respect to x a linearized approximation

$$\tilde{f}(x, z^k) := f(x^k, z^k) + \nabla_x f(x^k, z^k)(x - x^k) + \frac{1}{2\rho}\|x - x^k\|^2$$

of the quadratic objective function $f(x, z^k)$ in (32). The update to x in Line 3 of Algorithm 3.4 is replaced by (36) as a result. Similar to FISTA-p, FISTA is also a special skipping version of the full-split FALM-S. Considering that FISTA has an iteration complexity of $O(\frac{1}{k^2})$, it is not surprising that FISTA-p has the same iteration complexity.

Remark 3.4. *Since FISTA requires only the gradient of $f(x,y)$, it can easily handle any smooth convex loss function, such as the logistic loss for binary classification, $L(x) = \sum_{i=1}^N \log(1 + \exp(-b_i a_i^T x))$, where a_i^T is the i-th row of A, and b is the vector of labels. Moreover, when the scale of the data ($\min\{n, m\}$) is so large that it is impractical to compute the Cholesky factorization of $A^T A$, FISTA is a good choice to serve as the subroutine ApproxAugLagMin(x, y, v) in OGLasso-AugLag.*

Algorithm 3.5 FISTA

Given $\bar{x}^0, \bar{y}^0, v$. Choose ρ^0. Set $t_0 = 0, z_x^0 = \bar{x}^0, z_y^0 = \bar{y}^0$. Define $f(x, y)$ as in (13).
for $k = 0, 1, \cdots$ until stopping criterion is satisfied **do**
  Perform a backtracking line-search on ρ, starting from ρ_0.
  $\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} z_x^k \\ z_y^k \end{pmatrix} - \rho \nabla f(z_x^k, z_y^k)$
  $\bar{x}^{k+1} \leftarrow d_x$
  $\bar{y}_j^{k+1} \leftarrow \frac{d_{y_j}}{\|d_{y_j}\|} \max(0, \|d_{y_j}\| - \lambda\rho), \quad j = 1, \ldots, J.$
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$
  $z_x^{k+1} \leftarrow \bar{x}^k + \frac{t_k - 1}{t_{k+1}}(\bar{x}^{k+1} - \bar{x}^k)$
  $z_y^{k+1} \leftarrow \bar{y}^k + \frac{t_k - 1}{t_{k+1}}(\bar{y}^{k+1} - \bar{y}^k)$
end for
return $(\bar{x}^{K+1}, \bar{y}^{K+1})$

4 Overlapping Group l_1/l_∞-Regularization

The subproblems with respect to y (or $\bar{y}$) involved in all the algorithms presented in the previous sections take the following form

$$\min_y \frac{1}{2\rho}\|c - y\|^2 + \tilde{\Omega}(y), \tag{38}$$

where $\tilde{\Omega}(y) = \lambda \sum_{s \in \tilde{\mathcal{S}}} w_s \|y_s\|_\infty$ in the case of l_1/l_∞-regularization. In (8), for example, $c = Cx - \mu v$. The solution to (38) is the proximal operator of $\tilde{\Omega}$ [13, 12]. Similar to the classical Group Lasso, this problem is block-separable and hence all blocks can be solved simultaneously.

Again, for notational simplicity, we assume $w_s = 1 \quad \forall s \in \tilde{\mathcal{S}}$ and omit it from now on. For each $s \in \tilde{\mathcal{S}}$, the subproblem in (38) is of the form

$$\min_{y_s} \quad \frac{1}{2}\|c_s - y_s\|^2 + \rho\lambda\|y_s\|_\infty. \tag{39}$$

As shown in [48], the optimal solution to the above problem is $c_s - P(c_s)$, where P denotes the orthogonal projector onto the ball of radius $\rho\lambda$ in the dual norm of the l_∞-norm, i.e. the l_1-norm. The Euclidean projection onto the simplex can be computed in (expected) linear time [14, 9]. Duchi et al. [14] show that the problem of computing the Euclidean projection onto the l_1-ball can be reduced to that of finding the Euclidean projection onto the simplex in the following way. First, we replace c_s in problem (39) by $|c_s|$, where the absolute value is taken component-wise. After we obtain the projection z_s onto the simplex, we can construct the projection onto the l_1-ball by setting $y_s^* = sign(c_s)z_s$, where $sign(\cdot)$ is also taken component-wise.

5 Experiments

We tested the OGLasso-AugLag framework (Algorithm 2.2) with four subroutines: ADAL, APLM-S, FISTA-p, and FISTA. We implemented the framework with the first three subroutines in C++ to compare them with the ProxFlow algorithm proposed in [30]. We used the C interface and BLAS and LAPACK subroutines provided by the AMD Core Math Library (ACML)[2]. To compare

[2]http://developer.amd.com/libraries/acml/pages/default.aspx. Ideally, we should have used the Intel Math Kernel Library (Intel MKL), which is optimized for Intel processors, but Intel MKL is not freely available.

Algorithm	Outer rel. dual residual s^{l+1}	Inner iteration	
		Rel. primal residual	Rel. objective gradient residual
ADAL	$\frac{\|C^T(y^{l+1}-y^l)\|}{\|C^T y^l\|}$	-	-
FISTA-p	$\frac{\|C^T(\bar{y}^{K+1}-z^K)\|}{\|C^T z^K\|}$	$\frac{\|\bar{y}^{k+1}-z^k\|}{\|z^k\|}$	$\frac{\|C^T(\bar{y}^{k+1}-z^k)\|}{\|C^T z^k\|}$
APLM-S	$\frac{\|C^T(\bar{y}^{K+1}-y^{K+1})\|}{\|C^T y^{K+1}\|}$	$\frac{\|\bar{y}^{k+1}-y^{k+1}\|}{\|y^{k+1}\|}$	$\frac{\|C^T(\bar{y}^{k+1}-y^{k+1}\|)}{\|C^T y^{k+1}\|}$
FISTA	$\frac{\left\|\begin{pmatrix}\bar{x}^{K+1}\\ \bar{y}^{K+1}\end{pmatrix}-\begin{pmatrix}z_x^K\\ z_y^K\end{pmatrix}\right\|}{\left\|\begin{pmatrix}z_x^K\\ z_y^K\end{pmatrix}\right\|}$	$\frac{\left\|\begin{pmatrix}\bar{x}^{k+1}\\ \bar{y}^{k+1}\end{pmatrix}-\begin{pmatrix}z_x^k\\ z_y^k\end{pmatrix}\right\|}{\left\|\begin{pmatrix}z_x^k\\ z_y^k\end{pmatrix}\right\|}$	$\frac{\left\|\begin{pmatrix}\bar{x}^{k+1}\\ \bar{y}^{k+1}\end{pmatrix}-\begin{pmatrix}z_x^k\\ z_y^k\end{pmatrix}\right\|}{\left\|\begin{pmatrix}z_x^k\\ z_y^k\end{pmatrix}\right\|}$

Table 1: Specification of the quantities used in the outer and inner stopping criteria.

with ProxGrad [11], we implemented the framework and all four algorithms in Matlab. We did not include ALM-S in our experiments because it is time-consuming to find the right ρ for the inner loops as discussed in Remark 3.2, and our preliminary computational experience showed that ALM-S was slower than the other algorithms, even when the heuristic ρ-setting scheme discussed in Remark 3.2 was used, because a large number of steps were skipping steps, which meant that the computation involved in solving the linear systems in those steps was wasted. All of our experiments were performed on a laptop PC with an Intel Core 2 Duo 2.0 GHz processor and 4 Gb of memory.

5.1 Algorithm parameters and termination criteria

Each algorithm (framework + subroutine)[3] required several parameters to be set and termination criteria to be specified. We used stopping criteria based on the primal and dual residuals as in [8]. We specify the criteria for each of the algorithms below, but defer their derivation to Appendix C. The maximum number of outer iterations was set to 500, and the tolerance for the outer loop was set at $\epsilon_{out} = 10^{-4}$. The number of inner-iterations was capped at 2000, and the tolerance at the l-th outer iteration for the inner loops was ϵ_{in}^l. Our termination criterion for the outer iterations was

$$\max\{r^l, s^l\} \leq \epsilon_{out}, \tag{40}$$

where $r^l = \frac{\|Cx^l - y^l\|}{\max\{\|Cx^l\|, \|y^l\|\}}$ is the outer relative primal residual and s^l is the relative dual residual, which is given for each algorithm in Table 1. Recall that $K+1$ is the index of the last inner iteration of the l-th outer iteration; for example, for APLM-S, (x^{l+1}, y^{l+1}) takes the value of the last inner iterate $(x^{K+1}, \bar{y}^{K+1})$. We stopped the inner iterations when the maximum of the relative primal residual and the relative objective gradient for the inner problem was less than ϵ_{in}^l. (See Table 1 for the expressions of these two quantities.) We see there that s^{l+1} can be obtained directly from the relative gradient residual computed in the last inner iteration of the l-th outer iteration.

We set $\mu_0 = 0.01$ in all algorithms except that we set $\mu_0 = 0.1$ in ADAL for the data sets other than the first synthetic set and the breast cancer data set. We set $\rho = \mu$ in FISTA-p and APLM-S and $\rho_0 = \mu$ in FISTA.

For Theorem 2.1 to hold, the solution returned by the function ApproxAugLagMin(x, y, v) has to become increasingly more accurate over the outer iterations. However, it is not possible to

[3]For conciseness, we use the subroutine names (e.g. FISTA-p) to represent the full algorithms that consist of the OGLasso-AugLag framework and the subroutines.

evaluate the sub-optimality quantity α^l in (7) exactly because the optimal value of the augmented Lagrangian $\mathcal{L}(x, y, v^l)$ is not known in advance. In our experiments, we used the maximum of the relative primal and dual residuals ($\max\{r^l, s^l\}$) as a surrogate to α^l for two reasons: First, it has been shown in [8] that r^l and s^l are closely related to α^l. Second, the quantities r^l and s^l are readily available as bi-products of the inner and outer iterations. To ensure that the sequence $\{\epsilon_{in}^l\}$ satisfies (7), we basically set:

$$\epsilon_{in}^{l+1} = \beta_{in}\epsilon_{in}^l, \tag{41}$$

with $\epsilon_{in}^0 = 0.01$ and $\beta_{in} = 0.5$. However, since we terminate the outer iterations at $\epsilon_{out} > 0$, it is not necessary to solve the subproblems to an accuracy much higher than the one for the outer loop. On the other hand, it is also important for ϵ_{in}^l to decrease to below ϵ_{out}, since s^l is closely related to the quantities involved in the inner stopping criteria. Hence, we slightly modified (41) and used $\epsilon_{in}^{l+1} = \max\{\beta_{in}\epsilon_{in}^l, 0.2\epsilon_{out}\}$.

Recently, we became aware of an alternative 'relative error' stopping criterion [16] for the inner loops, which guarantees convergence of Algorithm 2.2. In our context, this criterion essentially requires that the absolute dual residual is less than a fraction of the absolute primal residual. For FISTA-p, for instance, this condition requires that the $(l+1)$-th iterate satisfies

$$2\left\|\begin{pmatrix} w_x^0 - x^{l+1} \\ w_y^l - y^{l+1} \end{pmatrix}\right\| \bar{s}^{l+1} + \frac{(\bar{s}^{l+1})^2}{\mu^2} \leq \sigma(\bar{r}^{l+1})^2,$$

where $\bar{r}$ and $\bar{s}$ are the numerators in the expressions for r and s respectively, $\sigma = 0.99$, w_x^0 is a constant, and w_y is an auxiliary variable updated in each outer iteration by $w_y^{l+1} = w_y^l - \frac{1}{\mu^2}C^T(\bar{y}^{K+1} - z^K)$. We experimented with this criterion but did not find any computational advantage over the heuristic based on the relative primal and dual residuals.

5.2 Strategies for updating μ

The penalty parameter μ in the outer augmented Lagrangian (6) not only controls the infeasibility in the constraint $Cx = y$, but also serves as the step-length in the y-subproblem (and the x-subproblem in the case of FISTA). We adopted two kinds of strategies for updating μ. The first one simply kept μ fixed. In this case, choosing an appropriate μ_0 was important for good performance. This was especially true for ADAL in our computational experiments. Usually, a μ_0 in the range of 10^{-1} to 10^{-3} worked well.

The second strategy is a dynamic scheme based on the values r^l and s^l [8]. Since $\frac{1}{\mu}$ penalizes the primal infeasibility, a small μ tends to result in a small primal residual. On the other hand, a large μ tends to yield a small dual residual. Hence, to keep r^l and s^l approximately balanced in each outer iteration, our scheme updated μ as follows:

$$\mu^{l+1} \leftarrow \begin{cases} \max\{\beta\mu^l, \mu_{min}\}, & \text{if } r^l > \tau s^l \\ \min\{\mu^l/\beta, \mu_{max}\}, & \text{if } s^l > \tau r^l \\ \mu^l, & \text{otherwise,} \end{cases} \tag{42}$$

where we set $\mu_{max} = 10$, $\mu_{min} = 10^{-6}$, $\tau = 10$ and $\beta = 0.5$, except for the first synthetic data set, where we set $\beta = 0.1$ for ADAL, FISTA-p, and APLM-S.

5.3 Synthetic examples

To compare our algorithms with the ProxGrad algorithm proposed in [11], we first tested a synthetic data set (ogl) using the procedure reported in [11] and [24]. The sequence of decision variables x

were arranged in groups of ten, with adjacent groups having an overlap of three variables. The support of x was set to the first half of the variables. Each entry in the design matrix A and the non-zero entries of x were sampled from i.i.d. standard Gaussian distributions, and the output b was set to $b = Ax + \epsilon$, where the noise $\epsilon \sim \mathcal{N}(0, I)$. Two sets of data were generated as follows: (a) Fix $n = 5000$ and vary the number of groups J from 100 to 1000 with increments of 100. (b) Fix $J = 200$ and vary n from 1000 to 10000 with increments of 1000. The stopping criterion for ProxGrad was the same as the one used for FISTA, and we set its smoothing parameter to 10^{-3}. Figure 1 plots the CPU times taken by the Matlab version of our algorithms and ProxGrad (also in Matlab) on theses scalability tests on l_1/l_2-regularization. A subset of the numerical results on which these plots are based is presented in Tables 4 and 5.

The plots clearly show that the alternating direction methods were much faster than ProxGrad on these two data sets. Compared to ADAL, FISTA-p performed slightly better, while it showed obvious computational advantage over its general version APLM-S. In the plot on the left of Figure 1, FISTA exhibited the advantage of a gradient-based algorithm when both n and m are large. In that case (towards the right end of the plot), the Cholesky factorizations required by ADAL, APLM-S, and FISTA-p became relatively expensive. When $\min\{n, m\}$ is small or the linear systems can be solved cheaply, as the plot on the right shows, FISTA-p and ADAL have an edge over FISTA due to the smaller numbers of inner iterations required.

We generated a second data set (dct) using the approach from [30] for scalability tests on both the l_1/l_2 and l_1/l_∞ group penalties. The design matrix A was formed from over-complete dictionaries of discrete cosine transforms (DCT). The set of groups were all the contiguous sequences of length five in one-dimensional space. x had about 10% non-zero entries, selected randomly. We generated the output as $b = Ax + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.01\|Ax\|^2)$. We fixed $n = 1000$ and varied the number of features m from 5000 to 30000 with increments of 5000. This set of data leads to considerably harder problems than the previous set because the groups are heavily overlapping, and the DCT dictionary-based design matrix exhibits local correlations. Due to the excessive running time required on Matlab, we ran the C++ version of our algorithms for this data set, leaving out APLM-S and ProxGrad, whose performance compared to the other algorithms is already fairly clear from Figure 1. For ProxFlow, we set the tolerance on the relative duality gap to 10^{-4}, the same as ϵ_{out}, and kept all the other parameters at their default values.

Figure 2 presents the CPU times required by the algorithms versus the number of features. In the case of l_1/l_2-regularization, it is clear that FISTA-p outperformed the other two algorithms. For l_1/l_∞-regularization, ADAL and FISTA-p performed equally well and compared favorably to ProxFlow. In both cases, the growth of the CPU times for FISTA follows the same trend as that for FISTA-p, and they required a similar number of outer iterations, as shown in Tables 6 and 7. However, FISTA lagged behind in speed due to larger numbers of inner iterations. Unlike in the case of the ogl data set, Cholesky factorization was not a bottleneck for FISTA-p and ADAL here because we needed to compute it only once.

To simulate the situation where computing or caching A^TA and its Cholesky factorization is not feasible, we switched ADAL and FISTA-p to PCG mode by always using PCG to solve the linear systems in the subproblems. We compared the performance of ADAL, FISTA-p, and FISTA on the previous data set for both l_1/l_2 and l_1/l_∞ models. The results for ProxFlow are copied from from Figure 2 and Table 9 to serve as a reference. We experimented with the fixed-value and the dynamic updating schemes for μ on all three algorithms. From Figure 3, it is clear that the performance of FISTA-p was significantly improved by using the dynamic scheme. For ADAL, however, the dynamic scheme worked well only in the l_1/l_2 case, whereas the performance turned worse in general in the l_1/l_∞ case. We did not include the results for FISTA with the dynamic scheme because the solutions obtained were considerably more suboptimal than the ones obtained

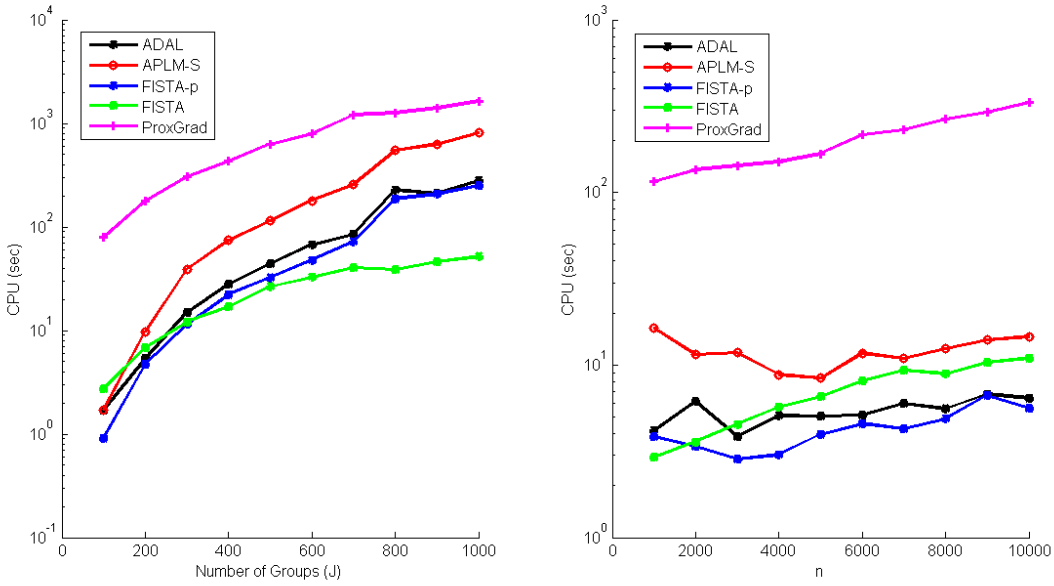

Figure 1: Scalability test results of the algorithms on the synthetic overlapping Group Lasso data sets from [11]. The scale of the y-axis is logarithmic. The dynamic scheme for μ was used for all algorithms except ProxGrad.

with the fixed-μ scheme. Tables 8 and 9 report the best results of the algorithms in each case. The plots and numerical results show that FISTA-p compares favorably to ADAL and stays competitive to ProxFlow. In terms of the quality of the solutions, FISTA-p and ADAL also did a better job than FISTA, as evidenced in Table 9. On the other hand, the gap in CPU time between FISTA and the other three algorithms is less obvious.

5.4 Real-world Examples

To demonstrate the practical usefulness of our algorithms, we tested our algorithms on two real-world applications.

5.4.1 Breast Cancer Gene Expressions

We used the breast cancer data set [47] with canonical pathways from MSigDB [44]. The data was collected from 295 breast cancer tumor samples and contains gene expression measurements for 8,141 genes. The goal was to select a small set of the most relevant genes that yield the best prediction performance. A detailed description of the data set can be found in [11, 24]. In our experiment, we performed a regression task to predict the length of survival of the patients. The canonical pathways naturally provide grouping information of the genes. Hence, we used them as the groups for the group-structured regularization term $\Omega(\cdot)$.

Table 2 summarizes the data attributes. The numerical results for the l_1/l_2-norm are collected in Table 10, which show that FISTA-p and ADAL were the fastest on this data set. Again, we had to tune ADAL with different initial values (μ_0) and updating schemes of μ for speed and quality of the solution, and we eventually kept μ constant at 0.01. The dynamic updating scheme for μ also did not work for FISTA, which returned a very suboptimal solution in this case. We instead

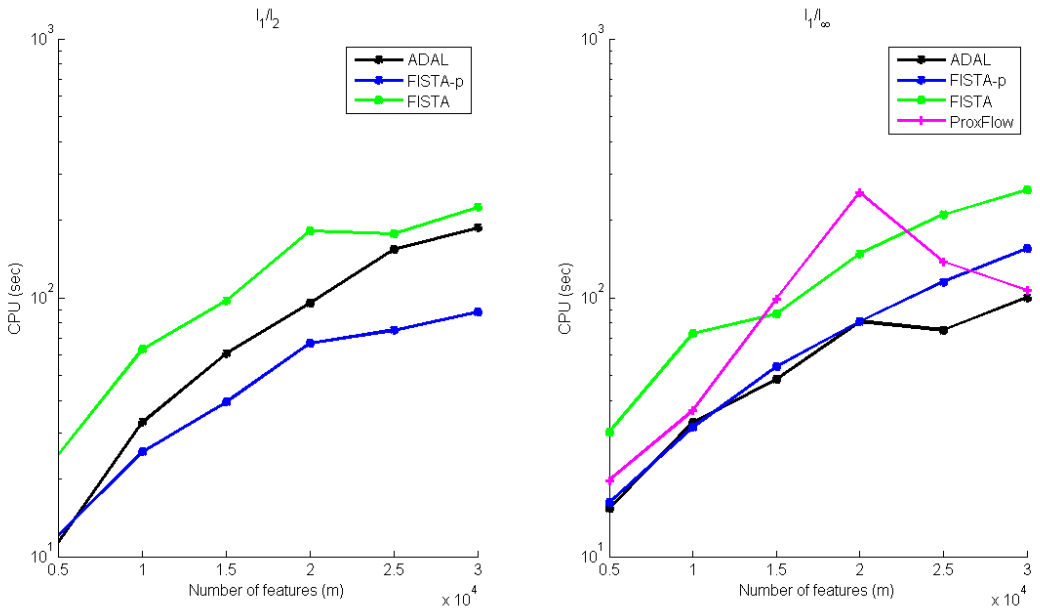

Figure 2: Scalability test results on the DCT set with l_1/l_2-regularization (left column) and l_1/l_∞-regularization (right column). The scale of the y-axis is logarithmic. All of FISTA-p, FITSA, and ADAL were run with a fixed $\mu = \mu_0$.

Data sets	N (no. samples)	J (no. groups)	group size	average frequency
BreastCancerData	295	637	23.7 (avg)	4

Table 2: The Breast Cancer Dataset

adopted a simple scheme of decreasing μ by half every 10 outer iterations. Figure 6 graphically depicts the performance of the different algorithms. In terms of the outer iterations, APLM-S behaved identically to FISTA-p, and FISTA also behaved similarly to ADAL. However, APLM-S and FISTA were considerably slower due to larger numbers of inner iterations.

We plot the root-mean-squared-error (RMSE) over different values of λ (which lead to different numbers of active genes) in the left half of Figure 4. The training set consists of 200 randomly selected samples, and the RMSE was computed on the remaining 95 samples. l_1/l_2-regularization achieved lower RMSE in this case. However, l_1/l_∞-regularization yielded better group sparsity as shown in Figure 5. The sets of active genes selected by the two models were very similar as illustrated in the right half of Figure 4. In general, the magnitudes of the coefficients returned by l_1/l_∞-regularization tended to be similar within a group, whereas those returned by l_1/l_2-regularization did not follow that pattern. This is because l_1/l_∞-regularization penalizes only the maximum element, rather than all the coefficients in a group, resulting in many coefficients having the same magnitudes.

5.4.2 Video Sequence Background Subtraction

We next considered the video sequence background subtraction task from [30, 23]. The main objective here is to segment out foreground objects in an image (frame), given a sequence of m

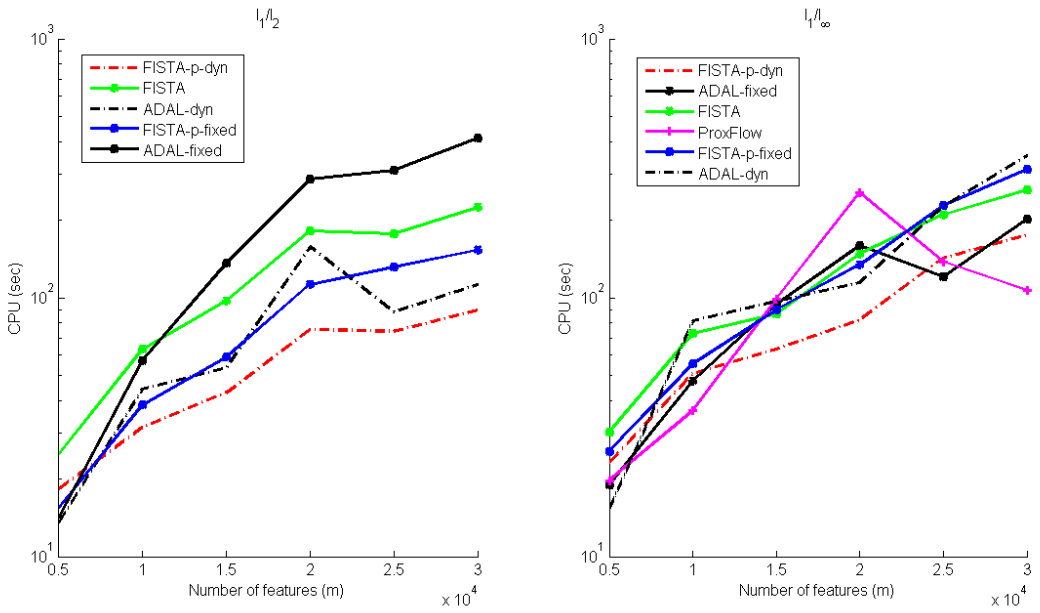

Figure 3: Scalability test results on the DCT set with l_1/l_2-regularization (left column) and l_1/l_∞-regularization (right column). The scale of the y-axis is logarithmic. FISTA-p and ADAL are in PCG mode. The dotted lines denote the results obtained with the dynamic updating scheme for μ.

frames from a fixed camera. The data used in this experiment is available online [4] [46]. The basic setup of the problem is as follows. We represent each frame of n pixels as a column vector $A_j \in \mathbb{R}^n$ and form the matrix $A \in \mathbb{R}^{n\times m}$ as $A \equiv \begin{pmatrix} A_1 & A_2 & \cdots & A_m \end{pmatrix}$. The test frame is represented by $b \in \mathbb{R}^n$. We model the relationship between b and A by $b \approx Ax + e$, where x is assumed to be sparse, and e is the 'noise' term which is also assumed to be sparse. Ax is thus a sparse linear combination of the video frame sequence and accounts for the background present in both A and b. e contains the sparse foreground objects in b. The basic model with l_1-regularization (Lasso) is

$$\min_{x,e} \frac{1}{2}\|Ax + e - b\|^2 + \lambda(\|x\|_1 + \|e\|_1). \tag{43}$$

It has been shown in [30] that we can significantly improve the quality of segmentation by applying a group-structured regularization $\Omega(\cdot)$ on e, where the groups are all the overlapping $k \times k$-square patches in the image. Here, we set $k = 3$. The model thus becomes

$$\min_{x,e} \frac{1}{2}\|Ax + e - b\|^2 + \lambda(\|x\|_1 + \|e\|_1 + \Omega(e)). \tag{44}$$

Note that (44) still fits into the group-sparse framework if we treat the l_1-regularization terms as the sum of the group norms, where the each groups consists of only one element.

We also considered an alternative model, where a Ridge regularization is applied to x and an Elastic-Net penalty [50] to e. This model

$$\min_{x,e} \frac{1}{2}\|Ax + e - b\|^2 + \lambda_1\|e\|_1 + \lambda_2(\|x\|^2 + \|e\|^2) \tag{45}$$

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

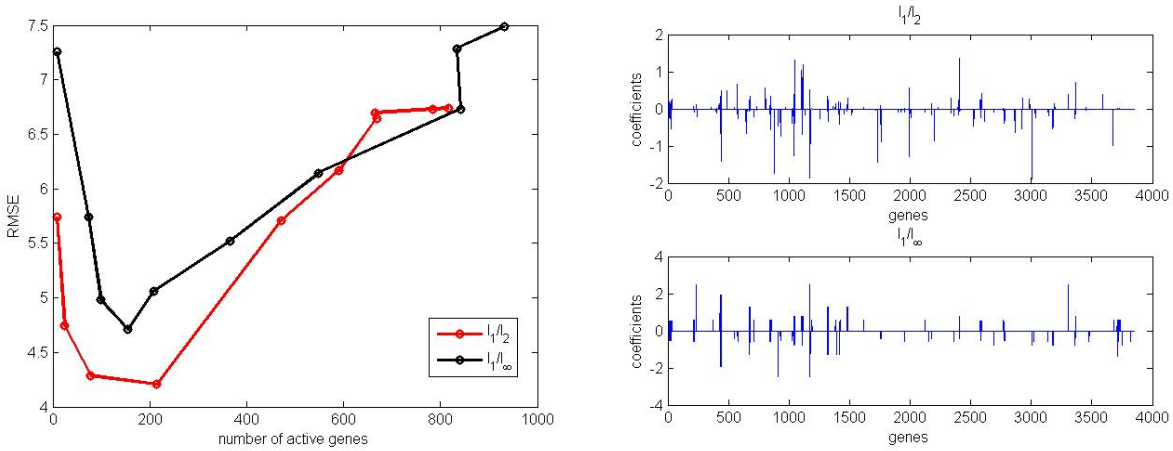

Figure 4: On the left: Plot of root-mean-squared-error against the number of active genes for the Breast Cancer data. The plot is based on the regularization path for ten different values for λ. The total CPU time (in Matlab) using FISTA-p was 51 seconds for l_1/l_2-regularization and 115 seconds for l_1/l_∞-regularization. On the right: The recovered sparse gene coefficients for predicting the length of the survival period. The value of λ used here was the one minimizing the RMSE in the plot on the left.

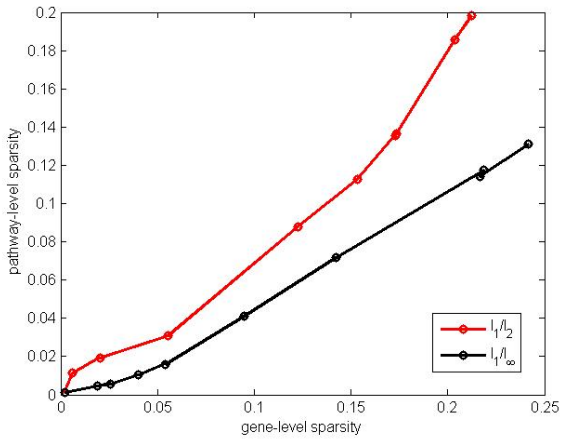

Figure 5: Pathway-level sparsity v.s. Gene-level sparsity.

does not yield a sparse x, but sparsity in x is not a crucial factor here. It is, however, well suited for our partial linearization methods (APLM-S and FISTA-p), since there is no need for the augmented Lagrangian framework. Of course, we can also apply FISTA to solve (45).

We recovered the foreground objects by solving the above optimization problems and applying the sparsity pattern of e as a mask for the original test frame. A hand-segmented evaluation image from [46] served as the ground truth. The regularization parameters λ, λ_1, and λ_2 were selected in such a way that the recovered foreground objects matched the ground truth to the maximum extent.

FISTA-p was used to solve all three models. The l_1 model (43) was treated as a special case of the group regularization model (44), with each group containing only one component of the feature

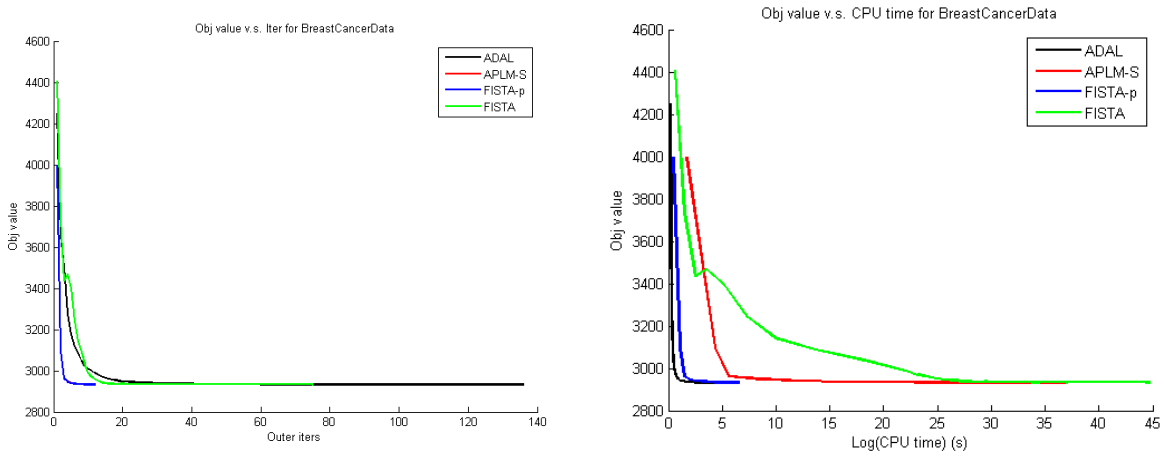

Figure 6: Objective values v.s. Outer iters and Objective values v.s. CPU time plots for the Breast Cancer data. The results for ProxGrad are not plotted due to the different objective function that it minimizes. The red (APLM-S) and blue (FISTA-p) lines overlap in the left column.

vector. [5] For the Ridge/Elastic-Net penalty model, we applied FISTA-p directly without the outer augmented Lagrangian layer.

The solutions for the $l_1/l_2, l_1/l_\infty$, and Lasso models were not strictly sparse in the sense that those supposedly zero feature coefficients had non-zero (albeit extremely small) magnitudes, since we enforced the linear constraints $Cx = y$ through an augmented Lagrangian approach. To obtain sparse solutions, we truncated the non-sparse solutions using thresholds ranging from 10^{-9} to 10^{-3} and selected the threshold that yielded the best accuracy.

Note that because of the additional feature vector e, the data matrix is effectively $\tilde{A} = \begin{pmatrix} A & I_n \end{pmatrix} \in \mathbb{R}^{n\times(m+n)}$. For solving (44), FISTA-p has to solve the linear system

$$\begin{pmatrix} A^TA + \frac{1}{\mu}D_x & A^T \\ A & I_n + \frac{1}{\mu}D_e \end{pmatrix} \begin{pmatrix} x \\ e \end{pmatrix} = \begin{pmatrix} r_x \\ r_e \end{pmatrix}, \tag{46}$$

where D is a diagonal matrix, and D_x, D_e, r_x, r_e are the components of D and r corresponding to x and e respectively. In this example, n is much larger than m, e.g. $n = 57600, m = 200$. To avoid solving a system of size $n \times n$, we took the Schur complement of $I_n + \frac{1}{\mu}D_e$ and solved instead the positive definite $m \times m$ system

$$\left(A^TA + \frac{1}{\mu}D_x - A^T(I + \frac{1}{\mu}D_e)^{-1}A\right)x = r_x - A^T(I + \frac{1}{\mu}D_e)^{-1}r_e, \tag{47}$$

$$e = diag(\mathbf{1} + \frac{1}{\mu}D_e)^{-1}(r_e - Ax). \tag{48}$$

The l_1/l_∞ model yielded the best background separation accuracy (marginally better than the l_1/l_2 model), but it also was the most computationally expensive. (See Table 3 and Figure 7.) Although the Ridge/Elastic-Net model yielded as poor separation results as the Lasso (l_1) model, it was orders of magnitude faster to solve using FISTA-p. We again observed that the dynamic scheme for μ worked better for FISTA-p than for ADAL. For a constant μ over the entire run,

[5] We did not use the original version of FISTA to solve the model as an l_1-regularization problem because it took too long to converge in our experiments due to extremely small step sizes.

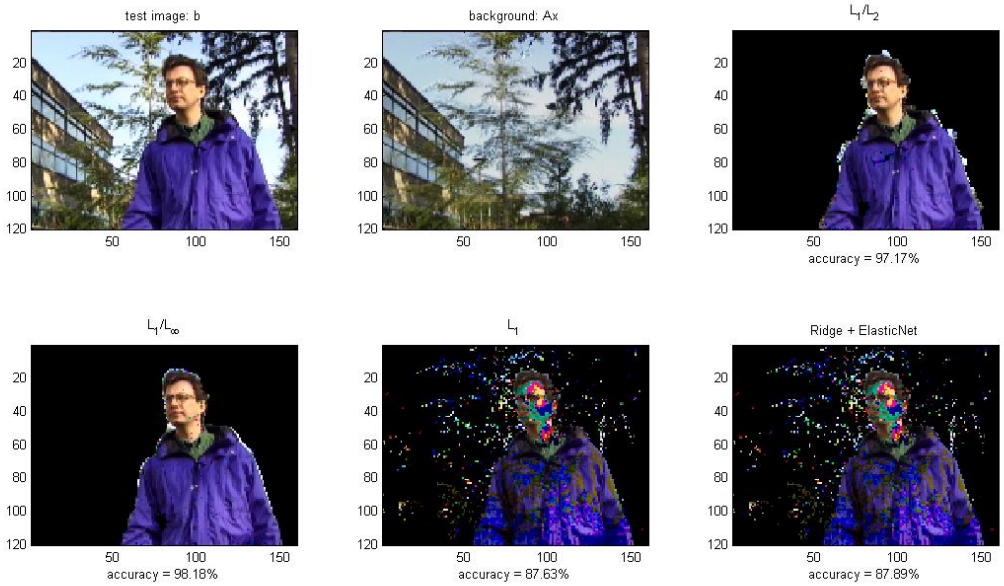

Figure 7: Separation results for the video sequence background substraction example. Each training image had 120×160 RGB pixels. The training set contained 200 images in sequence. The accuracy indicated for each of the different models is the percentage of pixels that matched the ground truth.

Model	Accuracy (percent)	Total CPU time (s)	No. parameter values on reg path
l_1/l_2	97.17	2.48e+003	8
l_1/l_∞	98.18	4.07e+003	6
l_1	87.63	1.61e+003	11
ridge + elastic net	87.89	1.82e+002	64

Table 3: Computational results for the video sequence background subtraction example. The algorithm used is FISTA-p. We used the Matlab version for the ease of generating the images. The C++ version runs at least four times faster from our experience in the previous experiments. We report the best accuracy found on the regularization path of each model. The total CPU time is recorded for computing the entire regularization path, with the specified number of different regularization parameter values.

ADAL took at least twice as long as FISTA-p to produce a solution of the same quality. A typical run of FISTA-p on this problem with the best selected λ took less than 10 outer iterations. On the other hand, ADAL took more than 500 iterations to meet the stopping criteria.

5.5 Comments on Results

The computational results exhibit two general patterns. First, the simpler algorithms (FISTA-p and ADAL) were significantly faster than the more general algorithms, such as APLM-S. Interestingly, the majority of the APLM-S inner iterations consisted of a skipping step for the tests on synthetic data and the breast cancer data, which means that APLM-S essentially behaved like ISTA-p in these cases. Indeed, FISTA-p generally required the same number of outer-iterations as APLM-

S but much fewer inner-iterations, as predicted by theory. In addition, no computational steps were wasted and no function evaluations were required for FISTA-p and ADAL. Second, FISTA-p converged faster (required less iterations) than its full-linearization counterpart FISTA. We have suggested possible reasons for this in Section 3. On the other hand, FISTA was very effective for data both of whose dimensions were large because it required only gradient computations and soft-thresholding operations, and did not require linear systems to be solved.

Our experiments showed that the performance of ADAL (as well as the quality of the solution that it returned) varied a lot as a function of the parameter settings, and it was tricky to tune them optimally. In contrast, FISTA-p exhibited fairly stable performance for a simple set of parameters that we rarely had to alter and in general performed better than ADAL.

It may seem straight-forward to apply FISTA directly to the Lasso problem (43) without the augmented Lagrangian framework.[6] However, as we have seen in our experiments, FISTA took much longer than AugLag-FISTA-p to solve this problem. We believe that this is further evidence of the 'load-balancing' property of the latter algorithm that we discussed in Section 3.2. It also demonstrates the versatility of our approach to regularized learning problems.

6 Conclusion

We have built a unified framework for solving sparse learning problems involving group-structured regularization, in particular, the l_1/l_2- or l_1/l_∞-regularization of arbitrarily overlapping groups of variables. For the key building-block of this framework, we developed new efficient algorithms based on alternating partial-linearization/splitting, with proven convergence rates. In addition, we have also incorporated ADAL and FISTA into our framework. Computational tests on several sets of synthetic test data demonstrated the relative strength of the algorithms, and through two real-world applications we compared the relative merits of these structured sparsity-inducing norms. Among the algorithms studied, FISTA-p and ADAL performed the best on most of the data sets, and FISTA appeared to be a good alternative choice for large-scale data. From our experience, FISTA-p is easier to configure and is more robust to variations in the algorithm parameters. Together, they form a flexible and versatile suite of methods for group-sparse problems of different sizes.

7 Acknowledgement

We would like to thank Katya Scheinberg and Shiqian Ma for many helpful discussions, and Xi Chen for providing the Matlab code for ProxGrad.

A Proof of Lemma 3.1

$$\begin{aligned} F(\bar{x},\bar{y}) - F(x,\bar{q}) &\geq F(\bar{x},\bar{y}) - \mathcal{L}_\rho(x,y,\bar{q},\nabla_y f(x,y)) \\ &= F(\bar{x},\bar{y}) - \Big(f(x,y) + \nabla_y f(x,y)^T(\bar{q}-y) \\ &\quad + \frac{1}{2\rho}\|\bar{q}-y\|^2 + g(\bar{q}) \Big). \end{aligned} \tag{49}$$

[6] To avoid confusion with our algorithms that consist of inner-outer iterations, we prefix our algorithms with 'AugLag' here.

From the optimality of $\bar{q}$, we also have

$$\gamma_g(\bar{q}) + \nabla_y f(x,y) + \frac{1}{\rho}(\bar{q} - y) = 0. \tag{50}$$

Since $F(x,y) = f(x,y) + g(y)$, and f and g are convex functions, for any $(\bar{x}, \bar{y})$,

$$F(\bar{x},\bar{y}) \geq g(\bar{q}) + (\bar{y} - \bar{q})^T \gamma_g(\bar{q}) + f(x,y) + (\bar{y} - y)^T \nabla_y f(x,y) + (\bar{x} - x)^T \nabla_x f(x,y). \tag{51}$$

Therefore, from (49), (50), and (51), it follows that

$$\begin{aligned} F(\bar{x},\bar{y}) - F(x,\bar{q}) &\geq g(\bar{q}) + (\bar{y} - \bar{q})^T \gamma_g(\bar{q}) + f(x,y) + (\bar{y} - y)^T \nabla_y f(x,y) + (\bar{x} - x)^T \nabla_x f(x,y) \\ &\quad - \left(f(x,y) + \nabla_y f(x,y)^T (\bar{q} - y) + \frac{1}{2\rho}\|\bar{q} - y\|^2 + g(\bar{q}) \right) \\ &= (\bar{y} - \bar{q})^T (\gamma_g(\bar{q}) + \nabla_y f(x,y)) - \frac{1}{2\rho}\|\bar{q} - y\|^2 + (\bar{x} - x)^T \nabla_x f(x,y) \\ &= (\bar{y} - \bar{q})^T \left(-\frac{1}{\rho}(\bar{q} - y) \right) - \frac{1}{2\rho}\|\bar{q} - y\|^2 + (\bar{x} - x)^T \nabla_x f(x,y) \\ &= \frac{1}{2\rho}(\|\bar{q} - \bar{y}\|^2 - \|y - \bar{y}\|^2) + (\bar{x} - x)^T \nabla_x f(x,y). \end{aligned} \tag{52}$$

The proof for the second part of the lemma is very similar, but we give it for completeness.

$$F(x,y) - F((p,q)) \geq F(x,y) - \left(f((p,q)) + g(\bar{y}) + \gamma_g(\bar{x},y)^T ((p,q)_y - \bar{y}) + \frac{1}{2\rho}\|(p,q)_y - \bar{y}\|^2 \right) \tag{53}$$

By the optimality of (p,q), we have

$$\nabla_x f((p,q)) = 0, \tag{54}$$

$$\nabla_y f((p,q)) + \gamma_g(\bar{y}) + \frac{1}{\rho}((p,q)_y - \bar{y}) = 0. \tag{55}$$

Since $F(x,y) = f(x,y) + g(y)$, it follows from the convexity of both f and g and (54) that

$$F(x,y) \geq g(\bar{y}) + (y - \bar{y})^T \gamma_g(\bar{x},y) + f((p,q)) + (y - (p,q)_y)^T \nabla_y f((p,q)). \tag{56}$$

Now combining (53), (55), and (56), it follows that

$$\begin{aligned} F(x,y) - F((p,q)) &\geq (y - (p,q)_y)^T (\gamma_g(\bar{x},y)_{\bar{y}} + \nabla_y f((p,q))) - \frac{1}{2\rho}\|(p,q)_y - \bar{y}\|^2 \\ &= (y - (p,q)_y)^T \left(\frac{1}{\rho}(\bar{y} - (p,q)_y) \right) - \frac{1}{2\rho}\|(p,q)_y - \bar{y}\|^2 \\ &= \frac{1}{2\rho}(\|(p,q)_y - y\|^2 - \|y - \bar{y}\|^2). \end{aligned} \tag{57}$$

B Proof of Theorem 3.1

Let I be the set of all regular iteration indices among the first $k-1$ iterations, and let I_c be its complement. For all $n \in I_c, y^{n+1} = \bar{y}^n$.

For $n \in I$, we can apply Lemma 3.1 since (21) automatically holds, and (19) holds when $\rho \leq \frac{1}{L(f)}$. In (22), by letting $(x,y) = (x^*, y^*)$, and $\bar{y} = \bar{y}^n$, we get $(p,q) = (x^{n+1}, y^{n+1})$, and

$$2\rho(F(x^*,y^*) - F(x^{n+1},y^{n+1})) \geq \|y^{n+1} - y^*\|^2 - \|\bar{y}^n - y^*\|^2. \tag{58}$$

In (20), by letting $(\bar{x}, \bar{y}) = (x^*, y^*), (x, y) = (x^{n+1}, y^{n+1})$, we get $\bar{q} = \bar{y}^{n+1}$ and

$$\begin{aligned} 2\rho(F(x^*, y^*) - F(x^{n+1}, \bar{y}^{n+1})) &\geq \|\bar{y}^{n+1} - y^*\|^2 - \|y^{n+1} - y^*\|^2 + (x^* - x^{n+1})^T \nabla_x f(x^{n+1}, y^{n+1}) \\ &= \|\bar{y}^{n+1} - y^*\|^2 - \|y^{n+1} - y^*\|^2., \end{aligned} \tag{59}$$

since $\nabla_x f(x^{n+1}, y^{n+1}) = 0$, for $n \in I$ by (54) and for $n \in I_c$ by (18). Adding (59) to (58), we get

$$2\rho(2F(x^*, y^*) - F(x^{n+1}, y^{n+1}) - F(x^{n+1}, \bar{y}^{n+1})) \geq \|\bar{y}^{n+1} - y^*\|^2 - \|\bar{y}^n - y^*\|^2. \tag{60}$$

For $n \in I_c$, since $\nabla_x f(x^{n+1}, y^{n+1}) = 0$, we have that (59) holds. Since $y^{n+1} = \bar{y}^n$, it follows that

$$2\rho(F(x^*, y^*) - F(x^{n+1}, \bar{y}^{n+1})) \geq \|\bar{y}^{n+1} - y^*\|^2 - \|\bar{y}^n - y^*\|^2. \tag{61}$$

Summing (60) and (61) over $n = 0, 1, \ldots, k-1$ and observing that $2|I| + |I_c| = k + k_n$, we obtain

$$\begin{aligned} & 2\rho \left((k + k_n) F(x^*, y^*) - \sum_{n=1}^{k-1} F(x^{n+1}, \bar{y}^{n+1}) - \sum_{n \in I} F(x^{n+1}, y^{n+1}) \right) \\ \geq & \sum_{n=0}^{k-1} (\|\bar{y}^{n+1} - y^*\|^2 - \|\bar{y}^n - y^*\|^2) \\ = & \|\bar{y}^k - y^*\|^2 - \|\bar{y}^0 - y^*\|^2 \\ \geq & -\|\bar{y}^0 - y^*\|^2. \end{aligned} \tag{62}$$

In Lemma 3.1, by letting $(\bar{x}, \bar{y}) = (x^{n+1}, y^{n+1})$ in (20) instead of (x^*, y^*), we have from (59) that

$$2\rho(F(x^{n+1}, y^{n+1}) - F(x^{n+1}, \bar{y}^{n+1})) \geq \|\bar{y}^{n+1} - y^{n+1}\|^2 \geq 0. \tag{63}$$

Similarly, for $n \in I$, if we let $(x, y) = (x^n, \bar{y}^n)$ instead of (x^*, y^*) in (58), we have

$$2\rho(F(x^n, \bar{y}^n) - F(x^{n+1}, y^{n+1})) \geq \|y^{n+1} - \bar{y}^n\|^2 \geq 0. \tag{64}$$

For $n \in I_c$, $y^{n+1} = \bar{y}^n$; from (18), since $x^{n+1} = \arg\min_x F(x, y)$ with $y = \bar{y}^n = y^{n+1}$,

$$2\rho(F(x^n, \bar{y}^n) - F(x^{n+1}, y^{n+1})) \geq 0. \tag{65}$$

Hence, from (63) and (64) to (65), $F(x^n, y^n) \geq F(x^n, \bar{y}^n) \geq F(x^{n+1}, y^{n+1}) \geq F(x^{n+1}, \bar{y}^{n+1})$. Then, we have

$$\sum_{n=0}^{k-1} F(x^{n+1}, \bar{y}^{n+1}) \geq kF(x^k, \bar{y}^k), \text{and} \quad \sum_{n \in I} F(x^{n+1}, y^{n+1}) \geq k_n F(x^k, y^k). \tag{66}$$

Combining (62) and (66) yields $2\rho(k + k_n)(F(x^*, y^*) - F(x^k, \bar{y}^k)) \geq -\|\bar{y}^0 - y^*\|^2$.

C Derivation of the Stopping Criteria

In this section, we show that the quantities that we use in our stopping criteria correspond to the primal and dual residuals [8] for the outer iterations and the gradient residuals for the inner iterations. We first consider the inner iterations.

FISTA-p The necessary and sufficient optimality conditions for problem (12) or (15) are primal feasibility

$$\bar{y}^* - y^* = 0, \tag{67}$$

and vanishing of the gradient of the objective function at $(x^*, \bar{y}^*)$, i.e.

$$\begin{aligned} 0 &= \nabla_x f(x^*, \bar{y}^*), & (68)\\ 0 &\in \nabla_y f(x^*, \bar{y}^*) + \partial g(\bar{y}^*). & (69) \end{aligned}$$

Since $y^{k+1} = z^k$, the primal residual is thus $\bar{y}^{k+1} - y^{k+1} = \bar{y}^{k+1} - z^k$. It follows from the optimality of x^{k+1} in Line 3 of Algorithm 3.4 that

$$\begin{aligned} A^T(Ax^{k+1} - b) - C^T v^l + \frac{1}{\mu} C^T(Cx^{k+1} - \bar{y}^{k+1}) + \frac{1}{\mu} C^T(\bar{y}^{k+1} - z^k) &= 0\\ \nabla_x f(x^{k+1}, \bar{y}^{k+1}) &= \frac{1}{\mu} C^T(z^k - \bar{y}^{k+1}). \end{aligned}$$

Similarly, from the optimality of $\bar{y}^{k+1}$ in Line 4, we have that

$$\begin{aligned} 0 &\in \partial g(\bar{y}^{k+1}) + \nabla_y f(x^{k+1}, z^k) + \frac{1}{\rho}(\bar{y}^{k+1} - z^k)\\ &= \partial g(\bar{y}^{k+1}) + \nabla_y f(x^{k+1}, \bar{y}^{k+1}) - \frac{1}{\mu}(\bar{y}^{k+1} - z^k) + \frac{1}{\rho}(\bar{y}^{k+1} - z^k)\\ &= \partial g(\bar{y}^{k+1}) + \nabla_y f(x^{k+1}, \bar{y}^{k+1}), \end{aligned}$$

where the last step follows from $\mu = \rho$. Hence, we see that $\frac{1}{\mu} C^T(z^k - \bar{y}^{k+1})$ is the gradient residual corresponding to (68), while (69) is satisfied in every inner iteration.

APLM-S The primal residual is $\bar{y}^{k+1} - y^{k+1}$ from (67). Following the derivation for FISTA-p, it is not hard to verify that (69) is always satisfied, and the gradient residual corresponding to (68) is $\frac{1}{\mu} C^T(y^{k+1} - \bar{y}^{k+1})$.

FISTA Similar to FISTA-p, the necessary and sufficient optimality conditions for problem (28) are primal feasibility

$$(x^*, y^*) = (\bar{x}^*, \bar{y}^*),$$

and vanishing of the objective gradient at $(\bar{x}^*, \bar{y}^*)$,

$$\begin{aligned} 0 &= \nabla_x f(\bar{x}^*, \bar{y}^*),\\ 0 &\in \nabla_y f(\bar{x}^*, \bar{y}^*) + \partial g(\bar{y}^*). \end{aligned}$$

Clearly, the primal residual is $(\bar{x}^{k+1} - z_x^k, \bar{y}^{k+1} - z_y^k)$ since $(x^{k+1}, y^{k+1}) \equiv (z_x^k, z_y^k)$. From the optimality of $(\bar{x}^{k+1}, \bar{y}^{k+1})$, it follows that

$$\begin{aligned} 0 &= \nabla_x f(z_x^k, z_y^k) + \frac{1}{\rho}(\bar{x}^{k+1} - z_x^k),\\ 0 &\in \partial g(\bar{y}^{k+1}) + \nabla_y f(z_x^k, z_y^k) + \frac{1}{\rho}(\bar{y}^{k+1} - z_y^k). \end{aligned}$$

Here, we simply use $\frac{1}{\rho}(\bar{x}^{k+1} - z_x^k)$ and $\frac{1}{\rho}(\bar{y}^{k+1} - z_y^k)$ to approximate the gradient residuals.

Data Sets	Algs	CPU (s)	Iters	Avg Sub-iters	$F(x)$
ogl-5000-100-10-3	ADAL	1.70e+000	61	1.00e+000	1.9482e+005
	APLM-S	1.71e+000	8	4.88e+000	1.9482e+005
	FISTA-p	9.08e-001	8	4.38e+000	1.9482e+005
	FISTA	2.74e+000	10	7.30e+000	1.9482e+005
	ProxGrad	7.92e+001	3858	-	-
ogl-5000-600-10-3	ADAL	6.75e+001	105	1.00e+000	1.4603e+006
	APLM-S	1.79e+002	9	1.74e+001	1.4603e+006
	FISTA-p	4.77e+001	9	8.56e+000	1.4603e+006
	FISTA	3.28e+001	12	1.36e+001	1.4603e+006
	ProxGrad	7.96e+002	5608	-	-
ogl-5000-1000-10-3	ADAL	2.83e+002	151	1.00e+000	2.6746e+006
	APLM-S	8.06e+002	10	2.76e+001	2.6746e+006
	FISTA-p	2.49e+002	10	1.28e+001	2.6746e+006
	FISTA	5.21e+001	13	1.55e+001	2.6746e+006
	ProxGrad	1.64e+003	6471	-	-

Table 4: Numerical results for ogl set 1. For ProxGrad, Avg Sub-Iters and $F(x)$ fields are not applicable since the algorithm is not based on an outer-inner iteration scheme, and the objective function that it minimizes is different from ours. We tested ten problems with $J = 100, \cdots, 1000$, but only show the results for three of them to save space.

Next, we consider the outer iterations. The necessary and sufficient optimality conditions for problem (5) are primal feasibility

$$Cx^* - y^* = 0, \tag{70}$$

and dual feasibility

$$\begin{aligned} 0 &= \nabla L(x^*) - C^T v^* & (71)\\ 0 &\in \partial\tilde{\Omega}(y^*) + v^*. & (72) \end{aligned}$$

Clearly, the primal residual is $r^l = Cx^l - y^l$. The dual residual is $\begin{pmatrix} \nabla L(x^{l+1}) - C^T(v^l - \frac{1}{\mu}(Cx^{l+1} - \bar{y}^{l+1})) \\ \partial\tilde{\Omega}(y^{l+1}) + v^l - \frac{1}{\mu}(Cx^{l+1} - \bar{y}^{l+1}) \end{pmatrix}$, recalling that $v^{l+1} = v^l - \frac{1}{\mu}(Cx^{l+1} - \bar{y}^{l+1})$. The above is simply the gradient of the augmented Lagrangian (6) evaluated at (x^l, y^l, v^l). Now, since the objective function of an inner iteration is the augmented Lagrangian with $v = v^l$, the dual residual for an outer iteration is readily available from the gradient residual computed for the last inner iteration of the outer iteration.

D Numerical Results

References

[1] M. Afonso, J. Bioucas-Dias, and M. Figueiredo. An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems. *Image Processing, IEEE Transactions on*, (99):1–1, 2009.

Data Sets	Algs	CPU (s)	Iters	Avg Sub-iters	$F(x)$
ogl-1000-200-10-3	ADAL	4.18e+000	77	1.00e+000	9.6155e+004
	APLM-S	1.64e+001	9	2.32e+001	9.6156e+004
	FISTA-p	3.85e+000	9	1.02e+001	9.6156e+004
	FISTA	2.92e+000	11	1.44e+001	9.6158e+004
	ProxGrad	1.16e+002	4137	-	-
ogl-5000-200-10-3	ADAL	5.04e+000	63	1.00e+000	4.1573e+005
	APLM-S	8.42e+000	8	8.38e+000	4.1576e+005
	FISTA-p	3.96e+000	9	6.56e+000	4.1572e+005
	FISTA	6.54e+000	10	9.70e+000	4.1573e+005
	ProxGrad	1.68e+002	4345	-	-
ogl-10000-200-10-3	ADAL	6.41e+000	44	1.00e+000	1.0026e+006
	APLM-S	1.46e+001	10	7.60e+000	1.0026e+006
	FISTA-p	5.60e+000	10	5.50e+000	1.0026e+006
	FISTA	1.09e+001	10	8.50e+000	1.0027e+006
	ProxGrad	3.31e+002	6186	-	-

Table 5: Numerical results for ogl set 2. We ran the test for ten problems with $n = 1000, \cdots, 10000$, but only show the results for three of them to save space.

[2] F. Bach. Consistency of the group Lasso and multiple kernel learning. *The Journal of Machine Learning Research*, 9:1179–1225, 2008.

[3] F. Bach. Structured sparsity-inducing norms through submodular functions. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 118–126. 2010.

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

[5] D. Bertsekas. Multiplier methods: a survey. *Automatica*, 12(2):133–145, 1976.

[6] D. Bertsekas. *Nonlinear programming*. Athena Scientific Belmont, MA, 1999.

[7] D. Bertsekas and J. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Machine Learning*, 3(1):1–123, 2010.

[9] P. Brucker. An o (n) algorithm for quadratic knapsack problems. *Operations Research Letters*, 3(3):163–166, 1984.

[10] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM journal on scientific computing*, 20(1):33–61, 1999.

[11] X. Chen, Q. Lin, S. Kim, J. Peña, J. Carbonell, and E. Xing. An Efficient Proximal-Gradient Method for Single and Multi-task Regression with Structured Sparsity. *Arxiv preprint arXiv:1005.4717*, 2010.

Data Sets	Algs	CPU (s)	Iters	Avg Sub-iters	$F(x)$
ogl-dct-1000-5000-1	ADAL	1.14e+001	194	1.00e+000	8.4892e+002
	FISTA-p	1.21e+001	20	1.11e+001	8.4892e+002
	FISTA	2.49e+001	24	2.51e+001	8.4893e+002
ogl-dct-1000-10000-1	ADAL	3.31e+001	398	1.00e+000	1.4887e+003
	FISTA-p	2.54e+001	41	5.61e+000	1.4887e+003
	FISTA	6.33e+001	44	1.74e+001	1.4887e+003
ogl-dct-1000-15000-1	ADAL	6.09e+001	515	1.00e+000	2.7506e+003
	FISTA-p	3.95e+001	52	4.44e+000	2.7506e+003
	FISTA	9.73e+001	54	1.32e+001	2.7506e+003
ogl-dct-1000-20000-1	ADAL	9.52e+001	626	1.00e+000	3.3415e+003
	FISTA-p	6.66e+001	63	6.10e+000	3.3415e+003
	FISTA	1.81e+002	64	1.61e+001	3.3415e+003
ogl-dct-1000-25000-1	ADAL	1.54e+002	882	1.00e+000	4.1987e+003
	FISTA-p	7.50e+001	88	3.20e+000	4.1987e+003
	FISTA	1.76e+002	89	8.64e+000	4.1987e+003
ogl-dct-1000-30000-1	ADAL	1.87e+002	957	1.00e+000	4.6111e+003
	FISTA-p	8.79e+001	96	2.86e+000	4.6111e+003
	FISTA	2.24e+002	94	8.54e+000	4.6111e+003

Table 6: Numerical results for dct set 2 (scalability test) with l_1/l_2-regularization. All three algorithms were ran in factorization mode with a fixed $\mu = \mu_0$.

[12] P. Combettes and J. Pesquet. Proximal splitting methods in signal processing. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212, 2011.

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

[14] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

[15] J. Eckstein and D. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318, 1992.

[16] J. Eckstein and P. Silva. A practical relative error criterion for augmented lagrangians. Technical report, Rutgers University, 2011.

[17] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[18] R. Glowinski and A. Marroco. Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualite d'une classe de problemes de dirichlet non lineares. *Rev. Francaise d'Automat. Inf. Recherche Operationelle*, (9):41–76, 1975.

Data Sets	Algs	CPU (s)	Iters	Avg Sub-iters	$F(x)$
ogl-dct-1000-5000-1	ADAL	1.53e+001	266	1.00e+000	7.3218e+002
	FISTA-p	1.61e+001	10	3.05e+001	7.3219e+002
	FISTA	3.02e+001	16	4.09e+001	7.3233e+002
	ProxFlow	1.97e+001	-	-	7.3236e+002
ogl-dct-1000-10000-1	ADAL	3.30e+001	330	1.00e+000	1.2707e+003
	FISTA-p	3.16e+001	10	3.10e+001	1.2708e+003
	FISTA	7.27e+001	24	3.25e+001	1.2708e+003
	ProxFlow	3.67e+001	-	-	1.2709e+003
ogl-dct-1000-15000-1	ADAL	4.83e+001	328	1.00e+000	2.2444e+003
	FISTA-p	5.40e+001	15	2.52e+001	2.2444e+003
	FISTA	8.64e+001	23	2.66e+001	2.2449e+003
	ProxFlow	9.91e+001	-	-	2.2467e+003
ogl-dct-1000-20000-1	ADAL	8.09e+001	463	1.00e+000	2.6340e+003
	FISTA-p	8.09e+001	16	2.88e+001	2.6340e+003
	FISTA	1.48e+002	26	2.93e+001	2.6342e+003
	ProxFlow	2.55e+002	-	-	2.6357e+003
ogl-dct-1000-25000-1	ADAL	7.48e+001	309	1.00e+000	3.5566e+003
	FISTA-p	1.15e+002	30	1.83e+001	3.5566e+003
	FISTA	2.09e+002	38	2.30e+001	3.5568e+003
	ProxFlow	1.38e+002	-	-	3.5571e+003
ogl-dct-1000-30000-1	ADAL	9.99e+001	359	1.00e+000	3.7057e+003
	FISTA-p	1.55e+002	29	2.17e+001	3.7057e+003
	FISTA	2.60e+002	39	2.25e+001	3.7060e+003
	ProxFlow	1.07e+002	-	-	3.7063e+003

Table 7: Numerical results for dct set 2 (scalability test) with l_1/l_∞-regularization. The algorithm configurations are exactly the same as in Table 6.

[19] D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. *Arxiv preprint arXiv:0912.4571v2*, 2009.

[20] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2:323, 2009.

[21] G. Golub and C. Van Loan. *Matrix computations*. Johns Hopkins Univ Pr, 1996.

[22] M. Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

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

[24] L. Jacob, G. Obozinski, and J. Vert. Group Lasso with overlap and graph Lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 433–440. ACM, 2009.

[25] R. Jenatton, J. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. *Stat*, 1050, 2009.

Data Sets	Algs	CPU (s)	Iters	Avg Sub-iters	$F(x)$
ogl-dct-1000-5000-1	FISTA-p	1.83e+001	12	2.34e+001	8.4892e+002
	FISTA	2.49e+001	24	2.51e+001	8.4893e+002
	ADAL	1.35e+001	181	1.00e+000	8.4892e+002
ogl-dct-1000-10000-1	FISTA-p	3.16e+001	14	1.73e+001	1.4887e+003
	FISTA	6.33e+001	44	1.74e+001	1.4887e+003
	ADAL	4.43e+001	270	1.00e+000	1.4887e+003
ogl-dct-1000-15000-1	FISTA-p	4.29e+001	14	1.51e+001	2.7506e+003
	FISTA	9.73e+001	54	1.32e+001	2.7506e+003
	ADAL	5.37e+001	216	1.00e+000	2.7506e+003
ogl-dct-1000-20000-1	FISTA-p	7.53e+001	13	2.06e+001	3.3416e+003
	FISTA	1.81e+002	64	1.61e+001	3.3415e+003
	ADAL	1.57e+002	390	1.00e+000	3.3415e+003
ogl-dct-1000-25000-1	FISTA-p	7.41e+001	15	1.47e+001	4.1987e+003
	FISTA	1.76e+002	89	8.64e+000	4.1987e+003
	ADAL	8.79e+001	231	1.00e+000	4.1987e+003
ogl-dct-1000-30000-1	FISTA-p	8.95e+001	14	1.58e+001	4.6111e+003
	FISTA	2.24e+002	94	8.54e+000	4.6111e+003
	ADAL	1.12e+002	249	1.00e+000	4.6111e+003

Table 8: Numerical results for the DCT set with l_1/l_2-regularization. FISTA-p and ADAL were ran in PCG mode with the dynamic scheme for updating μ. μ was fixed at μ_0 for FISTA.

[26] R. Jenatton, G. Obozinski, and F. Bach. Structured sparse principal component analysis. *Arxiv preprint arXiv:0909.1440*, 2009.

[27] S. Kim and E. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proceedings of the 27th Annual International Conference on Machine Learning*, 2010.

[28] Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *Arxiv preprint arXiv:1009.5055*, 2010.

[29] J. Liu and J. Ye. Fast Overlapping Group Lasso. *Arxiv preprint arXiv:1009.0306*, 2010.

[30] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1558–1566. 2010.

[31] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Convex and Network flow Optimization for Structured Sparsity. *Arxiv preprint arXiv:1104.1872v1*, 2011.

[32] S. Mosci, S. Villa, A. Verri, and L. Rosasco. A primal-dual algorithm for group sparse regularization with overlapping groups. In *at NIPS*, 2010.

[33] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[34] J. Nocedal and S. Wright. *Numerical optimization*. Springer verlag, 1999.

[35] J. Pesquet and N. Pustelnik. A parallel inertial proximal optimization method. *Preprint*, 2010.

Data Sets	Algs	CPU (s)	Iters	Avg Sub-iters	$F(x)$
ogl-dct-1000-5000-1	FISTA-p	2.30e+001	11	2.93e+001	7.3219e+002
	ADAL	1.89e+001	265	1.00e+000	7.3218e+002
	FISTA	3.02e+001	16	4.09e+001	7.3233e+002
	ProxFlow	1.97e+001	-	-	7.3236e+002
ogl-dct-1000-10000-1	FISTA-p	5.09e+001	11	3.16e+001	1.2708e+003
	ADAL	4.77e+001	323	1.00e+000	1.2708e+003
	FISTA	7.27e+001	24	3.25e+001	1.2708e+003
	ProxFlow	3.67e+001	-	-	1.2709e+003
ogl-dct-1000-15000-1	FISTA-p	6.33e+001	12	2.48e+001	2.2445e+003
	ADAL	9.41e+001	333	1.00e+000	2.2444e+003
	FISTA	8.64e+001	23	2.66e+001	2.2449e+003
	ProxFlow	9.91e+001	-	-	2.2467e+003
ogl-dct-1000-20000-1	FISTA-p	8.21e+001	12	2.42e+001	2.6341e+003
	ADAL	1.59e+002	415	1.00e+000	2.6340e+003
	FISTA	1.48e+002	26	2.93e+001	2.6342e+003
	ProxFlow	2.55e+002	-	-	2.6357e+003
ogl-dct-1000-25000-1	FISTA-p	1.43e+002	13	2.98e+001	3.5567e+003
	ADAL	1.20e+002	310	1.00e+000	3.5566e+003
	FISTA	2.09e+002	38	2.30e+001	3.5568e+003
	ProxFlow	1.38e+002	-	-	3.5571e+003
ogl-dct-1000-30000-1	FISTA-p	1.75e+002	13	3.18e+001	3.7057e+003
	ADAL	2.01e+002	361	1.00e+000	3.7057e+003
	FISTA	2.60e+002	39	2.25e+001	3.7060e+003
	ProxFlow	1.07e+002	-	-	3.7063e+003

Table 9: Numerical results for the DCT set with l_1/l_∞-regularization. FISTA-p and ADAL were ran in PCG mode. The dynamic updating scheme for μ was applied to FISTA-p, while μ was fixed at μ_0 for ADAL and FISTA.

[36] G. Peyre and J. Fadili. Group sparsity with overlapping partition functions. In *EUSIPCO 2011*, 2011.

[37] G. Peyre, J. Fadili, and C. Chesneau. Adaptive structured block sparsity via dyadic partitioning. In *EUSIPCO 2011*, 2011.

[38] M. Powell. *Optimization*, chapter A Method for Nonlinear Constraints in Minimization Problems. Academic Press, New York, New York, 1972.

[39] Z. Qin, K. Scheinberg, and D. Goldfarb. Efficient Block-coordinate Descent Algorithms for the Group Lasso. 2010.

[40] R. Rockafellar. The multiplier method of hestenes and powell applied to convex programming. *Journal of Optimization Theory and Applications*, 12(6):555–562, 1973.

[41] V. Roth and B. Fischer. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *Proceedings of the 25th international conference on Machine learning*, pages 848–855. ACM, 2008.

Data Sets	Algs	CPU (s)	Iters	Avg Sub-iters	$F(x)$
BreastCancerData	ADAL	6.24e+000	136	1.00e+000	2.9331e+003
	APLM-S	4.02e+001	12	4.55e+001	2.9331e+003
	FISTA-p	6.86e+000	12	1.48e+001	2.9331e+003
	FISTA	5.11e+001	75	1.29e+001	2.9340e+003
	ProxGrad	7.76e+002	6605	1.00e+000	-

Table 10: Numerical results for Breast Cancer Data using l_1/l_2-regularization. In this experiment, we kept μ constant at 0.01 for ADAL. The CPU time is for a single run on the entire data set with the value of λ selected to minimize the RMSE in Figure 4.

[42] S. Setzer, G. Steidl, and T. Teuber. Deblurring poissonian images by split bregman techniques. *Journal of Visual Communication and Image Representation*, 21(3):193–199, 2010.

[43] J. Spingarn. Partial inverse of a monotone operator. *Applied mathematics & optimization*, 10(1):247–265, 1983.

[44] A. Subramanian, P. Tamayo, V. Mootha, S. Mukherjee, B. Ebert, M. Gillette, A. Paulovich, S. Pomeroy, T. Golub, E. Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15545, 2005.

[45] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[46] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *iccv*, page 255. Published by the IEEE Computer Society, 1999.

[47] M. Van De Vijver, Y. He, L. van't Veer, H. Dai, A. Hart, D. Voskuil, G. Schreiber, J. Peterse, C. Roberts, M. Marton, et al. A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, 347(25):1999, 2002.

[48] S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

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

[50] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.