MPGL: An Efficient Matching Pursuit Method for Generalized LASSO

Dong Gong[†‡], Mingkui Tan[*], Yanning Zhang[†], Anton van den Hengel[‡§], Qinfeng Shi[‡]
[†]School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, China
[‡]School of Computer Science, The University of Adelaide, Australia, [§]The Australian Centre for Robotic Vision
[*]School of Software Engineering, South China University of Technology, China
edgong01@gmail.com, mingkuitan@scut.edu.cn, ynzhang@nwpu.edu.cn,
{anton.vandenhengel, javen.shi}@adelaide.edu.au

Abstract

Unlike traditional LASSO enforcing sparsity on the variables, Generalized LASSO (GL) enforces sparsity on a linear transformation of the variables, gaining flexibility and success in many applications. However, many existing GL algorithms do not scale up to high-dimensional problems, and/or only work well for a specific choice of the transformation. We propose an efficient Matching Pursuit Generalized LASSO (MPGL) method, which overcomes these issues, and is guaranteed to converge to a global optimum. We formulate the GL problem as a convex quadratic constrained linear programming (QCLP) problem and tailor-make a cutting plane method. More specifically, our MPGL iteratively activates a subset of nonzero elements of the transformed variables, and solves a subproblem involving only the activated elements thus gaining significant speed-up. Moreover, MPGL is less sensitive to the choice of the trade-off hyper-parameter between data fitting and regularization, and mitigates the long-standing hyper-parameter tuning issue in many existing methods. Experiments demonstrate the superior efficiency and accuracy of the proposed method over the state-of-the-arts in both classification and image processing tasks.

Introduction

Learning with sparsity-inducing norms has gained much success in many applications including medical data analysis (Tibshirani and Wang 2008), image processing (Rudin, Osher, and Fatemi 1992), feature selection (Tan, Tsang, and Wang 2014) and so on. One efficient way to enforce sparsity on the variables is to use the ℓ_1-norm as *LASSO* (Tibshirani 1996) instead of the ℓ_0-norm. Since then, many methods have been proposed to enforce some additional constraints (Huang, Zhang, and Metaxas 2011; Kim and Xing 2010; Tibshirani et al. 2011) to improve the results. A group of methods among them is called *generalized LASSO* (Tibshirani et al. 2011), which promotes the sparsity of the variables after a linear transformation (Liu, Yuan, and Ye 2013) instead of the variables themselves. The choice of such a transformation represents the property of the variables to be desired, and often depends on the application.

Generalized LASSO. Let $\mathbf{x} \in \mathbb{R}^n$ denote the target variable and $\mathbf{D} \in \mathbb{R}^{l \times n}$ be a linear transformation operator. A natural way to seek $\mathbf{x}$ with sparsity on $\mathbf{Dx}$ is as follows (Liu, Yuan, and Ye 2013),

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda \|\mathbf{Dx}\|_0, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a loss function (sometimes known as data fitting term) depending on the application, $\|\cdot\|_0$ denotes the ℓ_0-norm regularizer, and $\lambda \geq 0$ is known as the trade-off hyper-parameter between the data fitting and the regularization. By letting $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a designing matrix, $\mathbf{y} \in \mathbb{R}^m$ be a response vector, $\mathbf{n} \in \mathbb{R}^m$ be a vector of Gaussian noise, and assuming a linear regression model $\mathbf{y} = \mathbf{Ax} + \mathbf{n}$, a typical choice of f is $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{Ax}\|_2^2$, which will be used throughout the rest of the paper. Since problem (1) is NP-hard, a convex relaxation is widely used:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{Ax}\|_2^2 + \lambda \|\mathbf{Dx}\|_1, \tag{2}$$

which is referred to as *Generalized LASSO* (Tibshirani et al. 2011). $\mathbf{D}$ depends on the application, and reflects the desirable behaviors of the variables. Some well-known applications and choices of $\mathbf{D}$ are:

- 1-dimensional *total variation* (TV) model (Rudin, Osher, and Fatemi 1992) where $\mathbf{D} \in \mathbb{R}^{(n-1)\times n}$ and

$$\mathbf{D} = \mathbf{Q}(\mathrm{TV}) = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix}. \tag{3}$$

  Such a $\mathbf{D}$ penalizes the absolute differences in adjacent coordinates of $\mathbf{x}$.
- *Fused LASSO* (FL) (Tibshirani et al. 2005), which solves $\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{Ax}\|_2^2 + \lambda_1\|\mathbf{x}\|_1 + \lambda_2\|\mathbf{Qx}\|_1$, and can be re-written as problem (2) with $\mathbf{D} = [\frac{\lambda_1}{\lambda}\mathbf{I}^\mathsf{T}, \frac{\lambda_2}{\lambda}\mathbf{Q}(\mathrm{TV})^\mathsf{T}]^\mathsf{T}$, a concatenation of an identity matrix and the TV matrix $\mathbf{Q}(\mathrm{TV})$. Note that if letting $\mathbf{D} = \mathbf{I}$, the problem reduces to the classical LASSO problem.
- *Generalized fused LASSO* (GFL) (Xin et al. 2014), which extends the TV model using a graph. Let $G = (V, E)$ be a graph with vertices V and edges E, where each vertex corresponds to an element of the variable $\mathbf{x}$. The graph-guided TV-norm is defined as $\sum_{(i,j)\in E} |x_i - x_j|$. In the matrix form like (3), the k-th edge (i, j) corresponds to the k-th row of the matrix $\mathbf{Q}(G) \in \mathbb{R}^{|E|\times n}$

The first two authors contributed equally.

with $\mathbf{Q}(G)_{k,i} = 1$, $\mathbf{Q}(G)_{k,j} = -1$, and other elements as 0. Problem (2) with $\mathbf{D} = [\frac{\lambda_1}{\lambda}\mathbf{I}^\top, \frac{\lambda_2}{\lambda}\mathbf{Q}(G)^\top]^\top$ is referred to as a GFL problem.

Due to the flexibility and generality of the Generalized LASSO (GL), it has attracted much attention recently. Unlike traditional LASSO, solving GL efficiently on high-dimensional data is very challenging. A few attempts have been made to improve the efficiency of GL, but requires specific form of the $\mathbf{D}$ to work well (Tibshirani et al. 2011; Liu, Yuan, and Ye 2010; Xin et al. 2014). Furthermore, due to the regularization bias (the choice of the trade-off hyper-paramter) rooted in the ℓ_1-norm, achieving a sparse and unbiased solution simultaneously via solving (2) is very difficult (Zhang and Huang 2008; Zhang 2010; Tan, Tsang, and Wang 2014). To tackle these issues, we propose an efficient ***Matching Pursuit*** algorithm for **G**eneralized **L**ASSO problem (MPGL). The core contributions of this paper are summarized as follows:

- We formulate the GL problem as a QCLP problem for the first time, and propose a matching pursuit algorithm to solve it. Instead of handling all elements in $\mathbf{Dx}$, our method iteratively identifies the most advantageous subset of nonzero elements in $\mathbf{Dx}$, and solves a subproblem focusing on only the subset. This not only reduces the computational complexity substantially, but also reduces the impact of the noise and corruption in the data.
- We transform the subproblem into an equivalent but an easier optimization problem, which helps to improve the optimization speed by a few orders of magnitudes.
- Many sparsity-inducing norms based methods suffer from the extensive selection of the parameter λ to achieve a sparse solution whilst fits the data well. Due to iterative nature of our method, an early stopping can be applied, which makes the proposed method easier to achieve a sparse solution that fits the data well.
- Unlike many previous methods which only work well on specific choices of $\mathbf{D}$, the proposed method can handle more general GL problems with different $\mathbf{D}$'s.

Related Studies

The GL problem for the general formulation of $\mathbf{D}$ was first defined and summarized in (Tibshirani et al. 2011). Before that, there had been a range of studies for important special cases such as the Fused LASSO (FL) problem (Tibshirani et al. 2005), the TV regularizer (Rudin, Osher, and Fatemi 1992) and trending filtering (Kim et al. 2009).

One particularly popular special case of (2) is the fused LASSO problem. Tibshirani *et.al.* (Tibshirani et al. 2005) proposed the first algorithm for FL based a two-phase active set algorithm labelled SQOPT. This approach does not scale to high-dimensional problems guided by general graph, however. Friedman *et.al.* (Friedman et al. 2007) then derived a pathwise coordinate descent algorithm for a special case of FL with $\mathbf{A} = \mathbf{I}$, which has no guarantee of an exact solution. Following that, a faster path algorithm was proposed based on max flow subroutines (Hoefling 2010). In (Liu, Yuan, and Ye 2010), an efficient algorithm based on fast iterative shrinkage-thresholding (FISTA) was proposed, but this approach is not applicable to the general formulation of the problem. Wang *et.al.* (Wang, Fan, and Ye 2015) proposed a screening rule based method to accelerate the optimization. For some specific problems with a piecewise smooth and non-sparse solution, the TV regularizer is widely used (Rudin, Osher, and Fatemi 1992). Wang *et.al.* (Wang et al. 2008) solved an approximately relaxed problem for fast image restoration.

Recently, some methods were proposed for solving the general problem. In (Tibshirani et al. 2011), a dual path algorithm was proposed for the GL problem with any formulation of $\mathbf{D}$, which however tends to be slow on large-scale problems. Xin *et.al.* (Xin et al. 2014) proposed to solve the GFL problem based on the FISTA and a parametric flow algorithm. An augmented alternating direction methods of multipliers algorithm (Zhu 2016) was proposed for GL.

Formulation

Notation. Let $\mathbf{A} = [A_{i,j}] \in \mathbb{R}^{m\times n}$ and $\mathbf{v} = [v_1, ..., v_n]^\top \in \mathbb{R}^n$ denote a matrix and a vector, respectively, where $^\top$ denotes the transpose of a vector/matrix. Let $\mathbf{0}$ and $\mathbf{1}$ be vectors with all zeros and all ones, respectively, and let $\mathbf{I}$ denote the identity matrix. Let $\mathbf{v}^i$ or $\mathbf{v}_i$ be a vector indexed for some purpose. Given a vector $\mathbf{v}$, let $\text{diag}(\mathbf{v})$ be a diagonal matrix with diagonal elements equal to the vector $\mathbf{v}$, and $\|\mathbf{v}\|_p$ be the ℓ_p-norm. Let $\odot$ denote the element wise product. Given a positive integer n, let $[n] = \{1, ..., n\}$. Given any index set $\mathcal{T} \subseteq [n]$, let $\mathcal{T}^c$ be the complementary set of $\mathcal{T}$, i.e. $\mathcal{T}^c = [n] \setminus \mathcal{T}$. For a vector $\mathbf{v} \in \mathbb{R}^n$, let v_i denote the i-th element of $\mathbf{v}$, and $\mathbf{v}_\mathcal{T}$ denote the subvector indexed by $\mathcal{T}$.

QCLP Formulation for Generalized LASSO [1]

To handle the non-smooth and non-separable regularizer $\|\mathbf{Dx}\|_1$, we introduce an auxiliary variable $\mathbf{z} \in \mathbb{R}^l$ to replace $\mathbf{Dx}$. To identify the nonzero components in $\mathbf{Dx}$ w.r.t. to the response vector $\mathbf{y}$, we introduce a binary index vector $\boldsymbol{\tau} \in \{0,1\}^l$ to scale $\mathbf{z}$ by $(\mathbf{z} \odot \boldsymbol{\tau})$. Regarding the goal to induce sparsity in $\mathbf{Dx}$, we impose an ℓ_0-norm constraint $\|\boldsymbol{\tau}\|_0 \leq \kappa$, and constrain $\mathbf{Dx} = \mathbf{z}\odot\boldsymbol{\tau}$ in our model. For simplicity, let $\boldsymbol{\Lambda} = \{\boldsymbol{\tau} | \boldsymbol{\tau} \in \{0,1\}^l, \|\boldsymbol{\tau}\|_0 \leq \kappa\}$ be the feasible domain of $\boldsymbol{\tau}$. Let $\boldsymbol{\xi} = \mathbf{y} - \mathbf{Ax}$ represent the regression error. We consider solving the generalized LASSO by addressing

$$\begin{aligned} \min_{\boldsymbol{\tau}\in\boldsymbol{\Lambda}} \min_{\mathbf{x},\boldsymbol{\xi},\mathbf{z}} \ & \frac{1}{2}\|\boldsymbol{\xi}\|_2^2 + \lambda\|\mathbf{z}\|_1 \\ \text{s.t.} \ & \boldsymbol{\xi} = \mathbf{y} - \mathbf{Ax}, \ \mathbf{Dx} = (\mathbf{z} \odot \boldsymbol{\tau}). \end{aligned} \tag{4}$$

In (4), the integer κ reflects a rough knowledge of the sparsity of $\mathbf{Dx}$, and there are $|\boldsymbol{\Lambda}| = \sum_{i=0}^{\kappa} \binom{n}{i}$ feasible $\boldsymbol{\tau}$'s in $\boldsymbol{\Lambda}$. Problem (4) tends to find an optimal $\boldsymbol{\tau}$ from $\boldsymbol{\Lambda}$, which minimizes the regression loss $\|\boldsymbol{\xi}\|_2^2$ by constraining $\mathbf{x}$. Note that the optimal $\boldsymbol{\tau}$ according to (4) might not be unique.

Problem (4) is a mixed integer programming problem, thus it is hard to solve. We address it by relaxing it to a convex *QCLP* (Pee and Royset 2011) problem:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A},\theta\in\mathbb{R}} \theta, \ \text{ s.t. } \phi(\boldsymbol{\alpha}, \boldsymbol{\tau}) \leq \theta, \forall \boldsymbol{\tau} \in \boldsymbol{\Lambda}, \tag{5}$$

[1] All the proofs can be found in the supplementary materials.

where $\phi(\boldsymbol{\alpha}, \boldsymbol{\tau}) = -\frac{1}{2}\|\boldsymbol{\alpha}\|_2^2 + \boldsymbol{\alpha}^\mathsf{T}\mathbf{y}, \boldsymbol{\alpha} \in \mathcal{A}_{\boldsymbol{\tau}}$ and $\mathcal{A} = \cap \mathcal{A}_{\boldsymbol{\tau}}$ with $\mathcal{A}_{\boldsymbol{\tau}} = \{\boldsymbol{\alpha} | \mathbf{A}^\mathsf{T}\boldsymbol{\alpha} = \mathbf{D}^\mathsf{T}\boldsymbol{\beta}, \|\mathrm{diag}(\boldsymbol{\tau})\boldsymbol{\beta}\|_\infty \le \lambda, \boldsymbol{\alpha} \in [-h, h]^n\}$. Here $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ refer to the Lagrangian dual variables w.r.t. the two constraints in (4), respectively.

Cutting Planes for the QCLP Problem

Problem (5) is essentially a QCLP problem with $T = \sum_{i=0}^{\kappa} \binom{n}{i}$ constraints since there are T elements in $\boldsymbol{\Lambda}$, which makes the problem difficult to address directly. However, most of the constraints in (5) are inactive at the optimum, if only a subset of nonzero components in $\mathbf{Dx}$ are relevant in fitting the observation $\mathbf{y}$. Accordingly, we seek to address problem (5) using a cutting plane method (Mutapcic and Boyd 2009; Tan et al. 2012) as shown in Algorithm 1.

Instead of handling all constraints simultaneously, we iteratively find the most-violated constraint and add it into an active constraint set $\boldsymbol{\Lambda}_t$ which is initialized as an empty set $\emptyset$. Then we solve subproblem (6) using the active constraints in $\boldsymbol{\Lambda}_t$. In the following, we will discuss how to find the most violated constraint and solve subproblem (6).

Algorithm 1: Cutting Planes for the QCLP Problem (5)

Input: Observation $\mathbf{y}$, $\mathbf{A}$, λ, and initialization of $\boldsymbol{\alpha}^0$.

Initialize $\boldsymbol{\tau}_0 = \mathbf{0}$. Set $\boldsymbol{\Lambda}_0 = \emptyset$ and $t = 1$;

while *Stopping conditions are not achieved* **do**

  Find the most violated $\boldsymbol{\tau}_t$ based on $\boldsymbol{\alpha}^{t-1}$;

  Set $\boldsymbol{\Lambda}_t = \boldsymbol{\Lambda}_{t-1} \cup \{\boldsymbol{\tau}_t\}$;

  Solve the subproblem corresponding to $\boldsymbol{\Lambda}_t$

$$\min_{\boldsymbol{\alpha} \in \mathcal{A}, \theta \in \mathbb{R}} \theta, \ \text{ s.t. } \phi(\boldsymbol{\alpha}, \boldsymbol{\tau}) - \theta \le 0, \forall \boldsymbol{\tau} \in \boldsymbol{\Lambda}_t, \tag{6}$$

  obtaining the optimal solution $\boldsymbol{\alpha}^t$. Let $t = t + 1$.

Finding the Most Violated Constraint

We now seek to find the most active $\boldsymbol{\tau}$ (which corresponds to the most violated constraint) from a large number elements in $\boldsymbol{\Lambda}$. At the optimum of problem (5), the following condition should hold for all $\boldsymbol{\tau}$'s:

$$\|\mathrm{diag}(\boldsymbol{\tau})\boldsymbol{\beta}\|_\infty \le \lambda, \mathbf{A}^\mathsf{T}\boldsymbol{\alpha} = \mathbf{D}^\mathsf{T}\boldsymbol{\beta}. \tag{7}$$

In each iteration of Algorithm 1, with an updated and fixed $\boldsymbol{\alpha}$, we will find the value of $\boldsymbol{\beta}$ using the constraint $\mathbf{A}^\mathsf{T}\boldsymbol{\alpha} = \mathbf{D}^\mathsf{T}\boldsymbol{\beta}$ as the cue. It is ill-posed to recover $\boldsymbol{\beta}$ from $\boldsymbol{\alpha}$ by solving the linear system $\mathbf{D}^\mathsf{T}\boldsymbol{\beta} = \mathbf{A}^\mathsf{T}\boldsymbol{\alpha}$ directly. Thus we try to obtain $\boldsymbol{\beta}$ approximately by solving:

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{D}^\mathsf{T}\boldsymbol{\beta} - \mathbf{A}^\mathsf{T}\boldsymbol{\alpha}\|_2^2 + \frac{r}{2}\|\boldsymbol{\beta}\|_2^2, \tag{8}$$

where $\|\boldsymbol{\beta}\|_2^2$ is added to reduce the ill-posedness and $r \ge 0$ is a penalty parameter. Note that (8) has a closed-form solution. We can obtain $\boldsymbol{\beta}$ efficiently. In general, a conjugate gradient algorithm with $r = 2$ works well.

As shown in (7), any $|\beta_i| > \lambda$ violates the optimality condition, and the largest $|\beta_i|$ violates the condition the most. Due to the constraint $\|\boldsymbol{\tau}\|_0 \le \kappa$, we find the κ largest $|\beta_i|$, then set the corresponding τ_i to 1 and the rest to 0 to construct the most active $\boldsymbol{\tau}$. In practice, we record the κ indices into a set $\mathcal{C}_t$, i.e. $\mathcal{C}_t = \mathrm{support}(\boldsymbol{\tau}_t)$. Furthermore, let $\mathcal{S}_t = \cup_{i=1}^{t} \mathcal{C}_i$ record the indices of all activated constraints. To avoid overlapping components among $\mathcal{C}_t$, we form $\mathcal{C}_t$ from $[n] \setminus \mathcal{S}_{t-1}$. The algorithm for finding the most violated constraint is summarized in Algorithm 2.

Algorithm 2: Finding the Most Violated Constraint

Input: $\boldsymbol{\alpha}$, κ, $\mathbf{A}$ and regularizer parameter r.

Calculate $\boldsymbol{\beta}$ by solving problem (8);

Initialize $\boldsymbol{\tau} = \mathbf{0}$;

Find the κ largest $|\beta_i|$'s, and set corresponding τ_i to 1;

Form $\mathcal{C}$ and return $\boldsymbol{\tau}$ and $\mathcal{C}$.

Fast Optimization of Subproblem (6)

The number of the constraints is greatly reduced in the subproblem (6) compared to the original problem (5). Now we show how to utilize the relationship between the active constraints and the indices of the active components in $\mathbf{Dx}$ and $\mathbf{z}$, to transfer problem (6) to an equivalent problem (w.r.t. the primal variables $\mathbf{x}$ and $\mathbf{z}_\mathcal{S}$) that can be solved much faster.

Proposition 1. *Let $\mathcal{S} = \cup_{i=1}^{t} \mathcal{C}_i$. When there are no overlapping elements among $\mathcal{C}_i$'s, problem* (6) *can be addressed by solving*

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}_\mathcal{S}} \ & \frac{1}{2}\|\mathbf{y} - \mathbf{Ax}\|_2^2 + \lambda\|\mathbf{z}_\mathcal{S}\|_1 \\ \text{s.t. } & (\mathbf{Dx})_\mathcal{S} = \mathbf{z}_\mathcal{S}, (\mathbf{Dx})_{\mathcal{S}^c} = \mathbf{0}. \end{aligned} \tag{9}$$

Additionally, the optimal value of $\boldsymbol{\alpha}^*$ under problem (6) *can be recovered by $\boldsymbol{\alpha}^* = \boldsymbol{\xi}^*$ where $\boldsymbol{\xi}^* = \mathbf{y} - \mathbf{Ax}^*$.*

The new subproblem (9) facilitates the usage of an alternating direction method of multipliers (ADMM) (Boyd et al. 2011). More specifically, we iteratively update the primal and dual variables of the augmented Lagrangian function of (9). By introducing a dual variable $\boldsymbol{\gamma} \in \mathbb{R}^l$ and a positive penalty parameter $\rho > 0$, and letting subvectors $\boldsymbol{\gamma}_\mathcal{S}$ and $\boldsymbol{\gamma}_{\mathcal{S}^c}$ be the dual variables w.r.t. the two constraints, respectively, we obtain the augmented Lagrangian for (9) as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{z}_{\tilde{\mathcal{S}}}, \boldsymbol{\gamma}) = & \frac{1}{2}\|\mathbf{y} - \mathbf{Ax}\|_2^2 + \lambda\|\mathbf{z}_\mathcal{S}\|_1 + \boldsymbol{\gamma}_\mathcal{S}^\mathsf{T}((\mathbf{Dx})_\mathcal{S} - \mathbf{z}_\mathcal{S}) \\ & + \boldsymbol{\gamma}_{\mathcal{S}^c}^\mathsf{T}(\mathbf{Dx})_{\mathcal{S}^c} + \frac{\rho}{2}\|(\mathbf{Dx})_\mathcal{S} - \mathbf{z}_\mathcal{S}\|_2^2 + \frac{\rho}{2}\|(\mathbf{Dx})_{\mathcal{S}^c}\|_2^2. \end{aligned}$$

Let k denote the iteration index. The ADMM method for solving problem (9) is summarized in Algorithm 3, which carries out the following steps at each iteration:

Step 1 Compute $\mathbf{z}_\mathcal{S}^{k+1}$ with fixed $\mathbf{x}^k$ and $\boldsymbol{\gamma}^k$ by solving:

$$\min_{\mathbf{z}_\mathcal{S}} \lambda\|\mathbf{z}_\mathcal{S}\|_1 + \frac{\rho}{2}\|(\mathbf{Dx}^k)_\mathcal{S} - \mathbf{z}_\mathcal{S} + \frac{1}{\rho}\boldsymbol{\gamma}_\mathcal{S}^k\|_2^2, \tag{10}$$

which has a unique closed-form solution. Let $\boldsymbol{\mu} = (\mathbf{Dx}^k)_\mathcal{S} + \frac{1}{\rho}\boldsymbol{\gamma}_\mathcal{S}^k$. We can obtain the solution of (10) relying on a soft-thresholding operator $\mathbf{z}_\mathcal{S}^{k+1} = S_{\lambda/\rho}(\boldsymbol{\mu})$, where the i-th component of $\mathbf{z}_\mathcal{S}^{k+1}$ is calculated by $z_{\mathcal{S}i}^{k+1} = S_{\lambda/\rho}(\boldsymbol{\mu})_i =$

$\mathrm{sign}(\mu_i)\max\{|\mu_i| - \frac{\lambda}{\rho}, 0\}$. To simplify the representation, we define a temporary variable $\mathbf{z}' \in \mathbb{R}^l$, and let $\mathbf{z}'_{\mathcal{S}} = \mathbf{z}^{k+1}_{\mathcal{S}}$ and $\mathbf{z}'_{\mathcal{S}^c} = \mathbf{0}$.

Step 2 Update $\mathbf{x}^{k+1}$ by solving $\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{z}^{k+1}_{\mathcal{S}}, \boldsymbol{\gamma}^k)$ w.r.t. $\mathbf{x}$:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\rho}{2}\|\mathbf{D}\mathbf{x} - \mathbf{z}' + \frac{1}{\rho}\boldsymbol{\gamma}^k\|_2^2, \tag{11}$$

which is quadratic in $\mathbf{x}$. The minimizer can be achieved from the normal equation $\left[\mathbf{A}^\mathsf{T}\mathbf{A} + \rho(\mathbf{D}^\mathsf{T}\mathbf{D})\right]\mathbf{x} = \mathbf{A}^\mathsf{T}\mathbf{y} + \rho\mathbf{D}^\mathsf{T}(\mathbf{z}' - \frac{1}{\rho}\boldsymbol{\gamma}^k)$, which can be solved efficiently by a conjugate gradient algorithm.

Step 3 Update the dual variable $\boldsymbol{\gamma}$:

$$\boldsymbol{\gamma}^{k+1} = \boldsymbol{\gamma}^k + \rho(\mathbf{D}\mathbf{x}^{k+1} - \mathbf{z}'). \tag{12}$$

Algorithm 3: ADMM for Solving Subproblem (9)

Input: Observation $\mathbf{y}$, parameter λ and ρ, initialization of image $\mathbf{x}^0$, index set $\mathcal{S}_t$.

Initialize $\boldsymbol{\gamma}^0 = \mathbf{0}$. Set iteration number as $k = 0$;
while *Stopping conditions are not achieved* **do**
  Compute $\mathbf{z}^{k+1}_{\mathcal{S}}$ by solving (10);
  Generate $\mathbf{z}'$ by letting $\mathbf{z}'_{\mathcal{S}} = \mathbf{z}^{k+1}_{\mathcal{S}}$ and $\mathbf{z}'_{\mathcal{S}^c} = \mathbf{0}$;
  Compute $\mathbf{x}^{k+1}$ by solving problem (11);
  Update $\boldsymbol{\gamma}^{k+1}$ according to (12), and let $k = k + 1$;

Matching Pursuit for Generalized LASSO

Based on Proposition 1 and Algorithm 3, we implement our algorithm in the primal form as in Algorithm 4, which is referred to as *matching pursuit for generalized LASSO (MPGL)*. Due to $\boldsymbol{\alpha}^* = \boldsymbol{\xi}^*$ and $\boldsymbol{\xi}^* = \mathbf{y} - \mathbf{A}\mathbf{x}^*$, we can reconstruct the dual variable and find the most violated constraint even though the subproblem is solved in its primal form. In Algorithm 4, since no $\boldsymbol{\tau}$ is involved at the initial stage, e.g. $\mathcal{S}^0 = \emptyset$, we initialize $\mathbf{x}^0$ via solving (9) given $\mathcal{S} = \emptyset$. For example, for the fused LASSO problem, $\mathbf{x}^0 = \mathbf{0}$.

Algorithm 4: Matching Pursuit for Generalized LASSO.

Input: Observation $\mathbf{y}$, parameter λ and ρ.

Initialize $\mathbf{x}^0$, $\boldsymbol{\alpha}^0 = \mathbf{y} - \mathbf{A}\mathbf{x}^0$, $\mathcal{S}^0 = \emptyset$, iteration index $t = 0$;
while *Stopping conditions are not achieved* **do**
  Find the most violate constraint via Algorithm 2, and record the corresponding indices into $\mathcal{C}^{t+1}$;
  Let $\mathcal{S}^{t+1} = \mathcal{S}^t \cup \mathcal{C}^{t+1}$;
  Subproblem optimization: Solve subproblem (9) via the ADMM in Algorithm 3, obtaining $\mathbf{x}^{t+1}$. Let $\boldsymbol{\alpha}^{t+1} = \mathbf{y} - \mathbf{A}\mathbf{x}^{t+1}$, and $t = t + 1$;

The complexity of MPGL mainly includes two parts: 1) Finding the most violated constraint needs to calculate $\mathbf{A}^\mathsf{T}\boldsymbol{\alpha}$ and solve problem (8), thus these take $O(nm + nl)$ complexity. Fortunately, MPGL only needs to conduct this step several times. 2) Solving the subproblem in (9) takes $O(nm + nl)$ complexity, dominated by Step 2 in Algorithm 3. Since only a small set of active elements are involved, it converges much faster than that with all the elements.

Setting of Parameter κ

In general, a very large κ can decrease the iteration number of MPGL, and a very small κ helps to prevent the non-support elements being selected. To achieve the balance between efficiency and performance, we seek to provide a strategy for the setting of κ. Recall that κ reflects a rough knowledge of the support number in $\mathbf{D}\mathbf{x}$, i.e. $K = \|\mathbf{D}\mathbf{x}\|_0$. As K is unknown, we first obtain a $\boldsymbol{\beta}^0$ via solving problem (8) with the initialization $\boldsymbol{\alpha}^0$. Following the strategy in (Tan, Tsang, and Wang 2015), we set κ as the number of elements in $\boldsymbol{\beta}^0$ larger than $\zeta\|\boldsymbol{\beta}^0\|_\infty$. In practice, $\zeta \geq 0.5$ works well.

Early Stopping and Solution Bias

Many existing methods are very sensitive to the choice of λ and face a dilemma – a too big λ produces a sparse solution, but underfits the data; a too small λ fits the data well, but induces a rather dense solution contradicting the prior knowledge of the data. They often have to intensively search for a proper λ. This is impractical for the large real world problems, which are time consuming even for running the experiment once. Due to the nature of our optimization scheme, we can bypass the dilemma to a large degree, by proposing an early stopping condition below.

Recall that only κ nonzero elements in $\mathbf{D}\mathbf{x}$ are activated in each iteration of MPGL, the value of $\|\mathbf{D}\mathbf{x}\|_1$ increases from 0 gradually. By defining $g(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{D}\mathbf{x}\|_1$, the algorithm can be stopped using the relative function value difference: $\left[g(\mathbf{x}^{t-1}) - g(\mathbf{x}^t)\right]/g(\mathbf{x}^0) \leq \epsilon$, where ϵ is a small tolerance value.

The early stopping condition helps to obtain a sparse $\mathbf{D}\mathbf{x}$, thus our method works very well for a large range of λ, which significantly reduces the expenses for hyper-parameter tuning. In practice, we may choose a small λ to reduce the risk of biased solution.

Convergence Analysis

Before providing the convergence analysis, we first give the following lemma.

Lemma 1. *Let* $(\boldsymbol{\alpha}^*, \theta^*)$ *be the global optimal solution of* (5)*, and* $\{\theta_t\}_{t=1}^T$ *be a sequence of* θ *obtained in the iterations in Algorithm 1, where* $T = |\boldsymbol{\Lambda}|$ *denotes the possibly maximum iteration number. As the iteration index* t *increases,* $\{\theta_t\}$ *is monotonically increasing. And there is* $\theta_t \leq \theta^*$.

Based on Lemma 1, the following theorem shows that MPGL converges to a global solution of (5).

Theorem 1. *Let* $T = |\boldsymbol{\Lambda}| \leq \infty$*, and* $\{\boldsymbol{\alpha}_t, \theta_t\}_{t=1}^T$ *be the sequence generated by Algorithm 1. Assume that both the most violated constraint finding problem and the subproblem* (6) *in Algorithm 1 can be addressed. Algorithm 1 terminates at the* t*-th iteration after a finite number of iterations with a global optimal solution* $\{\boldsymbol{\alpha}_t, \theta_t\}$.

Experiments

We conduct the experiments on both synthetic data and real-world data to verify the effectiveness of our algorithm (MPGL). We implement the main scheme of MPGL in Algorithm 4 in Matlab, and that of the Algorithm 3 in C++ with a mex-file interface. All the experiments are performed on an Intel i5 CPU with 8G RAM.

As discussed before, there has been some algorithms for GL or some special cases of that, such as FL or GFL. In the experiments, we compare the proposed MPGL with following state-of-the-art GL algorithms:

- SLEP (Liu et al. 2009; Liu, Yuan, and Ye 2010): A package implemented in C and Matlab for 1-dimensional FL.
- "genlasso" (Tibshirani et al. 2011): A package implemented in R for GFL, which is limited to the cases $n \leq m$.
- fGFL (Xin et al. 2014): An algorithm for GFL, which is implemented in C++ and Matlab.
- CVX (Grant, Boyd, and Ye 2008): A general convex optimization package, which can be adapted for GL.

Experiments on Synthetic Data

We first investigate the efficiency of the proposed method on synthetic data under two settings: FL and GFL guided by a general graph. We tested on the problem with different dimensions: $n = 10^2, 10^3, 5 \times 10^3, 8 \times 10^3, 10^4$, and 1.5×10^5. For each n, we first generate a Gaussian random matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m = n$, and a Gaussian noise vector $\mathbf{n} \in \mathbb{R}^m$ from $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ with $\sigma = 0.05$. Then we generate the ground truth $\mathbf{x}^* \in \mathbb{R}^n$ in which $x_i^* = 0.5, \forall i \in \{n/2 - n/20, ..., n/2 + n/20\}$ and $x_i^* = 0$ for others and let $\mathbf{y} = \mathbf{A}\mathbf{x}^* + \mathbf{n}$. For FL, we let $\mathbf{D} = [\mathbf{I}^\top, \mathbf{Q}(\mathrm{TV})^\top]^\top$. For GFL, we generate a graph $G = \{V, E\}$ with $V = [n]$ and $2.2 \times n$ randomly sampled edges in E, and let $\mathbf{D} = [\mathbf{I}^\top, \mathbf{Q}(G)^\top]^\top$. In this experiment, we fix the parameter λ in (2) as 0.005 and test all algorithms on the data with different values of n.

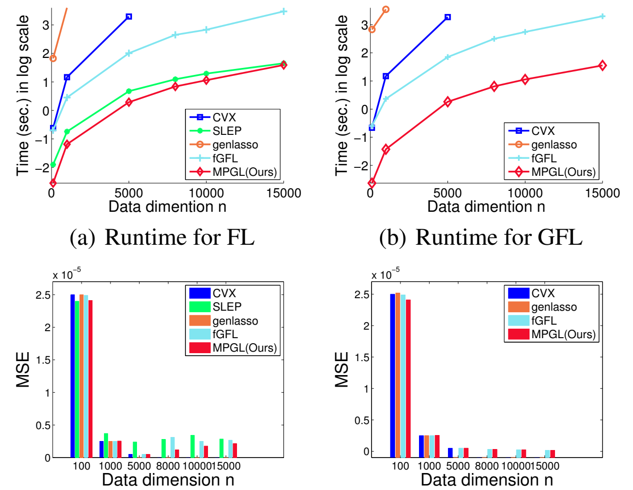
Figure 1 shows the comparison of runtime (second in log scale) and mean squared error (MSE). Furthermore, as shown in Figure 1(a) and 1(b), the proposed MPGL has the best efficiency on both the FL and GFL problem. To be fair, the running time is obtained by terminating the algorithms when they achieve similar objective values. As the result, the MSE values recorded in Figure 1(c) and 1(d) of the solutions stay around the same level. Note that because SLEP is limited for FL, its results on GFL is absent. Moreover, partial results of CVX and "genlasso" for high dimensional data are missed because of limitation of memory and running time.

Experiments on Medical Data Classification

In this section, we evaluate the proposed MPGL on a real medical data classification task. Specifically, least square loss and fused LASSO regularizer are used for the GL methods. In this case, each row of $\mathbf{A}$ denotes a training sample, and each element in $\mathbf{y}$ denotes the corresponding class label.

The experiments are performed on four real medical datasets for binary classification:

- `ArrayCGH` dataset (Stransky et al. 2006) contains array comparative genomic hybridization profiles of 57 bladder tumor samples and 2,385 features in each sample. Samples will be classified into 12 samples of Grade 1 and 45 samples of higher grade (2 or 3).
- `Leukemia` dataset (Golub et al. 1999) consists of gene expression data for 7,129 genes of 72 samples including 47 samples for acute lymphocytic leukemia and 25 samples for acute myelogenous leukemia.
- `Brain Tumor` dataset (Nutt et al. 2003) contains gene expression data for 12,626 genes of 50 brain tumor samples (21 classic glioblastomas against 29 non-classic).
- `Prostate Cancer` dataset (Petricoin et al. 2002) is obtained by protein mass spectrometry. It consists of 15,154 features of 132 patients (63 healthy against 69 with prostate cancer).



(a) Runtime for FL (b) Runtime for GFL

(c) MSE for FL (d) MSE for GFL

Figure 1: Experimental results on synthetic data for FL and GFL. The runtime is shown in log scale for visualization.

We compare the proposed MPGL method with the above-mentioned GL methods and support vector machine (SVM) (Fan et al. 2008) with linear kernel as a supervised learning baseline. The comparison with "genlasso" is skipped because the number of samples is smaller than the number of features ($m < n$) for all datasets. For each method, the parameters are tuned for the best performance.

We record the leave-one-out accuracy and the runtime for one training process in Table 1. Form Table 1, among the GFL algorithms, the proposed MPGL algorithm achieves the fastest training speed and comparable or better accuracy than others. Specifically, the computational costs of MPGL are lower than SLEP, which is specifically optimized for the FL problem. Considering the testing accuracy, the GL methods achieves better results than SVM, benefiting from the better biological interpretation from $\|\mathbf{D}\mathbf{x}\|_1$ (Tibshirani and Wang 2008). Moreover, we can observe that the de-biasing property of MPGL gives a better generalization.

Table 1: Experimental results on medical data. Both GL methods and non-GL methods are considered in comparison.

Type	Method	`ArrayCGH`		`Leukemia`		`Brain Tumor`		`Prostate Cancer`	
		Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)
non-GL	SVM	77.19%	0.0108	83.33%	0.0248	96.00%	0.0297	97.73%	0.1154
GL	CVX	87.72%	1.3213	94.44%	7.1228	**98.00%**	11.444	**98.48%**	45.560
	SLEP	85.96%	0.5043	93.06%	1.9877	96.00%	3.0616	**98.48%**	6.5361
	fGFL	87.72%	8.4336	47.22%	21.200	**98.00%**	54.111	94.70%	184.24
	MPGL	**89.47%**	**0.2555**	**95.83%**	**0.9371**	**98.00%**	**2.4868**	**98.48%**	**4.0370**

Table 2: Experimental results on image deconvolution. Higher PSNR and SSIM values reflect better quality.

Type	Method	`Cameraman`		`House`		`Lena`		`Pepper`	
		PSNR/SSIM	Time (s)	PSNR/SSIM	Time (s)	PSNR/SSIM	Time (s)	PSNR/SSIM	Time (s)
–	Input	20.64/0.6268	–	23.64/0.6803	–	23.48/0.6935	–	22.99/0.7057	–
non-GL	BM3D	25.91/0.7599	1.7405	26.02/0.6535	1.7511	24.69/0.7467	1.7979	22.95/0.6556	1.8921
	FTVd	20.36/0.3597	0.1697	20.17/0.3124	0.1654	19.78/0.3833	0.1765	18.87/0.3424	0.1754
	IRLS	25.11/0.8134	2.3217	30.74/0.8353	2.3322	28.88/0.8591	2.3211	29.96/0.8895	2.3018
GL	CVX	22.70/0.7563	1568.0	29.03/0.8279	1673.6	27.83/0.8251	1661.6	28.61/0.8519	1816.7
	fGFL	26.59/0.8270	1930.2	30.42/0.8343	2078.1	**29.65/0.8742**	1725.6	30.01/0.8902	1963.2
	MPGL	**27.25/0.8327**	**2.2560**	**31.84/0.8505**	**2.2732**	29.03/0.8691	**2.4068**	**30.41/0.8888**	**2.2689**

Figure 2: Visual quality comparison of image deconvolution on `Cameraman`. A local part is enlarged for visualization. Our result has a clearer background and sharper details than the others.

Experiments on Image Restoration

In this experiment, we apply the GL to an image restoration task – image deconvolution, which aims to remove the blur produced during camera exposure. Let $\mathbf{x}^*$ be an unknown sharp image in vector form, $\mathbf{A}$ be a convolution matrix corresponding to blur, the observed blurry image $\mathbf{y}$ can be modeled as $\mathbf{y} = \mathbf{A}\mathbf{x}^* + \mathbf{n}$, where $\mathbf{n}$ denotes random noise (Wang et al. 2008; Gong et al. 2016). Given $\mathbf{y}$ and $\mathbf{A}$, the task to recover $\mathbf{x}$ is referred to as *image deconvolution*.

A dataset consisting of four examples is studied, in which all images are 256×256 ($n = 65{,}536$). In this experiment, we consider the motion blur with length 15 and angel $45°$, and $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ with noise level[2] $\sigma = 0.005$. Since natural images are usually piecewise smooth and non-sparse, we let $\mathbf{D}$ represent the 2-dimensional TV (Wang et al. 2008), where $\mathbf{D} = [\mathbf{Q}(G_v)^\top, \mathbf{Q}(G_h)^\top]^\top$ with two graphs G_v and G_h recording the 2-dimensional neighborhood information on vertical and horizontal direction, respectively.

[2]The noise level σ is corresponding to the image in scale $[0, 1]$.

Apart from the GL methods fGFL and CVX, we also compare MPGL with some non-GL state-of-the-art image deconvolution methods: BM3D (Dabov et al. 2008), FTVd (Wang et al. 2008) and IRLS (Levin et al. 2007). We set $\lambda = 0.001$ for GL methods, and use default settings for other methods. Peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) (Wang et al. 2004) are used to measure the quality of the results. Note that our implementation directly performs the convolution to avoid extra computational costs on handling the large matrix $\mathbf{A}$. As shown in Table 2, compared to other GL methods, the proposed MPGL achieves comparable or even better image quality and more than 1,000 times faster computational speed. For the images with sparse gradients (e.g. images with significant edges and flat areas), our method can effectively detect the nonzero gradients, achieving significant improvement comparing to previous methods. For the images with many middle frequency components (e.g. gradually changing image intensities in `Pepper`), $\mathbf{D}\mathbf{x}^*$ are much denser, which makes our method activate many nonzero elements in $\mathbf{D}\mathbf{x}$ to fit the data, leading to less significant improvement. Even so, our method still achieves better or comparable performance. Though BM3D and FTVd have low computational costs, their performances are worse than others since they assume the periodic boundary condition. Figure 2 shows that our result has more sharp and natural details and suffers less from the ringing artifacts than others.

Conclusion

We have proposed a matching pursuit method for solving the generalized LASSO problems efficiently. By introducing a binary vector to indicate the nonzero elements in $\mathbf{D}\mathbf{x}$, we formulate the GL as a QCLP problem and solve it via a cutting plane algorithm. MPGL is guaranteed to converge to a global optimum. The proposed algorithm with early stopping helps to reduce the solution bias. Unlike many existing methods, MPGL can be applied to the general formulation of GL. Empirical studies on several datasets from different applications show the superior performance of the proposed MPGL over comparable state-of-the-art methods.

Acknowledgments

This work is in part supported by National Natural Science Foundation of China (No. 61231016, 61602185, 61301193), Australian Research Council grants (DP140102270, DP160100703, DP160103710) and China Scholarship Council.

References

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*.

Dabov, K.; Foi, A.; Katkovnik, V.; and Egiazarian, K. 2008. Image restoration by sparse 3d transform-domain collaborative filtering. In *Electronic Imaging 2008*. International Society for Optics and Photonics.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*.

Friedman, J.; Hastie, T.; Höfling, H.; and Tibshirani, R. 2007. Pathwise coordinate optimization. *The Annals of Applied Statistics*.

Golub, T. R.; Slonim, D. K.; Tamayo, P.; Huard, C.; Gaasenbeek, M.; Mesirov, J. P.; Coller, H.; Loh, M. L.; Downing, J. R.; Caligiuri, M. A.; et al. 1999. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*.

Gong, D.; Tan, M.; Zhang, Y.; van den Hengel, A.; and Shi, Q. 2016. Blind image deconvolution by automatic gradient activation. In *CVPR*.

Grant, M.; Boyd, S.; and Ye, Y. 2008. Cvx: Matlab software for disciplined convex programming.

Hoefling, H. 2010. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*.

Huang, J.; Zhang, T.; and Metaxas, D. 2011. Learning with structured sparsity. *Journal of Machine Learning Research*.

Kim, S., and Xing, E. P. 2010. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*.

Kim, S.-J.; Koh, K.; Boyd, S.; and Gorinevsky, D. 2009. \ell_1 trend filtering. *SIAM Review*.

Levin, A.; Fergus, R.; Durand, F.; and Freeman, W. T. 2007. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*.

Liu, J.; Ji, S.; Ye, J.; et al. 2009. Slep: Sparse learning with efficient projections. *Arizona State University*.

Liu, J.; Yuan, L.; and Ye, J. 2010. An efficient algorithm for a class of fused lasso problems. In *ACM SIGKDD*.

Liu, J.; Yuan, L.; and Ye, J. 2013. Guaranteed sparse recovery under linear transformation. In *ICML*.

Mutapcic, A., and Boyd, S. 2009. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods & Software*.

Nutt, C. L.; Mani, D.; Betensky, R. A.; Tamayo, P.; Cairncross, J. G.; Ladd, C.; Pohl, U.; Hartmann, C.; McLaughlin, M. E.; Batchelor, T. T.; et al. 2003. Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Research*.

Pee, E., and Royset, J. O. 2011. On solving large-scale finite minimax problems using exponential smoothing. *Journal of Optimization Theory and Applications*.

Petricoin, E. F.; Ornstein, D. K.; Paweletz, C. P.; Ardekani, A.; Hackett, P. S.; Hitt, B. A.; Velassco, A.; Trucco, C.; Wiegand, L.; Wood, K.; et al. 2002. Serum proteomic patterns for detection of prostate cancer. *Journal of the National Cancer Institute*.

Rudin, L. I.; Osher, S.; and Fatemi, E. 1992. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*.

Stransky, N.; Vallot, C.; Reyal, F.; Bernard-Pierrot, I.; De Medina, S. G. D.; Segraves, R.; De Rycke, Y.; Elvin, P.; Cassidy, A.; Spraggon, C.; et al. 2006. Regional copy number–independent deregulation of transcription in cancer. *Nature Genetics*.

Tan, M.; Tsang, I. W.; Wang, L.; and Zhang, X. 2012. Convex matching pursuit for large-scale sparse coding and subset selection. In *AAAI*.

Tan, M.; Tsang, I. W.; and Wang, L. 2014. Towards ultrahigh dimensional feature selection for big data. *Journal of Machine Learning Research*.

Tan, M.; Tsang, I. W.; and Wang, L. 2015. Matching pursuit lasso part i: Sparse recovery over big dictionary. *IEEE Transactions on Signal Processing*.

Tibshirani, R., and Wang, P. 2008. Spatial smoothing and hot spot detection for cgh data using the fused lasso. *Biostatistics*.

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*.

Tibshirani, R. J.; Taylor, J. E.; Candes, E. J.; and Hastie, T. 2011. The solution path of the generalized lasso. *The Annals of Statistics*.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*.

Wang, Y.; Yang, J.; Yin, W.; and Zhang, Y. 2008. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*.

Wang, J.; Fan, W.; and Ye, J. 2015. Fused lasso screening rules via the monotonicity of subdifferentials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xin, B.; Kawahara, Y.; Wang, Y.; and Gao, W. 2014. Efficient generalized fused lasso and its application to the diagnosis of alzheimer's disease. In *AAAI*.

Zhang, C.-H., and Huang, J. 2008. The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics* 1567–1594.

Zhang, C.-H. 2010. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics* 894–942.

Zhu, Y. 2016. An augmented admm algorithm with application to the generalized lasso problem. *Journal of Computational and Graphical Statistics*.