# Efficient Generalized Fused Lasso with its Application to the Diagnosis of Alzheimer's Disease

**Bo Xin*, Yoshinobu Kawahara†, Yizhou Wang* and Wen Gao***
*Nat'l Engineering Lab for Video Technology, Key Lab. of Machine Perception (MoE),
Sch'l of EECS, Peking University, Beijing, 100871, China
†Institute of Scientific and Industrial Research, Osaka University, Osaka, 567-0047, Japan

**Abstract**

Generalized Fused Lasso (GFL) penalizes variables with $L_1$ norms both on the variables and their pairwise differences. GFL is useful when applied to data of which prior information is expressed on a graph. However, the existing algorithms for GFL incur high computational cost and do not scale to high dimensionality. In this paper, we propose a fast and scalable algorithm for GFL. Based on the fact that the fusion penalty is the Lovász extension of a cut function, we show that the key building block of the optimization is equivalent to recursively solving graph cut problems. We then solve GFL efficiently via a parametric flow algorithm. Runtime comparison demonstrates a significant speed-up over the existing algorithms of GFL. Leveraging the scalability of the proposed algorithm, we formulate the diagnosis of Alzheimer's Disease as GFL. Experiments show that not only is the diagnosis performance promising, but the selected critical voxels are well structured – being connected, consistent in cross-validation and in accordance with clinical prior knowledge.

## Introduction

Learning with sparsity-inducing norms has been one of the main focuses in machine learning, and also has been successfully applied to a variety of applications. As is well known, learning with the $L_1$ penalty, such as Lasso, encourages variables to be sparse (Tibshirani 1996). Recently, this approach has been extended to explore structures on variables, called structured sparse learning. A variety of norms for different structures and efficient algorithms for solving corresponding optimizations have been proposed, e.g.,(Huang, Zhang, and Metaxas 2011; Bach et al. 2012). Fused Lasso is one of such variants, where pairwise differences between variables are penalized using $L_1$ norm, which encourages successive variables to be similar (Tibshirani et al. 2005).

### Generalized Fused Lasso

Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be a set of samples, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Also, denote by $\mathbf{X} \in \mathbb{R}^{d\times N}$ and $\mathbf{y} \in \mathbb{R}^N$ the concatenations of $\mathbf{x}_i$ and $y_i$, respectively. Then, we start from the definition of (1D) Fused Lasso, which has been first proposed by (Tibshirani et al. 2005) and is formulated as

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{y} - \mathbf{X}^T\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{i=1}^{d} |\beta_i| + \lambda_2 \sum_{i=2}^{d} |\beta_i - \beta_{i-1}|, \tag{1}$$

where $\boldsymbol{\beta} \in \mathbb{R}^d$ and $\lambda_1, \lambda_2 \geq 0$. Also, here, the variables (i.e., $\boldsymbol{\beta}$) are supposed to have some meaningful ordering (e.g. forming a chain structure). Due to the $L_1$ penalties on both single variable and consecutive pairs, solutions tend to be sparse and smooth, *i.e.*, consecutive variables tend to be similar. The third term is usually called the "fusion penalty".

Classical fused lasso method (Eq.(1)) is proposed to pursue sparse segments on a chain of variables. Thus, a natural generalization of 1D Fused Lasso is to encourage smoothness over neighboring variables on a general graph.

Assume that we have a graph $G = (V, E)$ with nodes $V$ and edges $E$, where each variable corresponds to a node on the graph. Then, such a generalization is given as

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{y} - \mathbf{X}^T\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{i=1}^{d} |\beta_i| + \lambda_2 \sum_{(i,j)\in E} |\beta_i - \beta_j|. \tag{2}$$

In general, Generalized Fused Lasso (GFL) is referred to as Eq. (2).

In this paper, we propose to solve a further generalization of the above problem with an arbitrary smooth convex loss $l : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} l(\boldsymbol{\beta}) + \lambda_1 \sum_{i=1}^{d} |\beta_i| + \lambda_2 \sum_{(i,j)\in E} |\beta_i - \beta_j|, \tag{3}$$

which we also refer to as GFL in this paper. The benefit of such a generalization will soon be clear in the application.

### Existing Algorithms

The first algorithm solving Fused Lasso, *i.e.* Eq. (1), proposed by (Tibshirani et al. 2005), is based on the two-phase active set algorithm SQOPT (Gill, Murray, and Saunders 1999). This algorithm can be extended to GFL and implemented using an off-the-shelf convex optimization solver. However it usually does not scale to high dimensional problems. Proximal gradient methods, such as the fast iterative shrinkage-thresholding algorithm (FISTA) (Beck and

Teboulle 2009), solves a convex problem whose objective consists of both smooth and non-smooth parts. Applying FISTA, Liu et. al. proposed to solve Eq. (1) by designing specific proximal operators (*i.e.*, generalized projections) (Liu, Yuan, and Ye 2010). Although their algorithm is efficient and scalable for the 1D case, in principle, it cannot be extended to GFL. Friedman et. al. has proposed a pathwise coordinate descent algorithm for a special case of Eq. (2) (Friedman et al. 2007), where the design matrix $\mathbf{X}$ is the identity matrix. The reported efficiency of the algorithm is impressive, but as claimed in (Friedman et al. 2007), this algorithm has no guarantee to find exact solutions for general problems. In (Tibshirani and Taylor 2011), a solution path algorithm is proposed for Eq. (2). This algorithm solves it for all possible parameters ($\lambda$'s) by finding "critical" changing points in a dual problem, which however, tend to be very dense in large problems.

In this paper, we propose an efficient and scalable algorithm for solving GFL. Using proximal methods (FISTA), the key building block of our algorithm is the Fused Lasso signal approximation (FLSA). Based on the fact that the fusion penalty is the Lovász extension of a cut function, we apply a parametric flow algorithm and then the soft-thresholding method to finally efficiently solve FLSA. The proposed algorithm can find an exact solution of GFL and also can be implemented with a stable and efficient parametric flow solver. The runtime experiments show that the speed of the proposed algorithm is competitive with the state-of-the-art 1D Fused Lasso algorithms and significantly outperforms the existing algorithms for GFL, especially on high dimensional data.

### Motivation: the Diagnosis of Alzheimer's Disease

Our work is motivated by a challenging real-world application, the diagnosis of Alzheimer's Disease (AD) problem. This is usually formulated as a classification task, where structural magnetic resonance images (sMRI) of human brains are used as input. Because of its practical benefit, this problem is attracting more and more researches from fields like medical image analysis and machine learning. Since the dimensionality of brain images can be as high as millions, whereas the number of available samples is usually limited, e.g. in hundreds, a proper regularization is needed.

Critical brain voxels should be both sparse and spatially assembled into several early damaged anatomical regions. Existing methods either assume independence between voxels (*e.g.* univariate selection (Dai et al. 2012)), or use Volume Of Interest (VOI) (Zhou et al. 2011) as processing unit which loses much pathological information, and therefore might not be sensitive enough for early diagnosis.

Considering the structure of a brain sMRI as a 3D grid graph, we propose to formulate the diagnosis of AD as GFL. However, the existing algorithms do not scale enough to solve this problem in feasible time. We demonstrate the effectiveness of the proposed algorithm, which not only solves it within limited memory and time, but achieves promising classification accuracy, which is among the state-of-the-art. Also, the selected voxels are well structured – being connected, consistent in cross-validation and in accordance with clinical prior knowledge.

## Efficient Optimization for GFL

In this section, we propose an efficient and scalable optimization for GFL. We first introduce the fast iterative shrinkage thresholding algorithm (FISTA), which is applied to solve GFL by iteratively calculating *proximal operators*. For GFL, we show that the computation of the proximal operator is formulated as one of the Fused Lasso signal approximations (FLSA). We then propose a parametric optimization formulation to efficiently solve FLSA: we introduce a soft-threshold strategy to remove the sparse term, transform FLSA to a minimum-norm-point problem under a submodular constraint, prove its equivalence to recursively solving graph cuts and solve this problem via a parametric flow method.

### Proximal Methods and Fused Lasso Signal Approximation

In smooth convex problems, it was shown (Nesterov 2004) that there exists a gradient method with $O(1/k^2)$ complexity, which is an "optimal" first order method in the sense of (Nemirovsky and Yudin 1983). By extending the Nesterov's method to the general case with non-smooth terms, FISTA achieves the same complexity (Beck and Teboulle 2009). FISTA has been applied to various sparse learning problems, e.g., (Beck and Teboulle 2009; Bach 2010), and for 1D Fused Lasso, e.g. (Liu, Yuan, and Ye 2010). We also apply FISTA to solving GFL in this paper.

It is known that for the optimization of any smooth objective function $l(\boldsymbol{\beta})$ can be achieved by a gradient method; the updating rule of $\boldsymbol{\beta}$ at each iteration, in general, can be seen as a proximal regularization to the linearization of $l()$ at the previous $\boldsymbol{\beta}_k$ ($k$ is the iteration index), (Polëiìak 1987) *i.e.*,

$$\begin{aligned}\boldsymbol{\beta}_{k+1} = \operatorname*{argmin}_{\boldsymbol{\beta}} \Big\{ & l(\boldsymbol{\beta}_k) \\ & + \langle \boldsymbol{\beta} - \boldsymbol{\beta}_k, \nabla l(\boldsymbol{\beta}_k) \rangle + \frac{L}{2} \|\boldsymbol{\beta} - \boldsymbol{\beta}_k\|_2^2 \Big\},\end{aligned} \tag{4}$$

where $L > 0$ is the Lipschitz constant of $l()$.

Let us denote the regularization terms in Eq. (3) as

$$\Omega(\boldsymbol{\beta}) = \lambda_1 \sum_{i=1}^{d} |\beta_i| + \lambda_2 \sum_{(i,j)\in E} |\beta_i - \beta_j|.$$

When there is a non-smooth part $\Omega(\boldsymbol{\beta})$ in the objective function of GFL (Eq.(3)), using FISTA, the updating rule turns into

$$\begin{aligned}\boldsymbol{\beta}_{k+1} = \operatorname*{argmin}_{\boldsymbol{\beta}} \Big\{ & l(\boldsymbol{\beta}_k) + \langle \boldsymbol{\beta} - \boldsymbol{\beta}_k, \nabla l(\boldsymbol{\beta}_k) \rangle \\ & + \frac{L}{2} \|\boldsymbol{\beta} - \boldsymbol{\beta}_k\|_2^2 + \Omega(\boldsymbol{\beta}) \Big\},\end{aligned} \tag{5}$$

where the minimization admits a unique solution. After some simple manipulation of Eq.(5) (ignoring some constant

term of $\beta_k$), we have

$$\boldsymbol{\beta}_{k+1} = \operatorname*{argmin}_{\boldsymbol{\beta}} \left\{ \Omega(\boldsymbol{\beta}) + \frac{L}{2} \left\| \boldsymbol{\beta} - (\boldsymbol{\beta}_k - \frac{1}{L} \nabla l(\boldsymbol{\beta}_k)) \right\|_2^2 \right\}. \tag{6}$$

Thus, the key to solve Eq. (3) is how efficiently we solve Eq. (6). The optimization Eq. (6) can be rewritten as

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \frac{1}{2} \|\boldsymbol{\beta} - \mathbf{z}\|_2^2 + \lambda_1 \sum_{i=1}^{d} |\beta_i| + \lambda_2 \sum_{(i,j) \in E} |\beta_i - \beta_j|, \tag{7}$$

where $\mathbf{z} = \boldsymbol{\beta}_k - \frac{1}{L} \nabla l(\boldsymbol{\beta}_k)$, and $\lambda_1$ and $\lambda_2$ are scaled from Eq. (3) by $L$. Problem (7) is equivalent to FLSA defined in (Friedman et al. 2007; Tibshirani and Taylor 2011).

## An Efficient Solution to FLSA by Parametric Flow

To our best knowledge, there lacks an efficient method to solve FLSA for high-dimensional problems in the literature. In this paper we propose an efficient solution to the minimization problem Eq.(7) using a parametric flow method.

**$L_1$ Soft-Thresholding** Let first denote the objective in Eq. (7) by $f(\boldsymbol{\beta}; \lambda_1, \lambda_2)$ and $\boldsymbol{\beta}_{\lambda_2}^{\lambda_1} = \arg\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}, \lambda_1, \lambda_2)$. We then introduce the following lemma (Friedman et al. 2007; Liu, Yuan, and Ye 2010) :

**Lemma 1.** *For any $\lambda_1, \lambda_2 \geq 0$, we have*

$$\boldsymbol{\beta}_{\lambda_2}^{\lambda_1} = \operatorname{sign}(\boldsymbol{\beta}_{\lambda_2}^0) \odot \max(|\boldsymbol{\beta}_{\lambda_2}^0| - \lambda_1, 0), \tag{8}$$

*where $\odot$ is an element-wise product operator.*

From Eq. (8), a solution to Eq. (7) can be obtained through a soft-thresholding process (Donoho and Johnstone 1995), which is often applied to solve Lasso problems.

Based on this lemma, we first solve the following problem:

$$\boldsymbol{\beta}_{\lambda_2}^0 = \operatorname*{argmin}_{\boldsymbol{\beta}} \frac{1}{2} \|\boldsymbol{\beta} - \mathbf{z}\|_2^2 + \lambda_2 \sum_{(i,j) \in E} |\beta_i - \beta_j|. \tag{9}$$

Then, using Eq.(8), a soft-threshold process to $\boldsymbol{\beta}_{\lambda_2}^0$ *w.r.t* $\lambda_1$, we have a solution to Eq. (7).

**Minimum-Norm-Point Problem under Submodular Constraint** According to Lemma 1, we can leave the $L_1$ term aside and focus on the fusion term (*i.e.*, Problem (9)) to calculate the proximal operator. However, since the second term of Eq.(9) is non-smooth and non-separable *w.r.t.* $\boldsymbol{\beta}$, its optimization is still not trivial [1]. In order to develop an efficient algorithm for Problem (9), we consider transform Problem (9) to a minimum-norm-point (MNP) problem under submodular constraint.

First, we prove the following lemma, which describes a relation between the fusion penalty and a cut function.

[1](Goldfarb and Yin 2009) has applied a parametric flow algorithm to solve (9), which is nontrivial to be extended to exact GFL, mainly because of both the theoretic gap we bridge over and their discretized formulation and optimization, where $\boldsymbol{\beta},\ \mathbf{z} \in \mathbb{Z}_+^d$.

Let denote by $\mathcal{V} := \{1, \ldots, d\}$ a finite set. Given a set of non-negative weights $w : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_+$, a cut function of a set $\mathcal{S} \subseteq \mathcal{V}$ is defined by

$$f_c(\mathcal{S}) = \sum_{i \in \mathcal{S}, j \in \mathcal{V} \setminus \mathcal{S}} w_{ij},\ (\mathcal{S} \subseteq \mathcal{V}).$$

**Lemma 2.** *The fusion term $\sum_{(i,j) \in E} |\beta_i - \beta_j|$ is equivalent to the Lovász extension of a cut function.*

*Proof.* We define the weights of the cut function $w_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise, which results in a cut function

$$f_c(\mathcal{S}) = \sum_{(i,j) \in E, i \in \mathcal{S}, j \in \mathcal{V} \setminus \mathcal{S}} 1.$$

We denote $(j_1, ..., j_d)$ is a decreasing ordering index such that $\beta_{j_1} \geq \cdots \geq \beta_{j_d}$ and $\mathcal{U}_k := \{j_1, ...j_k\}$ is a subset. Applying the definition of the Lovász extension (Fujishige 2005), we have

$$\begin{aligned}
\hat{f}_c(\boldsymbol{\beta}) &= \sum_{k=1}^{d} \beta_{j_k} (f_c(\mathcal{U}_k) - f_c(\mathcal{U}_{k-1})) \\
&= \sum_{k=1}^{d} (- \sum_{(i,j_k) \in E, i \in \mathcal{U}_k} \beta_{j_k} + \sum_{(i,j_k) \in E, i \in \mathcal{V} \setminus \mathcal{U}_k} \beta_{j_k}) \\
&= \sum_{(i,j) \in E, \beta_i \geq \beta_j} (\beta_i - \beta_{j_k}) + \sum_{(i,j) \in E, \beta_i < \beta_j} (\beta_j - \beta_i) \\
&= \sum_{(i,j) \in E} |\beta_i - \beta_j|.
\end{aligned}$$

Similarly, for arbitrary non-negative weights $w_{ij}$, the Lovász extension of the cut function can be shown equal to $\sum_{(i,j) \in E} w_{ij} |\beta_i - \beta_j|$. □

With this lemma, we can rewrite Eq. (9) as

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \frac{1}{2} \|\boldsymbol{\beta} - \boldsymbol{z}\|_2^2 + \lambda_2 \cdot \hat{f}_c(\boldsymbol{\beta}), \tag{10}$$

Since a cut function is submodular, this optimization problem can be transformed to a minimum-norm-point problem under a submodular constraint:

**Proposition 3.** *Problem* (10) *is equivalent to the following problem:*

$$\min_{\mathbf{t} \in \mathbb{R}^d, \mathbf{t} \in B(f_c - \lambda_2^{-1} \mathbf{z})} \|\mathbf{t}\|_2^2, \tag{11}$$

*where $B(\bullet)$ is the base polyhedron of a submodular function $\bullet$. A minimizer $\boldsymbol{\beta}^*$ of Problem* (10) *is obtained by $\boldsymbol{\beta}^* = -\lambda_2 \boldsymbol{t}^*$, where $\boldsymbol{t}^*$ is a minimizer of Problem* (11)*.*

*Proof.* From the definition of the Lovász extension, we have $\hat{f}_c(\boldsymbol{\beta}) = \max_{\mathbf{s} \in B(f_c)} \boldsymbol{\beta}^T \mathbf{s}$ (Fujishige 2005). Hence, we have

$$\begin{aligned}
&\min_{\boldsymbol{\beta}} \frac{1}{2} \|\boldsymbol{\beta} - \mathbf{z}\|_2^2 + \lambda_2 \cdot \hat{f}_c(\boldsymbol{\beta}) \\
&= \min_{\boldsymbol{\beta}} \max_{\mathbf{s} \in B(f_c)} \frac{1}{2} \|\boldsymbol{\beta} - \mathbf{z}\|_2^2 + \lambda_2 \cdot \boldsymbol{\beta}^T \mathbf{s} \\
&= \max_{\mathbf{s} \in B(f_c)} -\frac{1}{2} \|\lambda_2 \mathbf{s} - \mathbf{z}\|_2^2 + \frac{1}{2} \|\mathbf{z}\|_2^2 \ \text{(since } \boldsymbol{\beta}^* = \mathbf{z} - \lambda_2 \mathbf{s}) \\
&\Leftrightarrow \min_{\mathbf{s} \in B(f_c)} \|\mathbf{s} - \lambda_2^{-1} \mathbf{z}\|_2^2
\end{aligned}$$

Let $\mathbf{t} = \mathbf{s} - \lambda_2^{-1}\mathbf{z}$ and, with the basic property of the base polyhedron of a submodular function, we have

$$\min_{\mathbf{s}\in B(f_c)} \|\mathbf{s} - \lambda_2^{-1}\mathbf{z}\|_2^2 \Leftrightarrow \min_{\mathbf{t}\in B(f_c - \lambda_2^{-1}\mathbf{z})} \|\mathbf{t}\|_2^2$$

From the derivation, it follows $\boldsymbol{\beta}^* = -\lambda_2 \mathbf{t}^*$. □

For general submodular functions, Problem (11) is solvable using submodular minimization algorithms, such as the minimum-norm-point (MNP) algorithm (Fujishige, Hayashi, and Isotani 2006). However, since the fastest complexity of submodular minimization is $O(d^5EO+d^6)$ (Orlin 2009), this approach to our high-dimensionality scenario is infeasible in practice.

**Parametric Graph Cut** We utilize a parametric property of the MNP problem and apply a parametric flow algorithm, which can run much more efficiently, to solve Problem.(11).

A set function $g(\mathcal{S}) = f_c(\mathcal{S}) - \lambda_2^{-1}\mathbf{z}(\mathcal{S})$ in Eq. (11) is the sum of a cut function and a modular function, which is still submodular (but, not necessarily, non-decreasing). Thus, Problem (11) is a special case of a separable convex minimization problem under submodular constraints (Nagano and Aihara 2012), which can be solved by parametric optimization (if the submodular function is non-deceasing). We will describe how to satisfy the non-deceasing requirement later in Lemma.4 & 5.

For a parameter $\alpha \geq 0$, we define a set function $g_\alpha(\mathcal{S}) = g(\mathcal{S}) - \alpha \cdot \mathbf{1}(\mathcal{S})$. If $g$ is a non-decreasing submodular function, there exists $l+1$ ($\leq d$) subsets

$$\mathcal{S}^* = \{(\emptyset =)\, \mathcal{S}_0 \subset \mathcal{S}_1 \subset \ldots \subset \mathcal{S}_l\, (= \mathcal{V})\},$$

and $l+1$ subintervals of

$$R_0 = [0, \alpha_1),\ R_1 = [\alpha_1, \alpha_2),\ \ldots,\ R_l = [\alpha_l, \infty),$$

such that, for each $j \in \{0, \ldots, l\}$, $\mathcal{S}_j$ is the unique maximal minimizer of $g_\alpha(\mathcal{S})$ for all $\alpha \in R_j$ (Nagano and Aihara 2012). Then, the unique optimal solution $\mathbf{t}^* \in \mathbb{R}^d$ to Problem (11) is determined by, for each $i \in \mathcal{V}$ with $i \in \mathcal{S}_{j+1} \setminus \mathcal{S}_j$ ($j \in \{1, \ldots, l-1\}$),

$$t_i^* = \frac{f_c(\mathcal{S}_{j+1}) - f_c(\mathcal{S}_j)}{\mathbf{1}(\mathcal{S}_{j+1} \setminus \mathcal{S}_j)}. \tag{12}$$

That is, by computing the unique maximal minimizer of $g_\alpha$ for some appropriately chosen $\alpha$'s, we can find all $\mathcal{S}_j$'s and therefore $\mathbf{t}^*$. One possible option of finding all "proper" $\alpha$'s would be to apply the decomposition algorithm (Fujishige 2005; Nagano and Aihara 2012).

As stated above, in order to apply the above procedure, $g$ has to be a non-decreasing function. We first introduce two lemmas from (Nagano and Kawahara 2013) to apply the above to $g$:

**Lemma 4.** *For any $\gamma \in \mathbb{R}$ and a submodular function $f$, $\mathbf{t}^*$ is an optimal solution to $\min_{\mathbf{t}\in B(f)} \|\mathbf{t}\|_2^2$ if and only if $\mathbf{t}^* + \gamma\mathbf{1}$ is an optimal solution to $\min_{\mathbf{t}\in B(f+\gamma\mathbf{1})} \|\mathbf{t}\|_2^2$.*

**Lemma 5.** *Set $\gamma = \max_{i=1,\ldots,d}\{0, f(V\setminus\{i\}) - f(V)\}$, then $f + \gamma\mathbf{1}$ is a nondecreasing submodular function.*

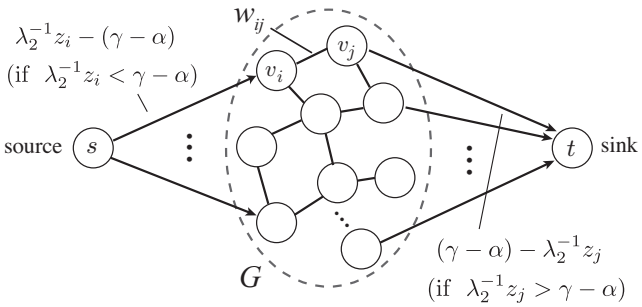



Figure 1: Construction of an $s$-$t$ graph for Problem (13). Given a graph $G = (V, E)$ for GFL, the capacities on edges are defined as in the following: $c(v_i, v_j) = w_{ij}$ $(i, j \in V)$, $c_{s,v_i} = \lambda_2^{-1}z_i - (\gamma - \alpha)$ if $\lambda_2^{-1}z_i < \gamma - \alpha$ or $c_{s,v_i} = 0$ otherwise $(i \in V)$, and $c_{v_i,t} = (\gamma - \alpha) - \lambda_2^{-1}z_i$ if $\lambda_2^{-1}z_i > \gamma - \alpha$ or $c_{v_i,t} = 0$ otherwise $(i \in V)$, where $c_{s,v_i}$ and $c_{v_i,t}$ mean the capacities on source-to-node and node-to-sink edges.

Applying Lemma 5 to our case with $f := g$, we solve

$$\min_{\mathcal{S}\subset\mathcal{V}} f_c(\mathcal{S}) - \lambda_2^{-1}\mathbf{z}(\mathcal{S}) + (\gamma - \alpha)\cdot\mathbf{1}(\mathcal{S})\ (:= g'_\alpha(\mathcal{S})). \tag{13}$$

and then apply Lemma 4 to obtain a solution of the original problem. Due to the specific form of Problem (13), we can solve it as an easier problem:

**Proposition 6.** *For any cut function $f_c$, Problem* (13) *is equivalent to an $s$-$t$ cut problem with the $s$-$t$ graph defined as in Figure 1.*

*Proof.* Problem (13) is composed of modular terms and a submodular pairwise term. This is a typical $\mathcal{F}^2$ type energy function (Kolmogorov and Zabin 2004), which is known to be "graph-representable" and to be able to be minimized via graph-cut algorithms. Hence, by following the construction way of an $s$-$t$ graph by (Kolmogorov and Zabin 2004), we can solve Problem (13) by solving a $s$-$t$ cut on this graph. Note that a detailed proof of a similar problem can be found also in (Azencott et al. 2013). □

As a consequence, we can obtain a solution to Eq. (11) by solving $s$-$t$ cut problems for some different $\alpha$'s:

$$\text{Find minimum } s\text{-}t \text{ cuts w.r.t. Eq. (13) for } \alpha \geq 0. \tag{14}$$

However, since the parameter $\alpha$ affects only edges from the source node or to the sink node, we do not need to search $\alpha$'s that give different solutions. That is, as can be seen from the construction of the $s$-$t$ graph, the capacities on source-to-node or node-to-sink edges have the following properties: (i) the capacities on source-to-node edges are non-decreasing functions of $\alpha$; (ii) the capacities on node-to-sink edges are non-increasing functions of $\alpha$; (iii) the capacities on node-to-node edges are constant with respect to $\alpha$. For such cases, it is known that the parametric flow algorithm by (Gallo, Grigoriadis, and Tarja 1989) (GGT algorithm) can be applied to find all solutions for all $\alpha \in \mathbb{R}$. Hence, we can obtain the sequence of solutions to Problem (13) for different $\alpha$'s by just applying GGT algorithm, which runs in $O(d|E|\log(d^2/|E|)$ at the worst case.

## Runtime Comparison

We investigated the efficiency of the proposed algorithm, abbreviated as fGFL (fast Generalized Fused Lasso). All experiments were carried out on an Intel(R) Xeon(R) E5-2687 CPU at 3.10GHz with 64G memory. Our implementation of FLSA is written in C++ and FISTA in Matlab.[2]

As mentioned above, several algorithms have been proposed for FLSA and GFL. Here we compare the proposed fGFL with the following state-of-the-arts:

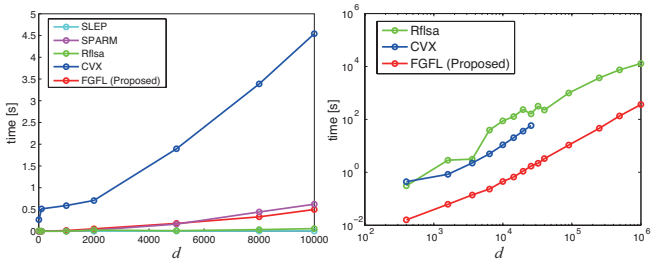
- SLEP package (Liu, Ji, and Ye 2009; Liu, Yuan, and Ye 2010): Implemented with Matlab and C for 1D Fused Lasso and 1D FLSA.
- SPAMS (Mairal et al. 2011): Implemented with C for 1D Fused Lasso and 1D FLSA.
- "flsa" R package: Implemented with R for general FLSA and includes accelerated implementations for 1D and 2D (grid) FLSA.
- "genlasso" R package (Tibshirani and Taylor 2011): Implemented with R for generalized Fused Lasso and includes accelerated implementations for 1D and 2D (grid) Fused Lasso (Note that these are limited to the cases of $N \geq d$.).
- CVX (Grant, Boyd, and Ye 2008): This is a general convex optimization toolbox. We used its general-use optimizer for GFL and FLSA.

We compared the algorithms in the application to 1D and 2D case of FLSA defined in Eq. (7) and GFL defined in Eq. (2). Note that the proposed fGFL can be applied to a more general case Eq. (3), where most existing algorithms do not work. We will demonstrate the advantage of Eq. (3) and our solution in the application to the AD problem.

We generated data for the runtime comparison in the following way. First, for 1D Fused Lasso *i.e.* Eq. (1), we set parameter $\boldsymbol{\beta}$ as: $\beta_i = 0.5$ for $i \in \{d/2-d/20, \ldots, d/2+d/20\}$ and 0 otherwise. For 2D Fused Lasso, we set $\beta_{i,j} = 0.5$ for $i, j \in \{d/2 - d/20, \ldots, d/2 + d/20\}$ and 0 otherwise. For FLSA defined in Eq. (7), we set $\mathbf{z} = \boldsymbol{\beta} + 0.05\mathbf{e}$, where $\mathbf{e}$ is a noise vector drawn from the standard normal distribution. For GFL as in Eq. (2), we generated $N = d$ samples (because "genlasso" cannot solve Eq. (2) when $N < d$): $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i = \boldsymbol{\beta}^T \mathbf{x}_i + 0.05 e_i$, $\mathbf{x}_i$ and $e_i$ for $i = 1, \ldots, N$ are drawn from the standard normal distribution. We fixed $\lambda_1, \lambda_2 = 0.1$ and applied the algorithm for different dimension $d$'s to compare the runtime. The graphs in Figure 2 and 3 show the runtimes by the algorithms.

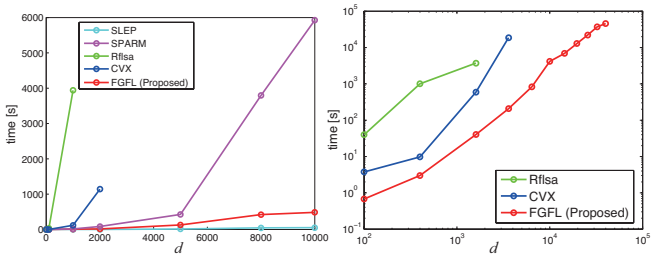
The algorithm using the standard optimizer (e.g., CVX) needs to handle the huge difference matrix $\mathbf{D} \in \mathbb{R}^{d \times |E|}$ for high-dimensional problems. This results in a memory shortage. The number of "critical" points found by "genlasso" explodes in high-dimensional problems, so we used the setting of "maxsteps=10,000" (this means "genlasso" will find at most 10,000 critical points). Both reasons explain for some missing comparisons in the Figure 2 and 3. Nevertheless, as illustrated, in 1D cases, our algorithm is not the fastest but competitive with the faster ones. In general cases of GFL (e.g. 2D), our algorithm ran the fastest, with tens to hundreds times speed-up.

[2]The codes can be found on the co-authors homepage.




(a) 1dflsa (b) 2dflsa

Figure 2: FLSA Runtime comparison (in seconds) by different algorithms along dimensionality $d$.




(a) 1dfl (b) 2dfl

Figure 3: GFL Runtime comparison (in seconds) by different algorithms along dimensionality $d$.

## Application to Alzheimer's Disease Problem

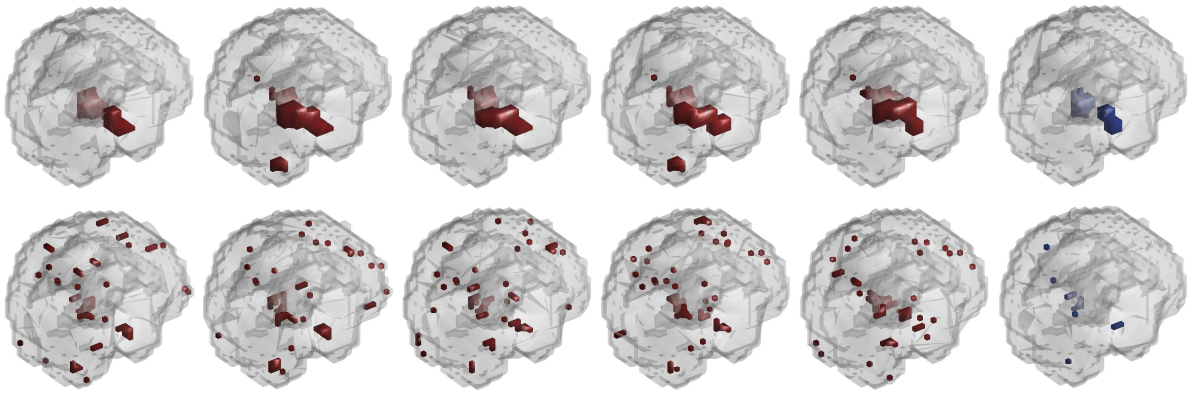
Alzheimer's Disease data used in this experiment were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database[3]. We used 1.5T MRI images of 62 AD patients, 71 NCs (healthy controls) and 141 MCIs (Mild Cognitive Impairment), 54 of them converted to AD ($\text{MCI}_C$) and 87 others did not ($\text{MCI}_S$). Preprocessing was performed following DARTEL VBM pipeline (Ashburner 2007). 2,873 $8\times8\times8$ mm$^3$ sizes voxels with values larger than 0.2 in the mean grey matter population template are used as features.

In the diagnosis of AD, two fundamental issues are AD/NC classification and MCI conversion prediction (namely $\text{MCI}_C$/$\text{MCI}_S$ classification). Let $\mathbf{x}_i \in \mathbb{R}^d$ be subject's sMRI features and $y_i = \{0, 1\}$ be subjects' disease status (AD/NC or $\text{MCI}_C$/$\text{MCI}_S$). Since our algorithm is applicable for smooth convex loss, we used logistic regression loss for such a classification task and formulate the problem as GFL in the following way

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d, c \in \mathbb{R}} \sum_{i=1}^{N} \log\left(1 + \exp\left(-y_i(\boldsymbol{\beta}^T \mathbf{x}_i + c)\right)\right) + \lambda \Omega(\boldsymbol{\beta}). \tag{15}$$

Notice that, even if we adopt the least square loss as in Eq. (2), other existing algorithms are not feasible in practice.

[3]http://adni.loni.ucla.edu



(a) fold 1 (b) fold 3 (c) fold 5 (d) fold 7 (e) fold 9 (f) overlap

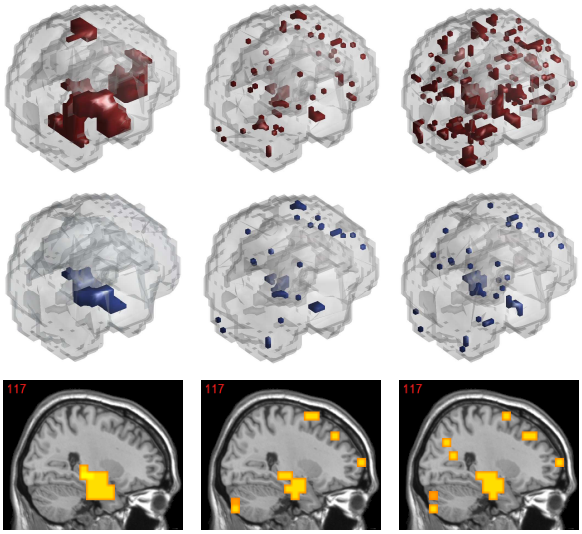
Figure 4: Consistency of selected voxels over different folds of cross validation. The results of 5 different folds are shown in (a)-(e) respectively and the overlapped voxels of all 10 folds are shown in (f). The top row illustrates the results from GFL and the bottom row shows the results from $L_1$. The percentages of the overlapped voxels are: GFL(66%) vs. $L_1$(22%).

Table 1: Comparison of classification accuracies in the Alzheimer's Disease problem.

| Task | LR | SVM | LR+$L_1$ | LR+GFL |
|---|---|---|---|---|
| AD/NC | 80.45% | 82.71% | 81.20% | **84.21%** |
| MCI | 63.83% | 67.38% | 68.79% | **70.92%** |

We compared GFL to logistic regression (LR), Support Vector Machine (SVM), and logistic regression with $L_1$ regularizer. The obtained classification accuracies with 10-fold cross validation (CV) are shown in Table 1. As can be seen, GFL achieves the highest accuracy in both tasks. Though on the same dataset, works are not strictly comparable because different work might use a different part of the dataset. Nevertheless, our results are among the state-of-the-art. In (Cheng, Zhang, and Shen 2012), their best performance on MCI tasks is 69.4% while ours reaches 70.92%. In (Chu et al. 2012), using similar sample size, our performance on ADNC tasks is comparable with or better than all theirs (84.21% vs. 81-84%) and our performance on MCI tasks is much better (70.92% vs. 65%).

We applied GFL to all samples with the optimal parameter setting determined by CV. In Figure 5, we compare the selected voxels with non-structured sparsity (*i.e.* $L_1$). We see that the selected voxels by GFL cluster into several spatially connected regions, while the selected voxels by $L_1$ scatter around. We considered the voxels corresponding to the top 50 negative $\beta_i$'s as the most atrophied voxels and projected them onto a slice. We see that the selected voxels of GFL are concentrated in Hippocampus, ParaHippocampal gyrus, which are believed to be early damaged regions related to A.D. On the other hand, $L_1$ either selects less critical voxels or probably selects noisy voxels not in the early damaged regions (see Figure 5(b) and 5(c) for an illustration). The selected voxels by GFL are also much more consistent than those selected by $L_1$, the percentage of overlapped voxels across 10-fold of cross validation is GFL(66%) vs. $L_1$(22%).



(a) (b) (c)

Figure 5: Compare GFL with $L_1$. The top row illustrates the selected voxels in a 3D brain model, the mid row shows the top 50 atrophied voxels, the bottom row shows the projection onto a brain slice. (a) GFL (best accuracy 84.21%); (b) $L_1$ (best accuracy 81.20%); (c) $L_1$ (using a similar number of voxels as in GFL).

An illustration can be seen in Figure 4.

## Conclusions

In this paper, we proposed an efficient and scalable algorithm for generalized Fused Lasso. We showed that the proposed algorithm significantly outperform the existing ones. Leveraging the efficiency and scalability of the proposed algorithm, we formulated the diagnosis of AD as GFL. Not only does GFL achieves the state-of-the-art classification accuracies, but the selected critical voxels are well structured.

## Acknowledgements

We would like to thank for the support from the following research grants NSFC-61272027, NSFC-61121002, NSFC-61231010 and NSFC-61210005.

## References

Ashburner, J. 2007. A fast diffeomorphic image registration algorithm. *Neuroimage* 38(1):95–113.

Azencott, C.; Grimm, D.; Sugiyama, M.; Kawahara, Y.; and Borgwardt, K. 2013. Efficient network-guided multi-locus association mapping with graph cuts. *Bioinformatics* 29(13):i171–i179.

Bach, F.; Jenatton, R.; Mairal, J.; and Obozinski, G. 2012. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning* 4(1):1–106.

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

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

Cheng, B.; Zhang, D.; and Shen, D. 2012. Domain transfer learning for mci conversion prediction. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012*. Springer. 82–90.

Chu, C.; Hsu, A.-L.; Chou, K.-H.; Bandettini, P.; and Lin, C. 2012. Does feature selection improve classification accuracy? impact of sample size and feature selection on classification using anatomical magnetic resonance images. *Neuroimage* 60(1):59–70.

Dai, Z.; Yan, C.; Wang, Z.; Wang, J.; Xia, M.; Li, K.; and He, Y. 2012. Discriminative analysis of early alzheimer's disease using multi-modal imaging and multi-level characterization with multi-classifier (m3). *Neuroimage* 59(3):2187–2195.

Donoho, D., and Johnstone, I. 1995. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association* 90(432):1200–1224.

Friedman, J.; Hastie, T.; Höfling, H.; and Tibshirani, R. 2007. Pathwise coordinate optimization. *The Annals of Applied Statistics* 1(2):302–332.

Fujishige, S.; Hayashi, T.; and Isotani, S. 2006. The minimum-norm-point algorithm applied to submodular function minimization and linear programming. Technical Report RIMS-1571, Research Institute for Mathematical Sciences, Kyoto University.

Fujishige, S. 2005. *Submodular Functions and Optimization*. Elsevier, 2 edition.

Gallo, G.; Grigoriadis, M.; and Tarja, R. 1989. A fast parametric maximum flow algorithm and applications. *SIAM Journal of Computing* 18(1):30–55.

Gill, P.; Murray, W.; and Saunders, M. 1999. User's guide for SNOPT 5.3: A fortran package for large-scale nonlinear programming. Technical report, University of California, San Diego.

Goldfarb, D., and Yin, W. 2009. Parametric maximum flow algorithms for fast total variation minimization. *SIAM Journal on Scientific Computing* 31(5):3712–3743.

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

Huang, J.; Zhang, T.; and Metaxas, D. 2011. Learning with structured sparsity. *Journal of Machine Learning Research* 12:3371–3412.

Kolmogorov, V., and Zabin, R. 2004. What energy functions can be minimized via graph cuts? *IEEE Trans. on Pattern Analysis and Machine Intelligence* 26(2):147–159.

Liu, J.; Ji, S.; and Ye, J. 2009. SLEP: Sparse learning with efficient projections.

Liu, J.; Yuan, L.; and Ye, J. 2010. An efficient algorithm for a class of fused Lasso problems. In *Proc. of the 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 323–332.

Mairal, J.; Jenatton, R.; Obozinski, G.; and Bach, F. 2011. Convex and network flow optimization for structured sparsity. *Journal of Machine Learning Research* 12:2681–2720.

Nagano, K., and Aihara, K. 2012. Equivalent of convex minimization problems over base polytopes. *Japan Journal of Industrial and Applied Mathematics* 29:519–534.

Nagano, K., and Kawahara, Y. 2013. Structured convex optimization under submodular constraints. In *in Proc. of the 29th Ann. Conf. on Uncertainty in Artificial Intelligence (UAI'13)*, 459–468.

Nemirovsky, A., and Yudin, D. 1983. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons.

Nesterov, Y. 2004. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer.

Orlin, J. 2009. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming* 118:237–251.

Polëiìak, B. 1987. *Introduction to Optimization*. Optimization Software, Publications Division (New York).

Tibshirani, R., and Taylor, J. 2011. The solution path of the generalized lasso. *Annals of Statistics* 39(3):1335–1371.

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* 67(1):91–108.

Tibshirani, R. 1996. Regression shrinkage and selection via the Lasso. *Journal of Royal Statistical Society B* 58(1):267–288.

Zhou, J.; Yuan, L.; Liu, J.; and Ye, J. 2011. A multi-task learning formulation for predicting disease progression. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 814–822. ACM.