Efficient Generalized Fused Lasso and Its Application to the Diagnosis of Alzheimer's Disease

Bo Xin*, Yoshinobu Kawahara†, Yizhou Wang* and Wen Gao*
*National Engineering Laboratory for Video Technology, Key Laboratory of Machine Perception,
School 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 based both on the variables and their pairwise differences. GFL is useful when applied to data where prior information is expressed using a graph over the variables. However, the existing GFL algorithms incur high computational costs and they do not scale to high-dimensional problems. In this study, we propose a fast and scalable algorithm for GFL. Based on the fact that 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 parametric graph-cut problems. Thus, we use a parametric flow algorithm to solve GFL in an efficient manner. Runtime comparisons demonstrated a significant speed-up compared with the existing GFL algorithms. By exploiting the scalability of the proposed algorithm, we formulated the diagnosis of Alzheimer's disease as GFL. Our experimental evaluations demonstrated that the diagnosis performance was promising and that the selected critical voxels were well structured i.e., connected, consistent according to cross-validation and in agreement with prior clinical knowledge.

Introduction

Learning with sparsity-inducing norms has been one of the main focuses of machine learning and it has been applied successfully to a variety of applications. It is well known that learning with the L_1 penalty, such as lasso, results in sparse variables (Tibshirani 1996). Recently, this approach was extended to explore structures of variables, which is called structured sparse learning. A variety of norms for different structures and efficient algorithms solving the corresponding optimizations have been proposed, such as (Huang, Zhang, and Metaxas 2011; Bach et al. 2012). Fused lasso is one of these variants, where pairwise differences between variables are penalized using the L_1 norm, which results in successive variables being 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 $y_i \in \mathbb{R}$. $\mathbf{X} \in \mathbb{R}^{d\times N}$ and $\mathbf{y} \in \mathbb{R}^N$ denote the concatenations of $\mathbf{x}_i$ and y_i, respectively. Then, we start from the definition of (1D) fused lasso, which was 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 $\lambda_1, \lambda_2 \geq 0$. Furthermore, the variables (*i.e.*, $\boldsymbol{\beta}$) are assumed to have a meaningful ordering (e.g., forming a chain structure). Due to the L_1 penalties on both single variables 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".

The classical fused lasso method (Eq.(1)) was proposed to pursue sparse segments on a chain of variables. Thus, a natural generalization of 1D fused lasso aims to promote 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, Eq. (2) is usually referred to as generalized fused lasso (GFL).

In the present study, we propose to solve a further generalization of the problem above 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 this generalization will soon be clear in the application.

Existing Algorithms

The first algorithm for solving fused lasso, i.e., Eq. (1), which was 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. In general, however, it does not scale to high-dimensional problems. Proximal gradient methods such as

the fast iterative shrinkage-thresholding algorithm (FISTA) (Beck and Teboulle 2009) solve convex problems where the objective comprises both smooth and non-smooth parts. Using 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, it cannot be extended to GFL in principle. Friedman et al. 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; however as suggested in (Friedman et al. 2007), this algorithm is not guaranteed to find exact solutions to general problems. In (Tibshirani and Taylor 2011), a solution path algorithm is proposed for Eq. (2). This algorithm solves for all possible parameters (λs) by finding critical changing points in a dual problem, although they tend to be very dense in large problems.

In the present study, 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 fusion penalty is the Lovász extension of a cut function, we apply a parametric flow algorithm and then the soft-thresholding method to solve the FLSA in an efficient manner. The proposed algorithm can find an exact solution to GFL and it can also be implemented with a stable and efficient parametric flow solver. Our runtime experiments demonstrate that the speed of the proposed algorithm is competitive with the state-of-the-art 1D fused lasso algorithms and it significantly outperforms existing GFL algorithms, especially with high-dimensional data.

Motivation: the Diagnosis of Alzheimer's Disease

The motivation of our research includes the diagnosis of Alzheimer's disease (AD), which is a challenging real-world application. This is usually formulated as a classification task, where structural magnetic resonance images (sMRI) of human brains are used as the inputs. Because of its practical benefit, this problem is increasingly attracting many researchers from various fields, such as medical image analysis and machine learning. The dimensionality of a brain image can be as high as millions whereas the number of available samples is usually limited, e.g., hundreds, thus appropriate regularization is required.

Critical brain voxels should be sparse and spatially assembled into several anatomical regions with early damage. Existing methods either assume independence between voxels (e.g., univariate selection (Dai et al. 2012)), or they use the volume of interest (VOI) (Zhou et al. 2011) as a processing unit, which loses much of the pathological information and might not be sufficiently sensitive for early diagnosis.

By 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 sufficiently well to solve this problem in feasible time. Thus, we demonstrate the effectiveness of the proposed algorithm, which solves the problem within limited memory and time, as well as yielding promising classification accuracy, which is similar to the state-of-the-art methods. The selected voxels are also well structured i.e., being connected, consistent according to cross-validation and in agreement with clinical prior knowledge.

Efficient Optimization for GFL

In this section, we propose an efficient and scalable optimization algorithm for GFL. First, we introduce FISTA, which is applied to solve GFL by iteratively calculating *proximal operators*. For GFL, we show that the computation of the proximal operator can be formulated as one of the FLSAs. We then propose a parametric optimization formulation to solve FLSA in an efficient manner, where we introduce a soft-threshold strategy to disgard the sparse term, transform the FLSA to a minimum-norm-point (MNP) problem under submodular constraints, prove its equivalence to recursively solving parametric graph-cut problems, and solve this problem using a parametric flow method.

Proximal Methods and FLSA

For 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 as per. (Nemirovsky and Yudin 1983). By extending 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 to 1D fused lasso, e.g., (Liu, Yuan, and Ye 2010). We also use FISTA to solve GFL in the present study.

It is known that the optimization of any smooth objective function $l(\boldsymbol{\beta})$ can be achieved using a gradient method, where the updating rule of $\boldsymbol{\beta}$ can be viewed as a proximal regularization to the linearization of $l()$ at the previous $\boldsymbol{\beta}_k$ (k is the iteration index), (Polëìak 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 changes the updating rule to

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

where the minimization admits a unique solution. After some simple manipulations of Eq.(5) (ignoring some con-

stant terms of β_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 solving Eq. (3) is how efficiently we can solve Eq. (6). The optimization in 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 λ_1 and λ_2 are scaled from Eq. (3) by L. Problem (7) is equivalent to the FLSA defined in (Friedman et al. 2007; Tibshirani and Taylor 2011).

An Efficient Solution to FLSA by Parametric Flow

To the best of our knowledge, there are no previous reports of an efficient method for solving the FLSA for high-dimensional problems. In the present study, therefore, we propose an efficient solution to the minimization problem Eq.(7) by using a parametric flow method.

L_1 Soft-Thresholding First, let us denote the objective in Eq. (7) by $f(\boldsymbol{\beta}; \lambda_1, \lambda_2)$ and $\boldsymbol{\beta}_{\lambda_2}^{\lambda_1} = \arg\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}, \lambda_1, \lambda_2)$. Then, we 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 using a soft-thresholding process (Donoho and Johnstone 1995), which is applied often for solving lasso problems.

First, based on this lemma, we 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* λ_1, we obtain a solution to Eq. (7).

Minimum-Norm-Point Problem under Submodular Constraints According to Lemma 1, we can neglect the L_1 term 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 nontrivial.[1] To develop an efficient algorithm for Problem (9), we consider a transformation of Problem (9) into an MNP problem under submodular constraints.

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

[1](Goldfarb and Yin 2009) used a parametric flow algorithm to solve (9), but it is nontrivial to be extended this to exact GFL, mainly because of both the theoretic gap that needs to be bridged and their discretized formulation and optimization, where $\boldsymbol{\beta}$, $\mathbf{z} \in \mathbb{Z}_+^d$.

Let $\mathcal{V} := \{1, \ldots, d\}$ denote a finite set. Given a set of non-negative weights $w : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}_+$, 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 $(j_1, ..., j_d)$ as 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. By 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 to equal $\sum_{(i,j) \in E} w_{ij} |\beta_i - \beta_j|$. □

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 into an MNP problem under submodular constraints.

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 that $\boldsymbol{\beta}^* = -\lambda_2 \mathbf{t}^*$. □

For general submodular functions, Problem (11) is solvable using submodular minimization algorithms, such as the MNP algorithm (Fujishige, Hayashi, and Isotani 2006). However, the known fastest time complexity of submodular minimization is $O(d^5 EO + d^6)$ (Orlin 2009), where EO is the cost for a function evaluation, thus this approach to high-dimensional problems is infeasible in practice.

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

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-decreasing). We describe how to satisfy the non-decreasing requirement in Lemmas 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 $i \in \mathcal{S}_{j+1} \setminus \mathcal{S}_j$ ($j \in \{1, \ldots, l-1\}$),

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

Thus, by computing the unique maximal minimizer of g_α for some appropriately selected αs, we can find all $\mathcal{S}_j$ and therefore $\mathbf{t}^*$. A possible option for finding all "appropriate" αs would be to apply the decomposition algorithm (Fujishige 2005; Nagano and Aihara 2012).

As stated above, g has to be a non-decreasing function in order to apply the above procedure. First, we 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.*

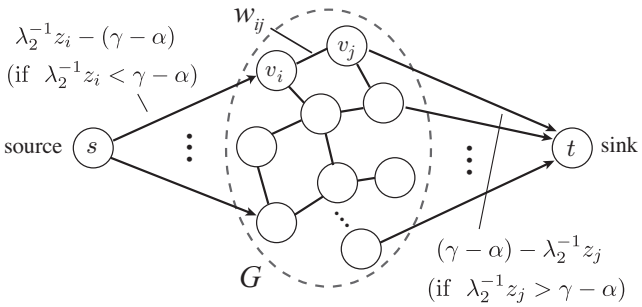


Figure 1: Construction of an s-t graph for Problem (13). Given a graph $G = (V, E)$ for GFL, the capacities on edges are defined as follows: $c(v_i, v_j) = w_{ij}$ $(i, j \in V)$, $c_{s,v_i} = \lambda_2^{-1}z_i - (\gamma - \alpha)$ if $\lambda_2^{-1}z_i < \gamma - \alpha$ or $c_{s,v_i} = 0$ otherwise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

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

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

Then we apply Lemma 4 to obtain a solution of the original problem. Owing to the specific form of Problem (13), we can solve it as an easier problem, as follows.

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

Proof. Problem (13) comprises 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 it can be minimized via graph-cut algorithms. Hence, by following the construction of an s-t graph according to (Kolmogorov and Zabin 2004), we can solve Problem (13) by solving an s-t cut on this graph. Note that a detailed proof of a similar problem can be found 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 α's:

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

However, since the parameter α only affects the edges from the source node or to the sink node, we do not need to search αs that yield different solutions. Thus, 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 α; (ii) the capacities on node-to-sink edges are non-increasing functions of α; (iii) the capacities on node-to-node edges are constant with respect to α. For such cases, it is known that the parametric flow algorithm reported by (Gallo, Grigoriadis, and Tarja 1989) (GGT algorithm) can be applied to find all solutions for all $\alpha \in \mathbb{R}$. Thus, we can obtain the sequence of solutions to Problem (13) for different αs by simply applying the GGT algorithm, which runs in $O(d|E|\log(d^2/|E|)$ as the worst case.

Runtime Comparison

We investigated the efficiency of the proposed algorithm, i.e., fast generalized fused lasso (fGFL). All of the experiments were performed using an Intel(R) Xeon(R) E5-2687 CPU at 3.10GHz with 64G memory. Our implementation of FLSA was written in C++ and that of 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-art algorithms:

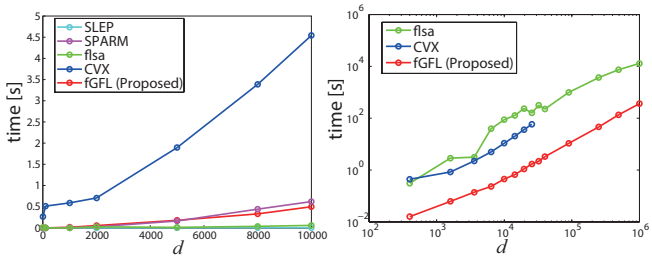
- SLEP package (Liu, Ji, and Ye 2009; Liu, Yuan, and Ye 2010): Implemented with Matlab and C for 1D fused lasso and 1D FLSA.
- SPAMS (Mairal et al. 2011): Implemented with C for 1D fused lasso and 1D FLSA.
- "flsa" R package: Implemented with R for general FLSA, which includes accelerated implementations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

We compared the application of the algorithms to 1D and 2D cases 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), for which most existing algorithms are not applicable. We demonstrate the advantage of Eq. (3) and our solution based on their application to the AD problem.

We generated data for the runtime comparison in the following manner. 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 to different dimension d to compare the runtime. The graphs in Figure 2 and 3 show the runtimes obtained using the algorithms.

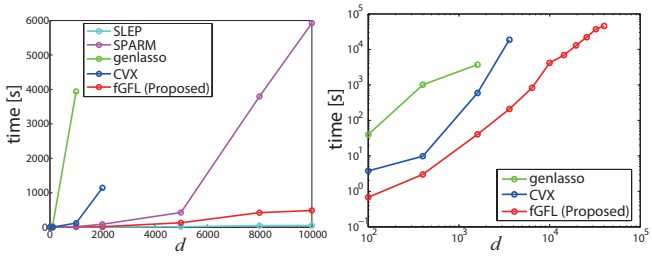
Algorithm that use the standard optimizer (e.g., CVX) need to handle the huge difference matrix $\mathbf{D} \in \mathbb{R}^{d \times |E|}$ for high-dimensional problems, which results in a memory shortage. The number of critical points found by "genlasso" significantly increases in high-dimensional problems, so we used the setting of "maxsteps=10,000" (i.e., "genlasso" will find a maximum of $10,000$ critical points). These explanations account for some of the missing comparisons in Figure 2 and 3. Nevertheless, as illustrated, in the 1D cases, our algorithm was not the fastest but it was competitive with the faster algorithms. In general cases of GFL (e.g. 2D), our algorithm was the fastest, with a speedup of tens to hundreds of times.

[2]The codes can be found on the co-author's homepage.



(a) 1dflsa (b) 2dflsa

Figure 2: FLSA Runtime comparison (in seconds) using different algorithms with variable dimensionality d.



(a) 1dfl (b) 2dfl

Figure 3: GFL Runtime comparison (in seconds) using different algorithms with variable dimensionality d.

Application to the AD Diagnosis Problem

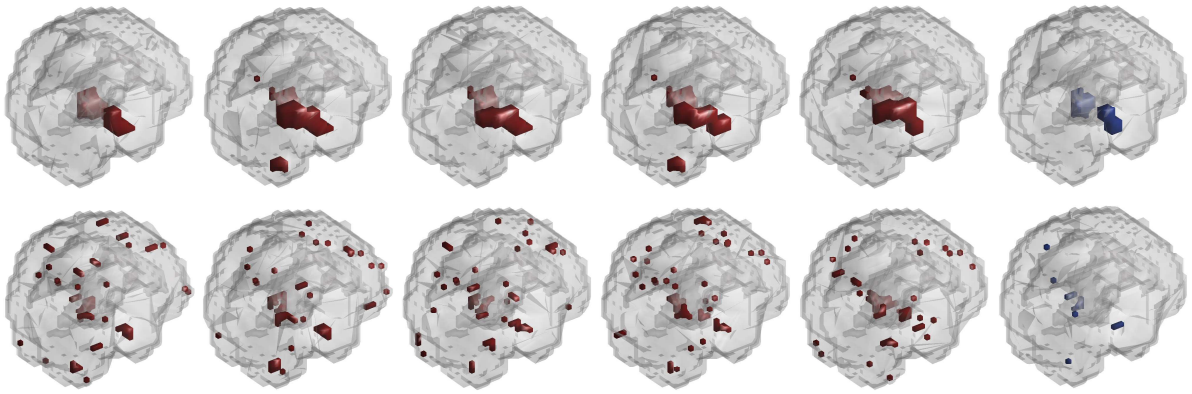
The AD data used in this experiment were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database[3]. We used 1.5T MRI of 62 AD patients, 71 NCs (healthy controls) and 141 patients with mild cognitive impairment (MCI), among whom 54 converted to AD (MCI_C) whereas 87 did not (MCI_S). Preprocessing was performed using the DARTEL VBM pipeline (Ashburner 2007). We used 2,873 $8 \times 8 \times 8$ mm^3 voxels with values larger than 0.2 in the mean grey matter population template as features.

During the diagnosis of AD, the two fundamental issues are AD/NC classification and MCI conversion prediction (i.e., MCI_C/MCI_S classification). Let $\mathbf{x}_i \in \mathbb{R}^d$ be a subject's sMRI features and $y_i = \{-1, 1\}$ be the subjects' disease status (AD/NC or MCI_C/MCI_S). Since our algorithm is applicable to any smooth convex loss, we used logistic regression loss for the classification task and formulated the problem as GFL in the following manner

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

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

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



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

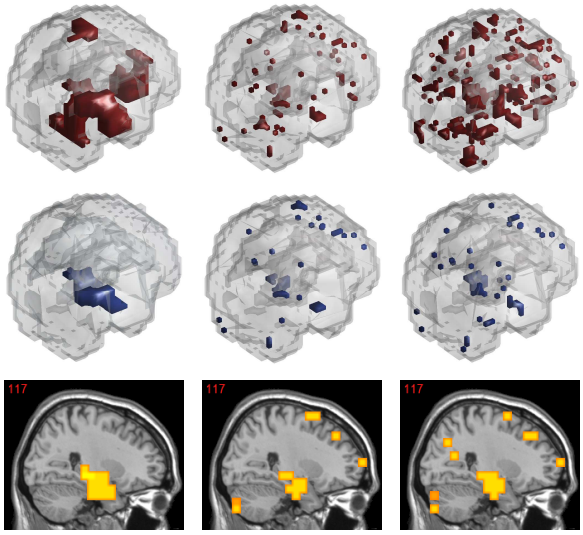
Figure 4: Consistency of selected voxels in different trials of cross-validations. The results of 5 different folds of cross-validations are shown in (a)-(e) and the overlapping voxels in all 10 folds are shown in (f). The top row shows the results for GFL and the bottom row shows the results for L_1. The percentages of the overlapping voxels were: GFL(66%) vs. L_1(22%).

Table 1: Comparison of the accuracy of AD classification.

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 an L_1 regularizer. The classification accuracies obtained based on a 10-fold cross validation (CV) are shown in Table 1, which shows that GFL yields the highest accuracy in both tasks. Furthermore, compared with other reported results, our performance are comparable with the state-of-the-art. In (Cheng, Zhang, and Shen 2012), the best performance with MCI tasks is 69.4% but our method reached 70.92%. In (Chu et al. 2012), a similar sample size is used as in our experiments, the performance of our method with ADNC tasks is comparable to or better than their reported results (84.21% vs. 81-84%) whereas our performance with MCI tasks is much better (70.92% vs. 65%).

We applied GFL to all the samples where the optimal parameter settings were determined by cross-validation. Figure 5 compares the selected voxels with non-structured sparsity (*i.e.* L_1), which shows that the voxels selected by GFL clustered into several spatially connected regions, whereas the voxels selected by L_1 were more scattered. We considered the voxels that corresponded to the top 50 negative β_i's as the most atrophied voxels and projected them onto a slice. The results show that the voxels selected by GFL were concentrated in hippocampus, parahippocampal gyrus, which are believed to be the regions with early damage that are associated with AD. By contrast, L_1 selected either less critical voxels or noisy voxels, which were not in the regions with early damage (see Figure 5(b) and 5(c) for details). The voxels selected by GFL were also much more consistent than those selected by L_1, where the percentages of overlapping voxels according to the 10-fold cross-validation were: GFL=66% vs. L_1=22%, as shown in Figure 4.



(a) (b) (c)

Figure 5: Comparison of GFL and L_1. The top row shows the selected voxels in a 3D brain model, the middle row shows the top 50 atrophied voxels, and the bottom row shows the projection onto a brain slice. (a) GFL (accuracy=84.21%); (b) L_1 (accuracy=81.20%); (c) L_1 (similar number of voxels as in GFL).

Conclusions

In this study, we proposed an efficient and scalable algorithm for GFL. We demonstrated that the proposed algorithm performs significantly better than existing algorithms. By exploiting the efficiency and scalability of the proposed algorithm, we formulated the diagnosis of AD as GFL. Our evaluations showed that GFL delivered state-of-the-art classification accuracy and the selected critical voxels were well structured.

Acknowledgments

We would like to express our thanks for support from the following research grants NSFC-61272027, NSFC-61121002, NSFC-61231010 and NSFC-61210005, and JSPS KAKENHI Grant Numbers 26280086 and 26120524.

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.