An Efficient Algorithm for a Class of Fused Lasso Problems

Jun Liu
Arizona State University
Tempe, AZ 85287
j.liu@asu.edu

Lei Yuan
Arizona State University
Tempe, AZ 85287
lei.yuan@asu.edu

Jieping Ye
Arizona State University
Tempe, AZ 85287
jieping.ye@asu.edu

ABSTRACT

The fused Lasso penalty enforces sparsity in both the coefficients and their successive differences, which is desirable for applications with features ordered in some meaningful way. The resulting problem is, however, challenging to solve, as the fused Lasso penalty is both non-smooth and non-separable. Existing algorithms have high computational complexity and do not scale to large-size problems. In this paper, we propose an Efficient Fused Lasso Algorithm (EFLA) for optimizing this class of problems. One key building block in the proposed EFLA is the Fused Lasso Signal Approximator (FLSA). To efficiently solve FLSA, we propose to reformulate it as the problem of finding an "appropriate" subgradient of the fused penalty at the minimizer, and develop a Subgradient Finding Algorithm (SFA). We further design a restart technique to accelerate the convergence of SFA, by exploiting the special "structures" of both the original and the reformulated FLSA problems. Our empirical evaluations show that, both SFA and EFLA significantly outperform existing solvers. We also demonstrate several applications of the fused Lasso.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining

General Terms

Algorithms

Keywords

Fused Lasso, ℓ_1 regularization, restart, subgradient

1. INTRODUCTION

The fused Lasso penalty introduced in [30] can yield a solution that has sparsity in both the coefficients and their successive differences. It has found applications in comparative genomic hybridization [25, 31], prostate cancer analysis [30], image denoising [8], and time-varying networks [1], where features can be ordered in some meaningful way. Some properties of the fused Lasso have been established in [26].

In this paper, we focus on optimizing the following class of optimization problems with the fused Lasso penalty:

$$\min_{\mathbf{x}\in\mathbb{R}^n} h(\mathbf{x}) = \mathrm{loss}(\mathbf{x}) + \mathrm{fl}(\mathbf{x}), \tag{1}$$

where $\mathrm{loss}(\mathbf{x})$ is a given smooth and convex loss (e.g., the least squares loss) defined on a set of training samples, and

$$\mathrm{fl}(\mathbf{x}) = \lambda_1 \sum_{i=1}^{n} |x_i| + \lambda_2 \sum_{i=2}^{n} |x_i - x_{i-1}| \tag{2}$$

is the fused Lasso penalty with the nonnegative λ_1 and λ_2. The problem in (1) is challenging to solve, as the fused Lasso penalty is both non-smooth and non-separable.

Existing algorithms reformulate (1) as the equivalent constrained smooth optimization problem by introducing additional variables and constraints, and then apply the standard solver for optimization. Let n denote the sample dimensionality. Tibshirani *et al.* proposed the fused Lasso with the least squares loss [30]. They derived a smooth reformulation by introducing $4n$ auxiliary variables, linear constraints of the type: $\mathbf{a} \leq A\mathbf{y} \leq \mathbf{b}$ (the matrix A is of size $(2n+2) \times 5n$ with $11n-1$ non-zero elements), and $4n$ non-negative constraints, and then solved the reformulated problem by the SQOPT[1] package. Ahmed and Xing proposed to solve the fused Lasso penalized logistic regression by introducing $2n$ auxiliary variables and $4n$ inequality constraints [1], and then solved the reformulated problem by the CVX[2] optimization package [10]. However, the constrained smooth reformulation usually does not scale well with n, due to the large number of auxiliary variables and inequality constraints introduced. Indeed, it was pointed out in [30] that, "one difficulty in using the fused Lasso is the computational speed", and "when $n > 2000$ and $m > 200$ (m denotes the number of samples), speed could become a practical limitation".

In this paper, we develop an Efficient Fused Lasso Algorithm (EFLA) by treating the objective function of (1) as a composite function with the smooth part $\mathrm{loss}(\cdot)$ and the other non-smooth part $\mathrm{fl}(\cdot)$. One appealing feature of EFLA is that it makes use of the special structure of (1) for achieving a convergence rate of $O(1/k^2)$ for k iterations, which is optimal for the first-order black-box methods. Note that, when directly applying the black-box first-order method for solving the non-smooth problem (1), one can only achieve a convergence rate of $O(1/\sqrt{k})$, much slower than $O(1/k^2)$.

In the proposed EFLA, a key building block (in each iteration) is the proximal operator [19] associated with the nonsmooth fused Lasso penalty $\mathrm{fl}(\cdot)$ (corresponding to a pair of λ_1 and λ_2), which

[1] tomopt.com/tomlab/products/snopt/solvers/SQOPT.php
[2] stanford.edu/∼boyd/cvx

KDD'10, July 25–28, 2010, Washington, DC, USA.

is also called the Fused Lasso Signal Approximator [8, FLSA]. Existing approaches [8, 12] for solving FLSA usually employ a path technique that requires exactly following the path for computing the solution corresponding to the desired pair of λ_1 and λ_2. Despite their advantage in obtaining the whole path solutions, they might not be efficient for our proposed EFLA, which needs the computation of FLSA corresponding to a pair of λ_1 and λ_2 only in each iteration. In addition, it is hard for them to incorporate the "warm" start technique for further improving the efficiency.

To efficiently solve FLSA corresponding to a pair of λ_1 and λ_2, we propose to reformulate it as the problem of finding an "appropriate" subgradient of the fused penalty at the minimizer, and develop a Subgradient Finding Algorithm (SFA). We further design a restart technique for accelerating the convergence of SFA, by exploiting the special "structures" of the original and the reformulated FLSA problems. When used as a building block in EFLA, SFA is shown to converge within dozens of iterations for problems of size up to 10^7 using the "cold" start; with the "warm" start, SFA usually converges within 10 iterations. Our empirical evaluations show that, both SFA and EFLA significantly outperform the existing solvers. We also demonstrate several applications of the fused Lasso.

Notations: $\|\cdot\|_1$, $\|\cdot\|$, and $\|\cdot\|_\infty$ denote the ℓ_1-, ℓ_2-, and ℓ_∞-norm, respectively. $R \in \mathbb{R}^{(n-1)\times n}$ is a sparse matrix defined as:

$$R_{ij} = \begin{cases} -1 & j = i, i = 1, 2, \ldots, n-1 \\ 1 & j = i+1, i = 1, 2, \ldots, n-1 \\ 0 & \text{otherwise.} \end{cases}$$

We can rewrite the fused Lasso penalty in (2) as

$$\text{fl}(\mathbf{x}) = \lambda_1 \|\mathbf{x}\|_1 + \lambda_2 \|R\mathbf{x}\|_1. \tag{3}$$

Let $\text{sgn}(\cdot)$ and $\text{SGN}(\cdot)$ be the operators defined in the componentwise fashion: if $t > 0$, $\text{sgn}(t) = 1$, $\text{SGN}(t) = \{1\}$; if $t < 0$, $\text{sgn}(t) = -1$, $\text{SGN}(t) = \{-1\}$; and if $t = 0$, $\text{sgn}(t) = 0$, $\text{SGN}(t) = [-1, 1]$. $P_{\lambda_2}(\mathbf{x})$ is an operator that projects each element of $\mathbf{x}$ onto the interval $[-\lambda_2, \lambda_2]$. Let $\mathbf{e} \in \mathbb{R}^n$ be a vector composed of 1's. Denote $[1:n]$ as the set of indices from 1 to n.

2. THE PROPOSED EFFICIENT FUSED LASSO ALGORITHM

We review several categories of first-order methods that can be applied to optimizing the composite function (1) in Section 2.1, present our proposed algorithm in Section 2.2, and discuss the key building block—FLSA in Section 2.3.

2.1 Composite Function Optimization

Subgradient Descent (SD) When treating $h(\mathbf{x})$ as the general non-smooth convex function, we can apply the subgradient descent [20, 21], which can achieve a convergence rate of $O(1/\sqrt{k})$ for k iterations. However, SD has the following two disadvantages: 1) the convergence is slow; and 2) the iterates of SD are very rarely at the points of non-differentiability [6], thus it might not achieve the desirable sparse solution (which is usually at the point of non-differentiability) within a limited number of iterations.

Coordinate Descent (CD) Coordinate descent [33] and its recent extension—coordinate gradient descent [34] are applicable for optimizing the non-differentiable composite function. Convergence results have been established, when the non-differentiable part is separable [33, 34]; and CD was applied for solving Lasso in [8]. However, when applied for solving (1), CD may not converge to the desirable solution, as the fused Lasso penalty is non-separable. Friedman et al. [8] derived a modified CD for solving FLSA—a special case of (1), and discussed its extension for solving the general fused Lasso; however, as explicitly mentioned in [8, Section 3, page 310], the resulting algorithm is not guaranteed to give the exact solution.

Nesterov's Method Nesterov's method [20, 21] is an optimal first-order black-box method for smooth convex optimization, achieving a convergence rate of $O(1/k^2)$. In the recent studies [2, 22], the Nesterov's method is extended to solve the composite function composed of one smooth part and the other non-smooth part. The resulting algorithm can achieve the optimal convergence rate of $O(1/k^2)$, at the expense that the proximal operator [19] associated with the non-smooth part needs to be solved at each iteration. For the problem in (1), the associated proximal operator is the FLSA. The Nesterov's method has been applied to solve various sparse learning formulations [2, 13, 16, 17, 18, 22, 32].

Forward Looking Subgradient (FOLOS) The FOrward-LOoking Subgradient [6] was proposed for optimizing the composite function. FOLOS is a forward-backward splitting method. It can be applied for both online and batch learning. For bach learning, the convergence rates of $O(1/\sqrt{k})$ and $O(1/k)$ were established for the general convex and the smooth convex loss functions.

Regularized Dual Averaging (RDA) The Regularized Dual Averaging [35] was proposed for solving the regularized composite function, based on the dual averaging method proposed in [23]. RDA was designed for stochastic learning and online learning, and the convergence rates of $O(1/\sqrt{k})$ and $O(\ln k/k)$ were established for the general convex and the strongly convex regularization.

In this paper, we consider solving (1) in the batch learning setting, and propose to apply the Nesterov's method due to its fast convergence rate. Note that, one can develop the online learning algorithms for (1) using algorithms such as FOLOS and RDA, where FLSA is also a key building block. The efficient computation of FLSA will be discussed in Section 3.

2.2 The Efficient Fused Lasso Algorithm

We first construct the following model for approximating the composite function $h(\cdot)$ at the point $\mathbf{x}$:

$$h_{L,\mathbf{x}}(\mathbf{y}) = [\text{loss}(\mathbf{x}) + \langle \text{loss}'(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle] + \text{fl}(\mathbf{y}) + \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2, \tag{4}$$

where $L > 0$. In the model $h_{L,\mathbf{x}}(\mathbf{y})$, we apply the first-order Taylor expansion at the point $\mathbf{x}$ (including all terms in the square bracket) for the smooth function $\text{loss}(\cdot)$, and directly put the non-smooth penalty $\text{fl}(\cdot)$ into the model. The regularization term $\frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2$ prevents $\mathbf{y}$ from walking far away from $\mathbf{x}$, thus the model can be a good approximation to $h(\mathbf{y})$ in the neighborhood of $\mathbf{x}$.

With the model (4), we can develop the following gradient descent like method for solving (1):

$$\mathbf{x}_{i+1} = \arg\min_{\mathbf{y}} h_{L_i,\mathbf{x}_i}(\mathbf{y}) \tag{5}$$

for some properly chosen $\{L_i\}$. It has been shown in [2, 21] that, the scheme in (5) can yield a convergence rate of $O(1/k)$, and can be further accelerated to $O(1/k^2)$.

The accelerated method can be derived using the "estimate sequence" [21, 22], which is quite involved. To make the presentation relatively easy to follow, we use the scheme provided in [2, 20] to present the Nesterov's method for solving (1).

The Nesterov's method is based on two sequences $\{\mathbf{x}_i\}$ and $\{\mathbf{s}_i\}$ in which $\{\mathbf{x}_i\}$ is the sequence of approximate solutions, and $\{\mathbf{s}_i\}$ is the sequence of search points. The search point $\mathbf{s}_i$ is the affine combination of $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$ as

$$\mathbf{s}_i = \mathbf{x}_i + \beta_i(\mathbf{x}_i - \mathbf{x}_{i-1}), \tag{6}$$

where β_i is a properly chosen coefficient. The approximate solution $\mathbf{x}_{i+1}$ is computed as the minimizer of $h_{L_i,\mathbf{s}_i}(\mathbf{y})$:

$$\mathbf{x}_{i+1} = \arg\min_{\mathbf{y}} h_{L_i,\mathbf{s}_i}(\mathbf{y}), \tag{7}$$

where L_i is determined by the line search according to the Armijo-Goldstein rule so that L_i should be appropriate for $\mathbf{s}_i$.

The algorithm for solving (1) is presented in Algorithm 1. Following the proof given in [20, 2], we can establish the following global convergence result:

$$h(\mathbf{x}_{k+1}) - h(\mathbf{x}^*) \leq \frac{2\max(2\tilde{L}, L_0)\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{(k+1)^2}, \tag{8}$$

where $\mathbf{x}^*$ is an optimal solution to (1), and $\tilde{L}$ is the Lipschitz continuous gradient of the smooth convex loss function $\mathrm{loss}(\cdot)$.

Algorithm 1 The Efficient Fused Lasso Algorithm (EFLA)

Input: $\lambda_1 \geq 0, \lambda_2 \geq 0, L_0 > 0, \mathbf{x}_0, k$
Output: $\mathbf{x}_{k+1}$

1: Initialize $\mathbf{x}_1 = \mathbf{x}_0$, $\alpha_{-1} = 0$, $\alpha_0 = 1$, and $L = L_0$.
2: **for** $i = 1$ to k **do**
3: Set $\beta_i = \frac{\alpha_{i-2}-1}{\alpha_{i-1}}$, $\mathbf{s}_i = \mathbf{x}_i + \beta_i(\mathbf{x}_i - \mathbf{x}_{i-1})$
4: Find the smallest $L = L_{i-1}, 2L_{i-1}, \ldots$ such that

$$h(\mathbf{x}_{i+1}) \leq h_{L,\mathbf{s}_i}(\mathbf{x}_{i+1}),$$

where $\mathbf{x}_{i+1} = \arg\min_{\mathbf{y}} h_{L,\mathbf{s}_i}(\mathbf{y})$
5: Set $L_i = L$ and $\alpha_{i+1} = \frac{1+\sqrt{1+4\alpha_i^2}}{2}$
6: **end for**

In Algorithm 1, a key building block is the problem (7), which is the fused Lasso signal approximator to be discussed in the next subsection.

2.3 Fused Lasso Signal Approximator

The Fused Lasso Signal Approximator (FLSA) solves the following problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f_{\lambda_2}^{\lambda_1}(\mathbf{x}) \equiv \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|^2 + \lambda_1\|\mathbf{x}\|_1 + \lambda_2\|R\mathbf{x}\|_1, \tag{9}$$

which is a special case of (1) by setting $\mathrm{loss}(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|^2$. Note that, FLSA in (9) is essentially the proximal operator [11, 14, 19] associated with the fused Lasso penalty $\mathrm{fl}(\mathbf{x})$.

Let

$$\pi_{\lambda_2}^{\lambda_1}(\mathbf{v}) = \arg\min_{\mathbf{x}} f_{\lambda_2}^{\lambda_1}(\mathbf{x}). \tag{10}$$

We can easily verify that

$$\arg\min_{\mathbf{y}} h_{L_i,\mathbf{s}_i}(\mathbf{y}) = \pi_{\lambda_2/L_i}^{\lambda_1/L_i}(\mathbf{s}_i - \mathrm{loss}'(\mathbf{s}_i)/L_i). \tag{11}$$

Thus the building block (7) in Algorithm 1 can be solved by FLSA in (9). In the sequel, we present its efficient computation.

The objective function $f_{\lambda_2}^{\lambda_1}(\cdot)$ is strictly convex, thus it admits a unique minimizer, denoted by $\mathbf{x}^*$. The optimality condition [21, Theorem 3.15, Chapter 3] requires that

$$\mathbf{0} \in \partial f_{\lambda_2}^{\lambda_1}(\mathbf{x}^*), \tag{12}$$

where $\partial f_{\lambda_2}^{\lambda_1}(\mathbf{x}^*)$ denotes the subdifferential of $f_{\lambda_2}^{\lambda_1}(\cdot)$ at $\mathbf{x}^*$. The subdifferential of $f_{\lambda_2}^{\lambda_1}(\cdot)$ can be computed as:

$$\partial f_{\lambda_2}^{\lambda_1}(\mathbf{x}) = \mathbf{x} - \mathbf{v} + \lambda_1\mathrm{SGN}(\mathbf{x}) + \lambda_2 R^{\mathrm{T}}\mathrm{SGN}(R\mathbf{x}). \tag{13}$$

Using the subgradient technique, it has been shown in [8] that, the minimizer of the problem (9) for any value of (λ_1, λ_2) can be obtained by a simple soft-thresholding of the solution obtained for $(0, \lambda_2)$, as stated in Theorem 1. We provide an alternative and simplified proof using the technique of subdifferential; and this simplified proof also motivates our proposed method in Section 3.

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

$$\pi_{\lambda_2}^{\lambda_1}(\mathbf{v}) = \mathrm{sgn}(\pi_{\lambda_2}^0(\mathbf{v})) \odot \max(|\pi_{\lambda_2}^0(\mathbf{v})| - \lambda_1, 0). \tag{14}$$

Proof: We first analyze the optimality condition for the solution $\pi_{\lambda_2}^0(\mathbf{v})$. According to (12) and (13), there exists

$$\mathbf{z}^* \in \lambda_2\mathrm{SGN}(R\pi_{\lambda_2}^0(\mathbf{v})) \tag{15}$$

such that $\pi_{\lambda_2}^0(\mathbf{v}) = \mathbf{v} - R^{\mathrm{T}}\mathbf{z}^*$. Let

$$\mathbf{x} = \mathrm{sgn}(\pi_{\lambda_2}^0(\mathbf{v})) \odot \max(|\pi_{\lambda_2}^0(\mathbf{v})| - \lambda_1, 0),$$

$$\mathbf{g} = \mathrm{sgn}(\pi_{\lambda_2}^0(\mathbf{v})) \odot \min(|\pi_{\lambda_2}^0(\mathbf{v})|, \lambda_1).$$

We can easily verify $\mathbf{x} - \mathbf{v} + \mathbf{g} + R^{\mathrm{T}}\mathbf{z}^* = \mathbf{0}$ and $\mathbf{g} \in \lambda_1\mathrm{SGN}(\mathbf{x})$. Utilizing the definition of $\mathbf{x}$, the special structure of R that each of its row has two nonzero elements -1 and 1, and (15), we can verify $\mathbf{z}^* \in \lambda_2\mathrm{SGN}(R\mathbf{x})$. Therefore,

$$\mathbf{0} = \mathbf{x} - \mathbf{v} + \mathbf{g} + R^{\mathrm{T}}\mathbf{z}^* \in \partial f_{\lambda_2}^{\lambda_1}(\mathbf{x}).$$

It follows from the optimality condition (12) that (14) holds. ■

Theorem 1 implies that, it suffices to solve (9) with $\lambda_1 = 0$. For discussion convenience, we omit the superscript to indicate that $\lambda_1 = 0$. The proof of Theorem 1 implies that, $\pi_{\lambda_2}(\mathbf{v})$ can be analytically solved as $\pi_{\lambda_2}(\mathbf{v}) = \mathbf{v} - R^{\mathrm{T}}\mathbf{z}^*$, provided that an appropriate $\mathbf{z}^* \in \lambda_2\mathrm{SGN}(R\pi_{\lambda_2}(\mathbf{v}))$ can be found. Interestingly, $\mathbf{z}^*$ is unique, as shown in the following analysis. It follows from $\pi_{\lambda_2}(\mathbf{v}) = \mathbf{v} - R^{\mathrm{T}}\mathbf{z}^*$ that $\mathbf{z}^*$ is a solution to the linear system $RR^{\mathrm{T}}\mathbf{z} = R\mathbf{v} - R\pi_{\lambda_2}(\mathbf{v})$. Since RR^{T} is positive definite (see the discussion in Section 3.1) and $\pi_{\lambda_2}(\mathbf{v})$ is unique, we conclude that $\mathbf{z}^*$ is unique. The above discussion motivates us to solve (9) via finding the *appropriate* and *unique* $\mathbf{z}^*$.

In the next section, we shall show that $\mathbf{z}^*$ can be efficiently computed by a special quadratic programming problem with the bound constraint. As $R^{\mathrm{T}}\mathbf{z}^*$ is a subgradient of the fused penalty $\lambda_2\|R\mathbf{x}\|_1$ at the minimizer, we term our proposed method as the Subgradient Finding Algorithm (SFA). Note that, SFA is our main technical contribution in this paper.

3. THE SUBGRADIENT FINDING ALGORITHM

In this section, we discuss solving (9) with $\lambda_1 = 0$, i.e.,

$$\min_{\mathbf{x}\in\mathbb{R}^n} f_{\lambda_2}(\mathbf{x}) \equiv \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|^2 + \lambda_2\|R\mathbf{x}\|_1. \tag{16}$$

Introducing the dual variable $\mathbf{z} \in \mathbb{R}^{n-1}$, we can reformulate (16) as the following equivalent min-max problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \max_{\|\mathbf{z}\|_\infty \leq \lambda_2} \phi(\mathbf{x}, \mathbf{z}) \equiv \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|^2 + \langle R\mathbf{x}, \mathbf{z}\rangle. \tag{17}$$

This is a saddle-point problem, and the existence of the saddle point is ensured by the well-known Von Neumann Lemma [20], as $\phi(\mathbf{x}, \mathbf{z})$ is differentiable, convex in $\mathbf{x}$, and concave in $\mathbf{z}$.

Exchanging min and max and setting the derivative of $\phi(\mathbf{x}, \mathbf{z})$ with regard to $\mathbf{x}$ to zero, we have

$$\mathbf{x} = \mathbf{v} - R^{\mathrm{T}}\mathbf{z}. \tag{18}$$

Plugging (18) into (17), we obtain the following optimization problem with regard to $\mathbf{z}$:

$$\min_{\|\mathbf{z}\|_\infty \le \lambda_2} \psi(\mathbf{z}) \equiv -\phi(\mathbf{v} - R^{\mathrm{T}}\mathbf{z}, \mathbf{z}) = \frac{1}{2}\|R^{\mathrm{T}}\mathbf{z}\|^2 - \langle R^{\mathrm{T}}\mathbf{z}, \mathbf{v}\rangle. \tag{19}$$

It follows from (18) that, once $\mathbf{z}^*$, the minimizer of (19), is found, we can analytically obtain $\pi_{\lambda_2}(\mathbf{v})$; and this coincides with the relationship $\pi_{\lambda_2}(\mathbf{v}) = \mathbf{v} - R^{\mathrm{T}}\mathbf{z}^*$ shown in the proof of Theorem 1.

In the sequel, we discuss the efficient optimization of the bound constrained quadratic programming problem (19). To this end, we exploit the special "structures" of (16), (17) and (19), and develop a novel restart technique for the fast convergence.

The rest of this section is organized as follows: we present the optimality condition for (19) in Section 3.1, derive the maximal value of λ_2 in Section 3.2, present the proposed SFA in Section 3.3, compute the duality gap of the solution in Section 3.4, develop a restart technique for accelerating the convergence in Section 3.5, and provide further discussions in Section 3.6.

3.1 The Optimality Condition

The Hessian of $\psi(\cdot)$ can be computed as

$$RR^{\mathrm{T}} = \begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ 0 & & & -1 & 2 \end{pmatrix}, \tag{20}$$

which is an $(n-1) \times (n-1)$ tridiagonal matrix. The $n-1$ eigenvalues of the Hessian RR^{T} can be analytically computed as $2 - 2\cos(i\pi/n), i = 1, 2, \ldots, n-1$. Therefore, the Hessian is positive definite, and the minimizer of (19) is *unique*.

According to [21, Theorem 2.2.5, Chapter 2], we have that $\mathbf{z}^*$, satisfying $\|\mathbf{z}^*\|_\infty \le \lambda_2$, is a minimizer of (19) if and only if

$$\langle \mathbf{z} - \mathbf{z}^*, \psi'(\mathbf{z}^*)\rangle \ge 0, \forall \mathbf{z} : \|\mathbf{z}\|_\infty \le \lambda_2. \tag{21}$$

The optimality condition leads to an important relationship between the minimizer and its gradient, as summarized in the following lemma (this lemma shall help derive the restart technique to be discussed in Section 3.5):

LEMMA 1. *Let $\mathbf{g}^* = \psi'(\mathbf{z}^*)$. We have: 1) if $g_i^* > 0$, then $z_i^* = -\lambda_2$; 2) if $g_i^* < 0$, then $z_i^* = \lambda_2$; and 3) if $|z_i^*| < \lambda_2$, then $g_i^* = 0$.*

3.2 Computing the Maximal Value for λ_2

When $\lambda_2 \to \infty$, the problem (19) becomes an unconstrained optimization problem. Intuitively, there exists a $\lambda_2^{\max}$ (the maximal value for λ_2), over which the problem (19) has the same solution. The following theorem shows how to compute $\lambda_2^{\max}$.

THEOREM 2. *The linear system*

$$RR^{\mathrm{T}}\mathbf{z} = R\mathbf{v} \tag{22}$$

has a unique solution, denoted by $\hat{\mathbf{z}}$. Let

$$\lambda_2^{\max} = \|\hat{\mathbf{z}}\|_\infty. \tag{23}$$

For any $\lambda_2 \ge \lambda_2^{\max}$, the solution of (19) is $\hat{\mathbf{z}}$, and the solution of (16) can be analytically computed as:

$$\pi_{\lambda_2}(\mathbf{v}) = \langle \mathbf{e}, \mathbf{v}\rangle \mathbf{e}/n. \tag{24}$$

Proof: As RR^{T}, the Hessian of $\psi(\cdot)$, is positive definite, the linear system (22) has a unique solution, denoted by $\hat{\mathbf{z}}$. For any $\lambda_2 \ge \lambda_2^{\max}$, we can easily verify that $\|\hat{\mathbf{z}}\|_\infty = \lambda_2^{\max} \le \lambda_2$ and $\psi'(\hat{\mathbf{z}}) = RR^{\mathrm{T}}\hat{\mathbf{z}} - R\mathbf{v} = \mathbf{0}$. It follows from the optimality condition (21) that $\hat{\mathbf{z}}$ is the optimal solution of (19) for any $\lambda_2 \ge \lambda_2^{\max}$.

When $\lambda_2 \ge \lambda_2^{\max}$, we have $\pi_{\lambda_2}(\mathbf{v}) = \mathbf{v} - R^{\mathrm{T}}\hat{\mathbf{z}}$ from (18). It follows that 1) $R\pi_{\lambda_2}(\mathbf{v}) = R(\mathbf{v} - R^{\mathrm{T}}\hat{\mathbf{z}}) = -\psi'(\hat{\mathbf{z}}) = \mathbf{0}$, and 2) $\mathbf{e}^{\mathrm{T}}\pi_{\lambda_2}(\mathbf{v}) = \mathbf{e}^{\mathrm{T}}\mathbf{v} - \mathbf{e}^{\mathrm{T}}R^{\mathrm{T}}\hat{\mathbf{z}} = \mathbf{e}^{\mathrm{T}}\mathbf{v}$, where the last equality holds as $R\mathbf{e} = \mathbf{0}$. Thus (24) holds for any $\lambda_2 \ge \lambda_2^{\max}$. ■

The linear system (22) can be efficiently computed in $O(n)$ time by using the special tridiagonal structure of the matrix RR^{T}. One well known algorithm for solving the general tridiagonal systems of equations is the Thomas algorithm [29], which consumes approximately $3n$ additions and $5n$ multiplications. In addition, when considering the special form of RR^{T} in (20), we can apply the Rose algorithm [27], which costs only n multiplications and $3n-5$ additions, as can be easily observed from Algorithm 2. Evans [7] presented an algorithm similar to the Rose algorithm, and proved that its rounding error is bounded.

Algorithm 2 The Rose Algorithm

Input: $\mathbf{u} \in \mathbb{R}^{(n-1)\times 1}$
Output: $\hat{\mathbf{z}} \in \mathbb{R}^{(n-1)\times 1}$ satisfying $RR^{\mathrm{T}}\hat{\mathbf{z}} = \mathbf{u}$
1: Compute the scalar $s = -n^{-1}\sum_{j=1}^{n-1} j \times u_j$
2: Compute $\hat{z}_j$ sequentially using $\hat{z}_{n-1} = u_{n-1} + s$ and $\hat{z}_j = \hat{z}_{j+1} + u_j, j = n-2, \ldots, 1$
3: Obtain $\hat{z}_j$ sequentially using $\hat{z}_j = \hat{z}_j + \hat{z}_{j-1}, j = 2, \ldots, n-1$

To solve (19), we can first compute $\hat{\mathbf{z}}$, the solution to the linear system (22) by Algorithm 2, and obtain $\lambda_2^{\max}$. If $\lambda_2 \ge \lambda_2^{\max}$, $\hat{\mathbf{z}}$ is the solution of (19); otherwise, we apply the algorithm to be discussed in the subsequent subsections for $0 \le \lambda_2 < \lambda_2^{\max}$. We note that, Algorithm 2 shall also be used in the restart technique to be discussed in Section 3.5.

3.3 SFA Via Gradient Descent

In the literature, there have been quite a few algorithms for solving the bound constrained optimization problem like (19); e.g., [4, 5, 15] and the references therein. However, these algorithms are either for the general quadratic programming or the general optimization.

In this paper, we propose to apply the gradient descent [21], and present the algorithm in Algorithm 3. According to [21, Chapter 2.2.4], Algorithm 3 converges linearly as

$$\|\mathbf{z}_k - \mathbf{z}^*\|^2 \le (1 - \frac{\mu}{L})^k \|\mathbf{z}_0 - \mathbf{z}^*\|^2, \tag{25}$$

where $L = 2 - 2\cos(\pi(n-1)/n)$ and $\mu = 2 - 2\cos(\pi/n)$ are the largest and the smallest eigenvalues of the Hessian RR^{T}, respectively. Algorithm 3 can be further accelerated with the Nesterov's method; and we denote the resulting algorithm as $\mathrm{SFA_N}$.

Our proposed $\mathrm{SFA_G}$ can be significantly accelerated with the restart technique (to be discussed in Section 3.5), by exploiting the special "structures" of the original and the reformulated problems. Before presenting the restart technique, we show in the next subsection how to compute the duality gap for checking the convergence of the algorithm.

Algorithm 3 SFA via Gradient Descent (SFA_G)

Input: $\mathbf{v} \in \mathbb{R}^{n\times 1}$, $0 \leq \lambda_2 < \lambda_2^{\max}$, $\mathbf{z}_0 \in \mathbb{R}^{(n-1)\times 1}$, k
Output: $\mathbf{z}_k \in \mathbb{R}^{(n-1)\times 1}$

1: Set $L = 2 - 2\cos(\pi(n-1)/n)$
2: **for** $i = 1$ to k **do**
3: Compute $\mathbf{g}_i = \psi'(\mathbf{z}_i) = RR^{\mathrm{T}}\mathbf{z}_i - R\mathbf{v}$
4: Set $\mathbf{z}_{i+1} = P_{\lambda_2}(\mathbf{z}_i - \mathbf{g}_i/L)$
5: **end for**

3.4 Computing the Duality Gap

In optimizing (19) via SFA_G (see Algorithm 3), SFA_N (the accelerated version of SFA_G via the Nesterov's method), and SFA_R (SFA with the restart technique to be discussed in the next subsection), it would be desirable that we can check the convergence of the algorithm based on the "goodness" of the approximate solution. To this end, we propose to compute the duality gap for the min-max optimization problem (17), as both (16) and (19) are its resulting problems by eliminating the variable $\mathbf{z}$ or $\mathbf{x}$.

Let $\tilde{\mathbf{z}}$ be an appropriate solution computed by SFA_G (or SFA_N and SFA_R). Note that, we have $\|\tilde{\mathbf{z}}\|_\infty \leq \lambda_2$ from Step 4 of Algorithm 3. Let $\tilde{\mathbf{x}} = \mathbf{v} - R^{\mathrm{T}}\tilde{\mathbf{z}}$ be the appropriate solution computed by (18). We can define the duality gap for (17) at $(\tilde{\mathbf{x}}, \tilde{\mathbf{z}})$ as:

$$\text{gap}(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}) = \max_{\mathbf{z}:\|\mathbf{z}\|_\infty \leq \lambda_2} \phi(\tilde{\mathbf{x}}, \mathbf{z}) - \min_{\mathbf{x}} \phi(\mathbf{x}, \tilde{\mathbf{z}}). \quad (26)$$

The following theorem shows that, $\text{gap}(\tilde{\mathbf{x}}, \tilde{\mathbf{z}})$ can be computed using $\tilde{\mathbf{z}}$ only, and it measures the "goodness" of the solutions $\tilde{\mathbf{z}}$ and $\tilde{\mathbf{x}}$ for $\psi(\cdot)$ and $f_{\lambda_2}(\cdot)$, respectively.

THEOREM 3. *The duality gap in (26) can be computed as:*

$$\text{gap}(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}) = \lambda_2\|\psi'(\tilde{\mathbf{z}})\|_1 + \langle \tilde{\mathbf{z}}, \psi'(\tilde{\mathbf{z}})\rangle. \quad (27)$$

In addition, we have

$$\psi(\tilde{\mathbf{z}}) - \psi(\mathbf{z}^*) \leq \text{gap}(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}), \quad (28)$$

$$f_{\lambda_2}(\tilde{\mathbf{x}}) - f_{\lambda_2}(\mathbf{x}^*) \leq \text{gap}(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}). \quad (29)$$

Proof: From (16-19), we can establish the following relationships:

$$-\psi(\tilde{\mathbf{z}}) = \phi(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}) = \min_{\mathbf{x}} \phi(\mathbf{x}, \tilde{\mathbf{z}}) \leq \phi(\mathbf{x}^*, \tilde{\mathbf{z}}), \quad (30)$$

$$\phi(\mathbf{x}^*, \tilde{\mathbf{z}}) \leq \max_{\mathbf{z}:\|\mathbf{z}\|_\infty \leq \lambda_2} \phi(\mathbf{x}^*, \mathbf{z}) = \phi(\mathbf{x}^*, \mathbf{z}^*) = -\psi(\mathbf{z}^*), \quad (31)$$

$$f_{\lambda_2}(\mathbf{x}^*) = \phi(\mathbf{x}^*, \mathbf{z}^*) = \min_{\mathbf{x}} \phi(\mathbf{x}, \mathbf{z}^*) \leq \phi(\tilde{\mathbf{x}}, \mathbf{z}^*), \quad (32)$$

$$\phi(\tilde{\mathbf{x}}, \mathbf{z}^*) \leq \max_{\mathbf{z}:\|\mathbf{z}\|_\infty \leq \lambda_2} \phi(\tilde{\mathbf{x}}, \mathbf{z}) = f_{\lambda_2}(\tilde{\mathbf{x}}). \quad (33)$$

From (26-33), we can write the duality gap as:

$$\begin{aligned} \text{gap}(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}) &= f_{\lambda_2}(\tilde{\mathbf{x}}) - \phi(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}) = \lambda_2\|R\tilde{\mathbf{x}}\|_1 - \langle \tilde{\mathbf{z}}, R\tilde{\mathbf{x}}\rangle \\ &= \lambda_2\|\psi'(\tilde{\mathbf{z}})\|_1 + \langle \tilde{\mathbf{z}}, \psi'(\tilde{\mathbf{z}})\rangle, \end{aligned} \quad (34)$$

where the last equality follows from $\psi'(\tilde{\mathbf{z}}) = RR^{\mathrm{T}}\tilde{\mathbf{z}} - R\mathbf{v}$ and $\tilde{\mathbf{x}} = \mathbf{v} - R\tilde{\mathbf{z}}$. The inequalities (28) and (29) can be easily verified using (26-33). This completes the proof. ■

3.5 SFA via the Restart Technique

Although Algorithm 3 has a linear convergence rate, it may converge slowly for large n, due to the high condition number $\frac{L}{\mu}$. For example, when $n = 10^2, 10^3, 10^4$ and 10^5, we have $\frac{L}{\mu} \approx 4\times 10^3, 4\times 10^5, 4\times 10^7$ and 4×10^9, respectively. In this subsection, we propose to make use of the special structures of (16), (17), and (19) to restart SFA_G for fast convergence; and we call the resulting method as SFA_R (SFA via the restart technique). Figure 1 illustrates SFA_G, SFA_N and SFA_R for solving (19). From this figure, we can clearly observe that, the restart technique can significantly accelerate the convergence and yield the exact solution using much fewer iterations than SFA_G and SFA_N.

Our proposed restart technique is based on the so-called *support set*[3], motivated by Lemma 1. Specifically, for any $\|\mathbf{z}\|_\infty \leq \lambda_2$, we define its *support set* as:

$$S(\mathbf{z}) = \{i \in [1:n-1] : |z_i| = \lambda_2, z_i g_i < 0, \mathbf{g} = \psi'(\mathbf{z})\}. \quad (35)$$

When S is nonempty, we denote the j-th largest element in the set S by $s_j, j = 1, 2, \ldots, |S|$. It is clear that $1 \leq s_1$ and $s_{|S|} \leq n-1$. For discussion convenience, we let $s_0 = 0$ and $s_{|S|+1} = n$. We note that the following discussion also holds for the case when S is empty. With $s_0, s_1, \ldots, s_{|S|+1}$, we can partition the indices in $[1:n]$ into $|S|+1$ non-overlapping groups:

$$G_j = \{i : s_{j-1} + 1 \leq i \leq s_j\}, 1 \leq j \leq |S| + 1. \quad (36)$$

Let $\mathbf{e}_{G_j}$ and $\mathbf{v}_{G_j}$ denote the j-th group of $\mathbf{e}$ and $\mathbf{v}$ corresponding to the indices in G_j, respectively.

Based on the *support set* S, we define the mapping $\mathbf{x} = \omega(\mathbf{z})$ as:

$$x_i = \frac{\langle \mathbf{e}_{G_j}, \mathbf{v}_{G_j}\rangle - z_{s_{j-1}} + z_{s_j}}{|G_j|}, i \in G_j, \quad (37)$$

where $j = 1, 2, \ldots, |S|+1$ and we have assumed $z_0 = z_n = 0$ for presentation convenience.

LEMMA 2. *For any $\|\mathbf{z}\|_\infty \leq \lambda_2$ and $i \in S(\mathbf{z})$, we have: 1) if $z_i = \lambda_2$, then $v_{i+1} > v_i$; and 2) if $z_i = -\lambda_2$, then $v_{i+1} < v_i$.*

Proof Let $\mathbf{g} = \psi'(\mathbf{z})$. For discussion convenience, we add $z_0 = z_n = 0$ into the $(n-1)$-dimensional vector $\mathbf{z}$. We have $g_i = -z_{i-1} + 2z_i - z_{i+1} - (v_{i+1} - v_i)$. If follows from the definition of $S(\mathbf{z})$ in (35) that, 1) if $z_i = \lambda_2$, then $g_i = -z_{i-1} + 2\lambda_2 - z_{i+1} - (v_{i+1} - v_i) < 0$, which leads to $v_{i+1} > v_i$; and 2) if $z_i = -\lambda_2$, then $g_i = z_{i-1} - 2\lambda_2 - z_{i+1} - (v_{i+1} - v_i) > 0$, which leads to $v_{i+1} < v_i$. ■

Lemma 2 shows that, for any $i \in S(\mathbf{z})$, the sign of z_i is indeed determined by $v_{i+1} - v_i$. Next, we show that the optimal solution to (16) can be exactly recovered using the *support set* $S(\mathbf{z}^*)$ only.

THEOREM 4. *Let $\mathbf{z}^*$ be the minimizer of (19). Then $\mathbf{x}^*$, the minimizer of (16), satisfies*

$$\mathbf{x}^* = \omega(\mathbf{z}^*). \quad (38)$$

Proof Let $\mathbf{g}^* = \psi'(\mathbf{z}^*)$. It follows from Lemma 1 and the definition of S in (35) that, $g_i^* = 0, \forall i \notin S$. Based on the relationship $R\mathbf{x}^* = -\psi'(\mathbf{z}^*) = -\mathbf{g}^*$, we have $x_i^* = x_{i'}^*, \forall i, i' \in G_j$. Let the matrix R be partitioned into $|S|+1$ non-overlapping blocks as $R = [R_{G_1}, R_{G_2}, \ldots, R_{G_{|S|+1}}]$. From $\mathbf{x}^* = \mathbf{v} - R^{\mathrm{T}}\mathbf{z}^*$, we have

$$\mathbf{x}_{G_j}^* = \mathbf{v}_{G_j} - R_{G_j}^{\mathrm{T}}\mathbf{z}^*, \forall j. \quad (39)$$

We can easily get (38) by left multiplying $\mathbf{e}_{G_j}^{\mathrm{T}}$ to both sides of (39), using (37), and incorporating the fact that $x_i^* = x_{i'}^*$ for all $i, i' \in G_j$. This completes the proof. ■

[3]We call $S(\mathbf{z})$ the *support set* due to the following two reasons: 1) as shall be shown in Theorem 4, $S(\mathbf{z}^*)$ supports the exact recovery of $\mathbf{x}^*$ for (16), and 2) $S(\mathbf{z})$ directly induces the mapping $\omega : \mathbf{z} \to \mathbf{x}$ in (37).

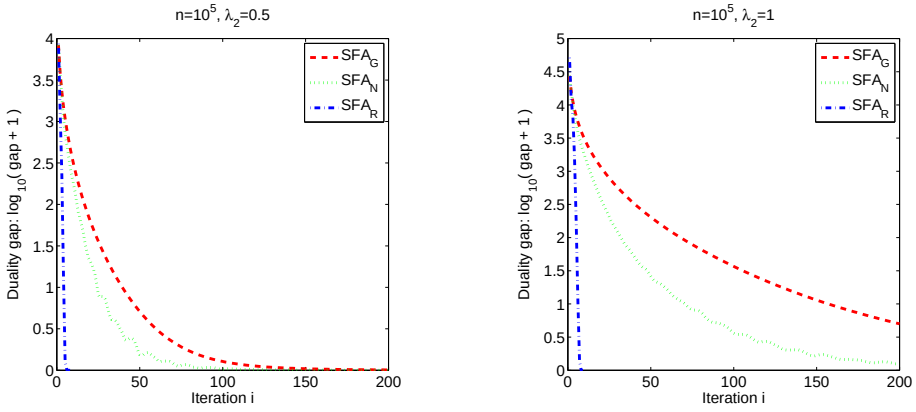

Figure 1: Illustration of SFA_G, SFA_N, and SFA_R for solving (19) (see the experimental setting in Section 4.1). SFA_G, SFA_N and SFA_R denote the proposed SFA via the gradient descent, the Nesterov's method and the restart technique, respectively. The duality gaps reported in this figure are in the logarithmic scale. In 200 iterations, SFA_G (SFA_N) achieves the duality gaps 0.01 (0.001), 4.02 (0.22) for $\lambda_2 = 0.5$ and 1, respectively. SFA_R achieves the exact solution in 8 and 9 iterations for $\lambda_2 = 0.5$ and 1, respectively.

Theorem 4 offers an alternative way for computing $\mathbf{x}^*$ from $\mathbf{z}^*$, which is quite different from (18). More specifically, (18) requires that all the entries in $\mathbf{z}^*$ are known; while (38) says that, $\mathbf{x}^*$ can be *exactly* computed, if the *support set* $S(\mathbf{z}^*)$ is known (we have used Lemma 2). In other words, if we have an appropriate solution $\tilde{\mathbf{z}} \neq \mathbf{z}^*$, we can still exactly obtain $\mathbf{x}^* = \omega(\tilde{\mathbf{z}})$, provided that $S(\tilde{\mathbf{z}}) = S(\mathbf{z}^*)$. Intuitively, this shows that, for a given appropriate solution $\tilde{\mathbf{z}} \neq \mathbf{z}^*$, $\mathbf{x} = \omega(\tilde{\mathbf{z}})$ can be a much better approximate solution than $\tilde{\mathbf{x}} = \mathbf{v} - R^{\mathrm{T}}\tilde{\mathbf{z}}$ for optimizing $f_{\lambda_2}(\cdot)$, provided that $S(\tilde{\mathbf{z}})$ is close to $S(\mathbf{z}^*)$. In our proposed restart technique, with $\mathbf{x} = \omega(\tilde{\mathbf{z}})$, we compute a restart point $\mathbf{z}_0$ using the relationship $\mathbf{x} = \mathbf{v} - R^{\mathrm{T}}\mathbf{z}_0$. Here, $\mathbf{z}_0$ can be easily computed by solving the linear system $RR^{\mathrm{T}}\mathbf{z}_0 = R\mathbf{v} - R\mathbf{x}$. We present the proposed restart technique in Algorithm 4, where Step 4 ensures that $\mathbf{z}_0$ is feasible for (19).

Algorithm 4 The Restart Technique

Input: $R\mathbf{v} \in \mathbb{R}^{(n-1)\times 1}$, $0 \leq \lambda_2 < \lambda_2^{\max}$, $\tilde{\mathbf{z}} \in \mathbb{R}^{(n-1)\times 1}$
Output: Restart point $\mathbf{z}_0 \in \mathbb{R}^{(n-1)\times 1}$

1: Compute the *support set* $S(\tilde{\mathbf{z}})$ according to (35)
2: Compute $\mathbf{x} = \omega(\tilde{\mathbf{z}})$ according to (37)
3: Calculate $\mathbf{z}_0$ as the solution to $RR^{\mathrm{T}}\mathbf{z}_0 = R\mathbf{v} - R\mathbf{x}$
4: Set $\mathbf{z}_0 = P_{\lambda_2}(\mathbf{z}_0)$

We illustrate the proposed Subgradient Finding Algorithm via the restart technique (SFA_R) in Figure 2, from which we can see that, SFA_R recursively calls SFA_G (Algorithm 3) and the restart technique (Algorithm 4). For the SFA_G block in the proposed SFA_R, we set the number of iteration(s) $k = 1$, as it yields the best performance in our experiments. It is clear that the per iteration cost of SFA_R is $O(n)$.

3.6 Discussion

We summarize our methodology for solving (16) as follows. We first make use of the dual of the ℓ_1-norm to rewrite the primal problem (16) as the equivalent saddle point problem (17). By using the relationship between the primal and dual variables in (18), we obtain the dual problem (19), which is a bound constrained quadratic programming problem and can be solved in linear time by the first-order methods such as gradient descent. To further accelerate the optimization of (19), we propose a restart technique. The underlying motivation is that, at a given appropriate solution $\tilde{\mathbf{z}}$, if we can obtain a better appropriate point $\mathbf{z}_0$, the optimization can be greatly accelerated with the restart of $\mathbf{z}_0$. For the problem discussed in this paper, $\mathbf{z}_0$ is obtained by exploiting the "structures" of the primal problem (16), the saddle point problem (17), and the dual problem (19). Such a restart technique can potentially be used for other problems, when utilizing the problem "structures" in a nice way.

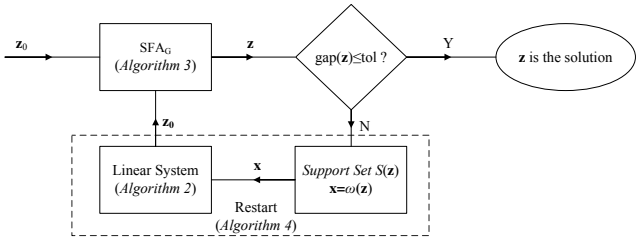

Figure 2: The flow chart of the proposed SFA_R (Subgradient Finding Algorithm via the restart technique). SFA_R recursively calls SFA_G (Algorithm 3) and the restart technique (Algorithm 4) until the duality gap is within a pre-specified precision parameter tol (set to 10^{-12} in our experiments).

Next, we compare our proposed SFA with the modified CD (mCD) proposed in [8] and the path algorithm (pathFLSA) proposed in [12]. First, both mCD and pathFLSA focus on solving the original formulation (16); while our proposed SFA focuses on solving the dual problem (19) utilizing the special structures of (16), (17) and (19). Second, in solving $\pi_{\lambda_2}(\mathbf{v})$, both mCD and pathFLSA need to start from $\lambda = 0$ and then increase λ according to certain strategies until $\lambda = \lambda_2$ to obtain the path solutions; while our proposed SFA directly solves (19) corresponding to the given λ_2. We note that, for EFLA in Algorithm 1, we need the efficient computation of $\pi_{\lambda_2}(\mathbf{v})$ corresponding to a given λ_2 rather than the path solutions; and this is also the case for the the online learning algorithms (e.g., FOLOS [6] and RDA [35]) for solving (1). Third, the starting point of our proposed SFA is quite flexible, thus it can benefit from the "warm" start technique, when used as a building block in EFLA (note that, the solution of FLSA in the previous EFLA iteration can potentially be close to that of FLSA in the next iteration; and Figure 4 illustrates the benefit of the"warm" start); while in solving

$\pi_{\lambda_2}(\mathbf{v})$, both mCD and pathFLSA need to exactly follow the solution path thus they cannot benefit from the "warm" start technique (note that the solution of FLSA in the previous EFLA iteration is not necessarily on the path of the next one).

4. EXPERIMENT

We first demonstrate the efficiency of the proposed SFA for solving FLSA in Section 4.1, and then the proposed EFLA in Section 4.2. All experiments were carried out on an Intel(R)Core(TM)2 Duo CPU (E6850) 2.99GHZ processor. The source codes, included in the SLEP package [18], are available online[4].

4.1 Performance of the Proposed SFA

Illustration of the Proposed SFA We generate a vector $\mathbf{v} \in \mathbb{R}^n$ with $n = 10^5$. The entries in $\mathbf{v}$ are randomly drawn from the standard normal distribution. By applying Theorem 2, we get $\lambda_2^{\max} = 300.2$ with Algorithm 2. All the algorithms start from the origin.

We first compare SFA_R with SFA_G and SFA_N, and present the results in Figure 1. For SFA_G (Algorithm 3) and its accelerated version via the Nesterov's method—SFA_N, we run them for 200 iterations; and for SFA_R (depicted in Figure 2), we terminate the algorithm until the duality gap is zero. From Figure 1, we can observe that: 1) SFA_N converges faster than SFA_G; 2) the duality gaps of SFA_G and SFA_N are not very small after 200 iterations, especially for large λ_2; and 3) SFA_R achieves the exact solution within 8 and 9 iterations for $\lambda_2 = 0.5$ and 1, respectively, and thus significantly outperforms both SFA_G and SFA_N. We attribute the superior performance of SFA_R to the restart technique using the special structures of (16), (17) and (19).

Next, we further explore the performance of SFA_R under different values of λ_2=0.1, 0.5, 1, 2, 3, 5, 10, 20, 200. In Figure 3, we report the duality gap and the number of elements in the *support set* during the iterations. We can observe from this figure that: 1) SFA_R converges within dozens of iterations, and 2) the number of elements in the *support set* decreases with an increasing λ_2.

Comparison with the Other Algorithms Before comparing the proposed SFA with the other algorithms, we first discuss the efficiency of the existing solvers for FLSA. Friedman et al. [8] showed that their proposed modified CD (mCD) outperforms the general solver SQOPT by factors of 50 up to 300 or more. Höefling [12] showed that his pathFLSA[5] is over 100 times faster than the general solver CVX. In addition, pathFLSA is significantly faster than mCD for problems with size up to 10^6. Therefore, in this paper, we compare our proposed SFA_R with pathFLSA.

We try problems of sizes $n = 10^2, 10^3, 10^4, 10^5, 10^6$ and 10^7. For each n, we generate 100 input vectors $\mathbf{v} \in \mathbb{R}^n$, whose entries are randomly drawn from the standard normal distribution, and set $\lambda_2 = r \times \lambda_2^{\max}$, where $\lambda_2^{\max}$ is computed according to Theorem 2, and $r = 10^{-3}, 10^{-2}, 10^{-1}$ and 1. For our proposed SFA_R, we set the origin as the starting point. We report the average results over 100 runs in Table 1, from which we can observe that, 1) our proposed SFA_R is much more efficient than pathFLSA for solving FLSA corresponding to a given parameter λ_2, 2) our proposed algorithm converges within dozens of iterations even for problems of size 10^7, and 3) with an increasing value of $\lambda_2 = r \times \lambda_2^{\max}$, both the number of iterations and the computational time for SFA_R increase, as the starting point (the origin) is much farther away from the solution for the larger λ_2 than that of the small ones.

We would like to emphasize the following two points. First, when the objective is to compute the path solutions for FLSA, pathFLSA should be a better choice than SFA_R, as it is specialized for the path solutions; however, when used as a building block in EFLA, we only need to solve FLSA corresponding to a given λ_2, and thus SFA_R is a better choice. Second, when used as the building block in EFLA, SFA_R can achieve much better practical performance than what reported in Table 1, as we can apply the "warm" start technique, i.e., using the solution obtained from the previous EFLA iteration as the "warm" start for the latter; and we have observed in our experiments that, the average SFA_R iterations is usually within 10 (see the following experiments). However, we note that, neither mCD nor pathFLSA can benefit from the "warm" start technique (see the discussion in Section 3.6).

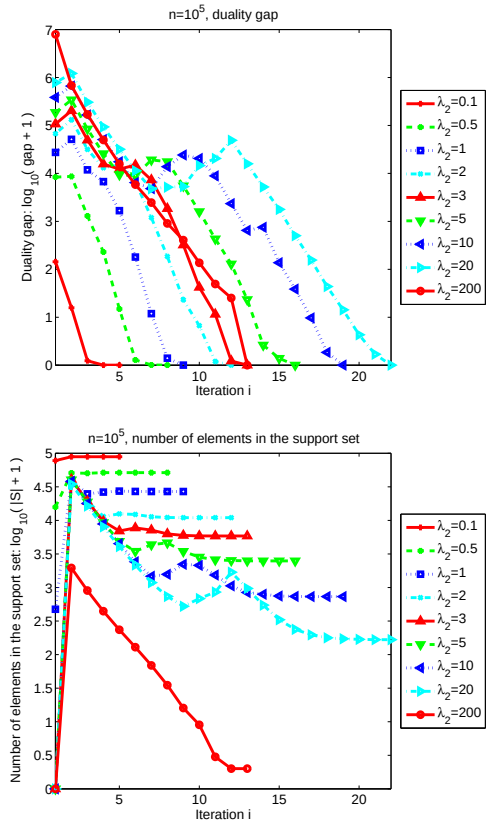

Figure 3: Illustration of SFA_R for solving (19) under different values of $\lambda_2 = 0.1, 0.5, 1, 2, 3, 5, 10, 20, 200$ in terms of the duality gap and the number of elements in the *support set* (shown in the logarithmic scale). In this experiment, $\lambda_2^{\max} = 300.2$. The number of iterations consumed by SFA_R are 5, 8, 9, 12, 13, 16, 19, 22 and 13, respectively; and the number of elements in the *support set* of the minimizer are 88882, 51507, 26970, 10986, 5857, 2492, 731, 166 and 1, respectively.

SFA_R as A Building Block for EFLA with the "Warm" Start To evaluate the efficiency of SFA_R in EFLA, we apply it for solving the following fused Lasso problem:

$$\min_{\mathbf{x}} \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|^2 + \lambda_1\|\mathbf{x}\|_1 + \lambda_2\|R\mathbf{x}\|_1. \qquad (40)$$

Here, $A \in \mathbb{R}^{m\times n}$ is a random matrix whose entries are drawn from the normal distribution with zero mean and unit variance, $\mathbf{b} = A\tilde{\mathbf{x}} + \boldsymbol{\epsilon}$ is the response, $\tilde{\mathbf{x}} \in \mathbb{R}^{n\times 1}$ is a vector whose en-

[4] www.public.asu.edu/~jye02/Software/SLEP
[5] cran.r-project.org/web/packages/flsa

Table 1: Comparison of SFA_R and pathFLSA under different values of n and λ_2. We set $\lambda_2 = r \times \lambda_2^{\max}$, where r is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$. "$\text{SFA}_\text{R}$ Iteration" and "$|S|$" denote the number of iterations consumed by SFA_R, and the number of elements in the *support set* of the minimizer, respectively. "SFA_R Time" and "pathFLSA Time" denote the computation time (in seconds) consumed by SFA_R and pathFLSA, respectively. The reported results are averaged over 100 simulations. "-" denotes that, pathFLSA does not run successfully in our study; note that, the computational time reported in [12] for the same problem size is 108 seconds.

n		10^2	10^3	10^4	10^5	10^6	10^7
10^{-3}	SFA_R Iteration ($\|S\|$)	1 (98)	2 (969)	3 (9025)	7 (70058)	10 (324617)	15 (782902)
	SFA_R Time	2.9×10^{-5}	1.6×10^{-4}	2.1×10^{-3}	4.8×10^{-2}	0.72	9.2
	pathFLSA Time	5.9×10^{-4}	6.1×10^{-3}	6.2×10^{-2}	0.76	9.6	-
10^{-2}	SFA_R Iteration ($\|S\|$)	2 (90)	4 (714)	9 (3363)	14 (7394)	20 (10789)	29 (13092)
	SFA_R Time	3.0×10^{-5}	2.6×10^{-4}	4.2×10^{-3}	5.9×10^{-2}	1.1	14
	pathFLSA Time	6.1×10^{-4}	6.1×10^{-3}	6.1×10^{-2}	0.76	9.6	-
10^{-1}	SFA_R Iteration ($\|S\|$)	5 (34)	9 (85)	16 (113)	24 (119)	32 (127)	42 (139)
	SFA_R Time	4.1×10^{-5}	3.5×10^{-4}	5.1×10^{-3}	8.7×10^{-2}	1.5	20
	pathFLSA Time	6.4×10^{-4}	6.1×10^{-3}	6.1×10^{-2}	0.76	9.7	-
1	SFA_R Iteration ($\|S\|$)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
	SFA_R Time	1.7×10^{-5}	4.8×10^{-5}	3.5×10^{-4}	5.0×10^{-3}	5.4×10^{-2}	0.53
	pathFLSA Time	4.8×10^{-4}	6.1×10^{-3}	6.2×10^{-2}	0.77	9.7	-

Table 2: Computational time (seconds) and average number of iterations when applying SFA_R as a building block for EFLA with the "warm" start.

n	10^3	10^4	10^5
SFA_R average iterations	1.4	2.6	3.5
EFLA time	0.35	5.8	67
SFA_R time	0.13	1.5	21
pathFLSA time	6.1	61	760

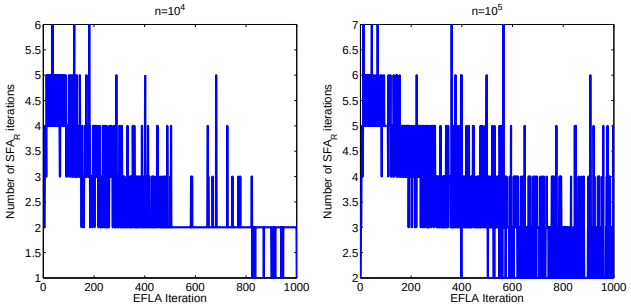

Figure 4: Number of SFA_R iterations when used as a building block for EFLA with the "warm" start.

tries are drawn from the normal distribution with zero mean and unit variance, and $\boldsymbol{\epsilon}$ is the noise vector whose entries are drawn from the normal distribution with zero mean and variance equal to 0.01. We set $m = 100$, $\lambda_1 = \lambda_2 = 0.01$, and try different values of $n = 10^3, 10^4$ and 10^5.

We run EFLA for 1,000 iterations, and report the results in Table 2. We can observe from the second row of this table that, the average number of iterations consumed by SFA_R is very small (within 10). In Figure 4 , we further report the number of SFA_R iterations with increasing EFLA iterations. We observe a similar trend. We attribute the small number of iterations to the usage of the "warm" start, i.e., to solve SFA_R, we use the solution $\mathbf{z}$ in the previous EFLA iteration as the "warm" start for the successive one.

In rows 3 and 4 of Table 2, we report the the computational time consumed by EFLA and SFA_R, from which we can observe that SFA_R consumes about 1/3 of the total computational time for the above experiments. If we apply pathFLSA [12] for fulfilling the same task as SFA_R, it consumes much more computational time, as shown in the last row of Table 2; and this shall make the EFLA much slower than that with SFA_R. From the results in Tables 1 and 2, we can conclude that our proposed SFA_R is much more efficient than pathFLSA for solving FLSA (which acts as a building block in the proposed EFLA) corresponding to a given parameter, especially when the "warm" start technique is applied.

4.2 Performance of the Proposed EFLA

We apply the proposed EFLA to several real world applications, with the least squares loss. Specifically, we solve the fused Lasso (40), where each row of A denotes a sample, and each row of $\mathbf{b}$ contains the corresponding class label information.

Data Description We conduct experiments on the following three data sets: ArrayCGH [28], Prostate Cancer [24], and Leukemias [9].

The ArrayCGH (Array comparative genomic hybridization) [28] is a micro-array based technology that can detect genomic copy number variation (CNV) at different locations along the genome. Each measurement (feature) is the log ratio of CNV, and adjacent features correspond to adjacent locations along the genome. This data set contains ArrayCGH profiles of 57 bladder tumor samples, and each profile contains 2,385 measurements. Here we consider the tumor grade classification problem, with 12 samples of Grade 1 and 45 samples of higher grades (2 or 3).

The Prostate Cancer [24] data set used in our experiments is based on the protein mass spectrometry, where the features are indexed by many time-of-flight values. Time of flight is related to the mass over charge ratio m/z of the constituent proteins in the blood. The data set contains 15,154 measurements of 132 patients, including 63 healthy and 69 with prostate cancer.

The Leukemias [3, 9] is a DNA microarray data set. It contains 7,129 genes and 72 samples: 47 of acute lymphocytic leukaemia and 25 of acute myelogenous leukaemia. Unlike the ArrayCGH and Prostate Cancer data sets, the features in Leukemias have no prespecified order [30]. We follow [30] to reorder the features in the data, using the binary hierarchical clustering; and we call the resulting data set as "Leukemias Reordered".

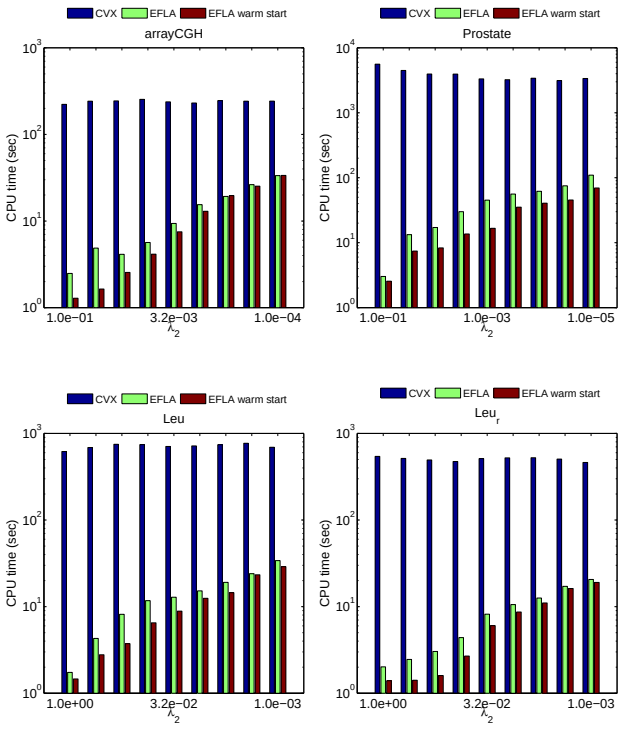

Figure 5: Computational time (seconds) on ArrayCGH (top left), Prostate Cancer (top right), Leukemias (bottom left), and Leukemias Reordered (bottom right).The x-axis denotes the different values of λ_2, while the y-axis represents the total computational time (seconds) corresponding the given λ_2. For illustration convenience, the y-axis is plotted in the logarithmic scale.

Computational Efficiency of EFLA We compare our proposed EFLA with the CVX optimization package [10], for solving the fused Lasso (40). In the comparison, we terminate EFLA till it achieves an objective function value less than or equal to that of CVX. The parameters λ_1 and λ_2 are specified by a 9×9 grid sampled using the logarithmic scale from the parameter space. We report the computational time (seconds) in Figure 5, where the x-axis denotes the different values of λ_2, and the y-axis represents the total computational time (seconds) corresponding to the given λ_2. For a given λ_2, we can solve (40) by applying the solution corresponding to the large λ_1 as the "warm"-start to the smaller one; and this is the so-called "warm" start technique widely employed in the literature [8, 16]. From Figure 5, we can observe that EFLA is one or two orders of magnitude faster than the standard solver CVX, especially for larger λ_2. In addition, the"warm" start helps improve the efficiency. The superior efficiency of EFLA attributes to the following reasons: 1) EFLA directly solves the composite function (40) utilizing the composite structure, while CVX is a general solver optimizing the smooth reformulation of (40) by introducing many additional variables and constraints; 2) EFLA enjoys the optimal convergence rate for the first-order black-box methods; 3) the SFA_R developed in Section 3 can efficiently solve FLSA, the key building block of the proposed EFLA (see Tables 1 & 2).

Classification Performance We follow [25] to report the leave-one-out performance of the fused Lasso (via EFLA). The parameters λ_1 and λ_2 are specified by a 9×9 grid sampled using the logarithmic scale from the parameter space. The classification errors corresponding to different parameter values are visualized by a heat map in Figure 6. We also report in Table 3 the best leave-one-out accuracy (%) on different data sets by Fused Lasso and Lasso; and we can observe that Fused Lasso can achieve comparable or better classification performance than Lasso, benefited by the additional fused penalty $\|R\mathbf{x}\|_1$. We refer the readers to [30] for detailed comparison between Lasso and Fused Lasso, and [25, 31] for the biological interpretation.

Table 3: The best leave-one-out accuracy (%) on different data sets by Fused Lasso and Lasso.

	Fused Lasso	Lasso
Array CGH	88%	82%
Prostate Cancer	98%	98%
Leukemias	96%	94%
Leukemias Reordered	97%	94%

5. CONCLUSION

In this paper, we consider solving the class of problems with the fused Lasso penalty, leading to a class of non-smooth and non-separable optimization problems. By treating its objective function as the composite function (composed of one smooth part and the other non-smooth part), we propose to apply the Nesterov's method to develop the Efficient Fused Lasso Algorithm (EFLA), achieving the optimal convergence rate of $O(1/k^2)$. In the proposed EFLA, a key building block in each iteration is FLSA, for which we propose an efficient Subgradient Finding Algorithm (SFA), equipped with a restart technique for fast convergence. When used as a building block in EFLA, SFA is shown to converge within 10 iterations with the "warm" start. Our empirical evaluations show that, both SFA and EFLA significantly outperform the existing solvers, thus making it applicable for large-scale problems.

We plan to apply the proposed EFLA for the construction of time-varying networks [1], and other applications with either spatially or temporally ordered features. In addition, we plan to develop the online and stochastic versions of EFLA using FOLOS [6] and RDA [35], where the proposed SFA again acts as a building block. Finally, we plan to extend our methodology to multidimensional fused Lasso, where the features form more complex graph structures, e.g., the two-dimensional fused Lasso [8].

6. ACKNOWLEDGEMENT

This work was supported by NSF IIS-0612069, IIS-0812551, CCF-0811790, NGA HM1582-08-1-0016, NSFC 60905035, and the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the US Army.

7. REFERENCES

[1] A. Ahmed and E. P. Xing. Recovering time-varying networks of dependencies in social and biological studies. *Proceedings of the National Academy of Sciences*, 106(29):11878–11883, 2009.

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

[3] C. Chang and C. Lin. Libsvm : a library for support vector machines. *http://www.csie.ntu.edu.tw/ cjlin/libsvm*, 2001.

[4] Z. Dostál. Box constrained quadratic programming with proportioning and projections. *SIAM Journal on Optimization*, 7(3):871–887, 1997.

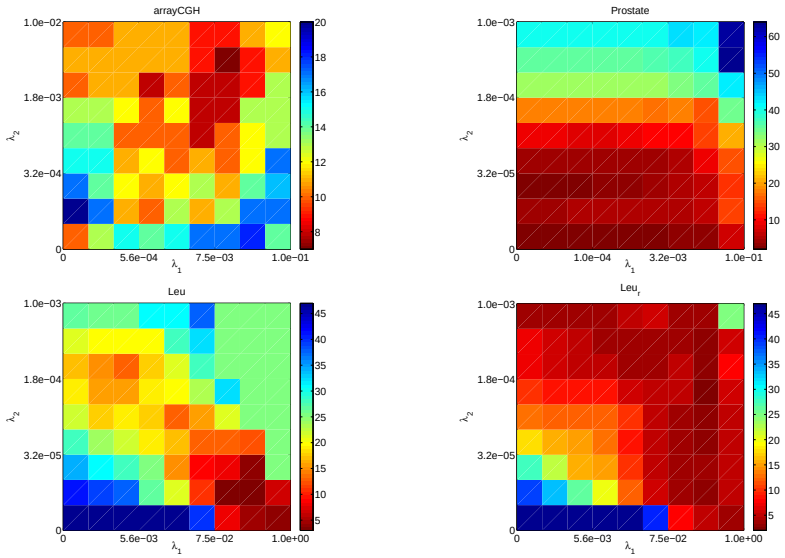

Figure 6: The Leave-one-out test error of the Fused Lasso (via EFLA) on ArrayCGH (top left), Prostate Cancer (top right), Leukemias (bottom left), and Leukemias Reordered (bottom right). For each plot, the x-axis corresponds to λ_1, while the y-axis corresponds to λ_2. The color indicates the number of leave-one-out errors, with dark blue indicating more errors and red indicating fewer errors.

[5] Z. Dostál and J. Schöberl. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Computational Optimization and Applications*, 30(1):23–43, 2005.

[6] J. Duchi and Y. Singer. Online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 2009.

[7] D. J. Evans. An algorithm for the solution of certain tridiagonal systems of linear equations. *The Computer Journal*, 15(4):356–359, 1972.

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

[9] T. R. Golub and et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[10] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, 2009.

[11] J. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I & II*. Springer Verlag, Berlin, 1993.

[12] H. Höfling. A path algorithm for the fused lasso signal approximator. *arXiv*, 2009.

[13] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *International Conference on Machine Learning*, 2009.

[14] C. Lemaréchal and C. Sagastizábal. Practical aspects of the moreau-yosida regularization I: Theoretical properties. *SIAM Journal on Optimization*, 7(2):367–385, 1997.

[15] C. J. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999.

[16] J. Liu, J. Chen, and J. Ye. Large-scale sparse logistic regression. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, 2009.

[17] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. In *Uncertainty in Artificial Intelligence*, 2009.

[18] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

[19] J.-J. Moreau. Proximité et dualité dans un espace hilbertien. *Bull. Soc. Math. France*, 93:273–299, 1965.

[20] A. Nemirovski. *Efficient methods in convex programming*. Lecture Notes, 1994.

[21] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2003.

[22] Y. Nesterov. Gradient methods for minimizing composite objective function. *CORE Discussion Paper*, 2007.

[23] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

[24] E. Petricoin and et al. Serum proteomic patterns for detection of prostate cancer. *Journal of National Cancer Institute*, 94(20):1576–1578, 2002.

[25] F. Rapaport, E. Barillot, and J. Vert. Classification of arrayCGH data using fused svm. *Bioinformatics*, 24(13):i375–i382, 2008.

[26] A. Rinaldo. Properties and refinements of the fused lasso. *Annals of Statistics*, 37(5B):2922–2952, 2009.

[27] D. J. Rose. An algorithm for solving a special class of tridiagonal systems of linear equations. *Communications of the ACM*, 12(4):234–236, 1969.

[28] N. Stransky and et al. Regional copy number-independent deregulation of transcription in cancer. *Nature Genetics*, 38(12):1386–1396, 2006.

[29] L. H. Thomas. *Elliptic Problems in Linear Difference Equations over a Network*. Waston Scientific Computing Laboratory, Columbia, NY, 1949.

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

[31] R. Tibshirani and P. Wang. Spatial smoothing and hot spot detection for CGH data using the fused lasso. *Biostatistics*, 9(1):18–29, 2008.

[32] K. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, to appear, 2009.

[33] P. Tseng. Convergence of block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:474–494, 2001.

[34] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1):387–423, 2009.

[35] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. In *Advances in Neural Information Processing Systems*, 2009.