ALTERNATING LINEARIZATION FOR STRUCTURED REGULARIZATION PROBLEMS

BY MINH PHAM

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Operations Research

Written under the direction of

Andrzej Ruszczyński - Xiaodong Lin

and approved by

New Brunswick, New Jersey

May, 2014

ABSTRACT OF THE DISSERTATION

Alternating linearization for structured regularization problems

by Minh Pham

Dissertation Director: Andrzej Ruszczyński - Xiaodong Lin

We adapt the alternating linearization method for proximal decomposition to structured regularization problems. The method is related to two well-known operator splitting methods, the Douglas-Rachford and the Peaceman-Rachford method, but it has descent properties with respect to the objective function. Its convergence mechanism is related to that of bundle methods of nonsmooth optimization. A block coordinate descent method is developed to facilitate fast convergence. We also discuss implementation for large problems, with the use of specialized algorithms and sparse data structures. We present numerical results for several synthetic and real-world examples, including a three-dimensional fused lasso problem, which illustrate the scalability, efficacy, and accuracy of the method. We further extend the alternating linearization framework to the structured regularization problems with non-convex penalties. In this framework, the non-convex part of the objective function is approximated via a linear model. Therefore, in each iteration, the method solves a structured regularization problem. We present several numerical studies with synthetic data and cancer research data.

Acknowledgements

Dr. Andrzej Ruszczyński and Dr. Xiaodong Lin, my advisors, for their dedication and encouragement of this long term project. I was introduced to the fields of optimization and machine learning in the Fall of 2010 when I was at the beginning of the fourth year in graduate school. At that point, the thought of quitting school had been constantly on my mind since I could not find a topic that was interesting enough for me. Fortunately, Andrzej and Xiaodong got me to join them in a project whose goal was to developing an optimization framework for a class of regularization problem and I have never looked back. Throughout the project, Andrzej and Xiaodong, as leading researchers in their fields, have been a great source of knowledge and support.

Dr. MK Jeong, for his support and understanding for the five years I spent at RUTCOR. I look forward to further collaboration with him in the future.

Dr. Endre Boros, the chair of RUTCOR, for his encouragement and dedication towards all of the graduate students.

Anh Ninh, my best friend in graduate school for being not only a brother, a peer scientist, and a supporter. I want to express my gratitude towards his being around through all the ups and downs. Without him, this thesis will never be completed.

Clare, Terry, Lynn, Katie, and Chong, without you, there would be no RUTCOR. Thank you for your hard work to keep us graduate students in check.

My parents, my brother and his wife, for being supportive of me in the last ten years in the US.

Dedication

For my parents and my brother.

Table of Contents

Abstract ii

Acknowledgements iii

Dedication iv

1. Introduction 1

2. Preliminaries 3

2.1. Literature review 3

2.2. Mathematical Notation 9

2.3. Selected optimization methods 10

2.3.1. Block coordinate descent method 10

2.3.2. Conjugate gradient method 11

2.4. Problem background 12

3. Convex penalties 14

3.1. Introduction 14

3.2. Optimality condition 17

3.3. Alternating linearization method 19

3.3.1. Outline of the method 19

3.3.2. Convergence 21

3.4. Application to lasso regression 21

3.5. Application to general structured regularization problems 23

3.6. Numerical experiments 26

3.6.1. ℓ_1 regularization 26

3.6.2. Fused Lasso regularization 28

3.6.3. CGH data example 31

3.6.4. Wavelet based and Total variation based image reconstruction 32

3.6.5. Application to a narrative comprehension study for children 38

4. Nonconvex penalties 44

4.1. Introduction 44

4.2. Alternating linearization method for nonconvex structured penalties 47

4.2.1. Structured nonconvex penalties 47

4.2.2. Logistic regression with nonconvex penalties 49

4.2.3. ALIN for nonconvex penalties with Gaussian loss and Logistic loss 50

4.2.4. ALIN for structured nonconvex penalties 51

4.2.5. Nonlinear conjugate gradient method 52

4.3. Numerical examples 53

4.3.1. Simulation studies - Linear regression 53

4.3.2. Simulation studies - Logistic regression 58

4.3.3. Cancer research data 60

4.4. Conclusion 63

Chapter 1
Introduction

Optimization techniques devised for achieving parsimonious model estimations have drawn great attention across different communities including statistics, computer science, and operations research. The reemergence of these regularization methods is the driving force for recent advances in statistical learning and machine learning. These methods have a long history dating back to Tikhonov's contribution for solving ill-posed problems. Regularization refers to the introduction of additional information, usually in the form of a penalty for solving an ill-posed inverse problem to prevent overfitting. The ℓ_1 penalty is one of the most well-known and widely-used type of penalty functions in practice. There are several reasons to its popularity. The solution to the problem is usually sparse since the important features remain in the model and non-important features are shrinked to zero. Computation of the solution is also simple due to the separability of the penalty. The coordinate descent method has achieved huge success in solving this formulation by exploiting the separability of the penalty.

In many applications, especially in image processing, brain imaging, or biology, certain physical and domain constraints might impose structures on the solution. This extra piece of information is crucial to the solvability of the problem and the quality of the solution. However, a nonsmooth non-separable penalty function makes the optimization problem hard to solve. In addition, the large size of the resulting optimization problems, in the order of millions of variables, poses a huge computational challenge. An optimization method for this class of problem is required to be fast, reliable, accurate and scalable.

This dissertation focuses on developing an unified optimization framework based on alternating linearization for the structural regularization problem. Chapter one and two will give an introduction on the problem of interests, literature reviews, and several selected optimization techniques that play an important role in the success of our framework. Chapter three introduces the idea of alternating linearization and the application of the method to the *Generalized Lasso* problem. Numerical studies involving interesting problems in neuro-imaging and image processing are also provided in this chapter. Chapter four presents an extension of alternating linearization to solve the regularization problem with structured non-convex penalties. Structured non-convex penalty is a new concept although the convex equivalence has been studied rigorously. Chapter four also provides some numerical studies on synthetic data and cancer research data to show that structured non-convex penalty can boost the perfomance of current supersived learning methods. Overall, numerical studies show that alternating linearization can be very competitive against competing methods in the literature, in most cases outperform them by a factor of two or three in terms of running time while still providing accurate solutions.

Chapter 2

Preliminaries

2.1 Literature review

First, we consider the class of *Generalized Lasso* regularization problem defined as follows:

$$\min_{\beta \in \mathcal{R}^p} f(\beta) + \lambda \|R\beta\|_1,$$

where $f(\beta)$ is a loss function, $R \in \mathcal{R}^{k \times p}$ is a specified penalty matrix. The matrix R is determined according to the structural information from the domain and the nature of the data set so that $R\beta$ has some desired parsimonious structure. Several well-known regularization problems can be casted in this form. For specific choices of matrix R, many efficient algorithms have been proposed to solve the problem. These algorithms can be categorized into four main frameworks: path algorithm, proximal gradient algorithm, alternating direction algorithm, and methods for constrained optimization. There are a few exceptions in the case $R = \mathbb{I}$ but these methods are either not directly applicable or not very efficient in the general framework.

When $R = \mathbb{I}$, the resulting ℓ_1-regularization is the most simple form of the structured regularization problem. It has been applied extensively in almost all learning problems from both supervised and unsupervised perspective. A large amount of literature has been devoted to studying efficient computation of the solutions and its theoretical properties. There are a few reasons for this popularity. First, it can be formulated as a convex programming problem and there exists many methods in optimization to solve the formulation efficiently. Moreover, especially when the dimensions of data sets are large, ℓ_1 penalty can be used to obtain a parsimonious model by letting significant features into the model and shrinking others to 0. The formulation of a ℓ_1 regularization problem takes the form:

$$\min_{\beta} f(\beta) + \lambda \sum_i |\beta_i|$$

where $f(\beta)$ is a convex loss function and λ is a positive parameter.

In the context of linear regression, $f(\beta)$ is the least square loss. In logistic regression and support vector machine, $f(\beta)$ are the logistic loss function and hinge loss respectively. The tuning parameter λ is used to balanced regularization and loss terms. The problem in general produces a sparse solution, and the degree of sparsity depends on the magnitude of λ.

Many efficient techniques have been proposed to solve this problem. The earlier and more straight-forward approach is to reformulate it as a constrained optimization problem. In the least square setup, we solve the following constrained optimization problem:

$$\min_\beta \tfrac{1}{2}\|y - X\beta\|_2^2$$
$$\text{subject to: } \textstyle\sum_i^p |\beta_i| \le t$$

where $X \in \mathcal{R}^{n\times p}$ is the design matrix, $y \in \mathcal{R}^{n\times 1}$ is the vector of response variables, and $\beta \in \mathcal{R}^{p\times 1}$ is the vector of coefficients. In this approach [Tibshirani, 1996], the absolute value constraint is replaced by a number of linear inequality constraints with new variables introduced. The quadratic programming problem can be solved by many optimization methods including interior point and active-set methods. This approach, however, can not be directly applied to more general setting such as Generalized Linear Models (GLM). Moreover, in high dimensional scenarios, solving such a large quadratic programming problem is computational challenging. [Tibshirani, 1996] also proposed an alternative that works with the unconstrained formulation. It is an iterative method where in each iteration, $|\beta_i|$ is approximated by $\frac{\beta_i^2}{|\beta_i|}$. Then a solution for each iteration can be obtained with closed form formula. This method is efficient but proved to be numerically unstable. In another line of work, [Kim et al., 2007] and [Koh et al., 2007] proposed an interior-point approach for solving large scale ℓ_1 regularized problem with logistic loss and least square loss to solve the constrained quadratic programming. They used preconditioned conjugated gradient method to solve the sub-problems. Another interesting method that can be used to solve the box constrained quadratic programming problem is the spectral gradient method [Birgin and Martínez, 2002]. In this method, each iteration moves from a face of the box constraints to another to find a new estimate of the solution. In each face, a modified conjugate gradient method is used to find the best estimate. The algorithm terminates when the spectral gradient is small. This method tends to have slow convergence when X has $p \gg n$.

The most popular method in many aspects such as efficiency, ease of implementation, and practicality is the coordinate descent method. Coordinate descent is a classical method in optimization. In each iteration, it keeps all but one variable fixed and the task remained is minimizing a one-variable function. For the least square loss function, the sub-problem is very simple and has a closed form solution. The variables can be chosen in a cyclic manner (Gauss-Seidel method) [Tseng and Yun, 2007]. Some other important related works are [Wu and Lange, 2008] and [Friedman et al., 2007]. Although, the method works very well for least square loss function, its extension to logistic loss function and hinge loss is not straight-forward. A more detailed review in this aspect is given in [Yuan et al., 2010].

Another popular methods for the ℓ_1 regularized problem are proximal gradient methods and path-following methods. As described in [Liu et al., 2009], [Becker et al., 2011], and [Beck and Teboulle, 2009], proximal gradient method relies on approximating the objective function with a quadratic function at a carefully chosen point. This method scales very well and is very popular in image processing community due to its capability of dealing with a large number of variables. Another advantage of this method is its flexibility. Given that the proximity operator can be calculated efficiently, it can be applied to a wide range of penalty functions.

Path-following algorithms produce the whole piecewise linear solution path with respect to the parameter λ. These methods are very efficient although some can not be extended to other types of loss function such as logistic loss. Moreover, they tend to be sensitive when the independent varibles have high degree of multicollinearity. [Efron et al., 2004], [Rosset and Zhu, 2007], [Zhao and Yu., 2007] are examples of this type of algorithms. A modified Least Angle Regression (LAR) algorithm was one of the earliest methods invented to solve the Lasso problem. It can compute the entire path of lasso solution for λ varied from 0 to ∞.The algorithm starts with all coefficients β equal to 0. The coefficient corresponding to the predictor with most correlation with the residual is picked and moved from 0 towards its least square estimate until another predictor has as much correlation with the current residual is selected. The method iterates until all coefficients have been estimated. For a Lasso coefficient path, there are only a few critical points where new predictors enter the model.

For many practical applications, physical constraints and domain knowledge may mandate additional structural constraints on the parameters. For example, in cancer research, it may be important to consider groups of interacting genes in each pathway rather than individual genes. In image analysis, it is natural to regulate the differences between neighboring pixels in order to achieve smoothness and reduce noise. In light of these popular demands, a variety of structured penalties have been proposed to incorporate prior information regarding model parameters. One of the most important structural penalties is the *fused lasso* proposed in [Tibshirani et al., 2005]. It utilizes the natural ordering of input variables to achieve parsimonious parameter estimation on neighboring coefficients. This penalty was used with much success in spatial smoothing and hotspot detection of CGH data in cancer research. [Tibshirani et al., 2005] formulated this problem as a constrained quadratic programming with sparse linear constraints. [Rapaport et al., 2008] proposed a similar *Fused Lasso* formulation in support vector machine. In these approaches, the numbers of constraints can extremely highg which makes conventional quadratic programming solver infeasible. Several attempts were made to solve this fused lasso penalty using path algorithm. [Hoefling, 2010] proposed a fast path algorithm for Fused Lasso Signal Approximator when $X = \mathbb{I}$. The method can be extended to solve more complex design matrix $X \in \mathcal{R}^{m \times p}$ when $rank(X) = p$. However, the bottleneck of this approach is the max flow algorithm. Although the method was shown to be faster than the quadratic programming formulation, it is still very slow for high dimensional settings.

Another framework that attracts much attention is the proximal gradient method with acceleration [Nesterov, 2007]. This method is theoretically and practically attractive since it was shown to achieve the best convergence rate for first-order methods. Moreover, it is very flexible and easy to implement. SLEP [Liu et al., 2010b] is a very efficient method built on this framework for solving the fused lasso problem. SLEP proposed a very intelligent gradient descent method to solve the sub-problem:

$$\min_{x \in \mathcal{R}^p} f(y) + \langle x - y, \nabla f(y) \rangle + \frac{L}{2} \|x - y\|_2^2 + \sum_{i=1}^{p-1} |x_{i+1} - x_i|.$$

which is usually the overall bottleneck. Chen et al. [2010] developed the *graph induced fused lasso* that penalizes differences between coefficients associated with nodes in a graph that are connected. Chen's method approximates the non-smooth penalty by a smooth function and utilize gradient method with acceleration.

Alternating direction method of multipliers (ADMM) is another important framework for this type of fused lasso penalty. It is also very flexible and easy to implement. Moreover, it can utilize distributed computing facility for very large scale problem [Boyd et al., 2010]. Split Bregman method belongs to this category [Ye and Xie, 2011]. All these methods are very efficient and can handle large scale data sets. However, the performance of this implementation relies heavily on the choice of tuning parameters. Poor choice of the parameters can lead to very low quality solution. In many practical studies, the convergence of this method can not be guaranteed.

Discrete total variation penalty is a widely-used regularization in image processing which also belongs to the class of *Generalized Lasso*. This penalty penalizes the difference between a pixel and its neighboring pixels in an image. Thus, a high quality image can be restored from a noisy and blurred observered image. Beck and Teboulle [2009] proposed the *total variation penalty* for image denoising and deblurring, in a similar fashion to the two-dimensional fused lasso. This method is based on proximal gradient method with acceleration. It is considered one of the best methods for this line of work. Several methods for total variation deblurring based on ADMM can be found in [B. Wahlberg, 2012] [C. Li and Zhang, 2013] [D. Goldfarb and Scheinberg., 2013] [Z. Qin and Ma.].

Similar penalty functions have been successfully applied to several neuroimaging studies [Michel et al., 2011, Grosenick et al., 2011, 2013]. More recently, Zhang et al. [2012] applied a generalized version of fused lasso to reconstruct gene copy number variant regions. A general structural lasso framework was proposed in [Tibshirani and Taylor, 2011], with the following form:

$$\mathcal{L}(\beta) = f(\beta) + \lambda \|R\beta\|_1, \quad \lambda > 0, \tag{2.1}$$

where R is an $k \times p$ matrix that defines the structural constraints one wants to impose on the coefficients. Many regularization problems, including high dimensional fused lasso and graph induced fused lasso, can be formulated in this framework.

In general, the iterative methods that have been proposed to solve the *Generalized Lasso* can be cast into three categories: path algorithm, proximal gradient method, and alternating direction methods and its related algorithms. Several path algorithms have also been proposed to compute the whole regularization path for the general fused lasso problem. Hoefling [2010] developed a path algorithm for solving (4.1) when the matrix $X^T X$ is nonsingular. This technique is not applicable to cases with large dimension of β and small number of observations, such as gene expression and brain imaging data sets. Tibshirani and Taylor [2011] extended the path algorithm to include all design matrices X, by computing the regularization path of the dual problem. Although fairly general, this version of the path algorithm does not scale well with data dimension, as the knots of the piecewise linear solution path become very dense. [Zhou and Wu, 2012] proposed a generic path algorithm for *Generalized Lasso*, however this method requires certain conditions on the rank of the structured matrix R.

The proximal gradient is a very popular method among machine learning community. It is mainly based on Nesterov's method of accelerated gradient [Nesterov, 2007]. A very successful variation is Fast Iterative Shrinkage-Thresholding Algorithm (FISTA), a method used for image deblurring with discrete total variation penalty. FISTA minimizes an objective function that is a composite of two functions: a smooth loss function and a non-smooth penalty function. In each iteration of FISTA, the smooth loss function is replaced by a linearization at a specific point and a seperable quadratic term.

$$\min_x f(y) + \langle x - y, \nabla f(y) \rangle + \frac{L}{2}\|x - y\|_2^2 + \|Rx\|_1.$$

This specific point is chosen as a combination of the two previous estimates of the solution.

$$y_{k+1} = x_k + \frac{k-2}{k+1}(x_k - x_{k-1}).$$

FISTA is shown to have the optimal convergence rate among the first-order methods. However, FISTA's performance depends heavily on approximating the Lipschitz constant of ∇f. This is not very hard to do in certain image deblurring problem, but in the $p \gg n$ setting of the linear inverse problem, FISTA can be very slow. In this setting, the Lipschitz constant of ∇f can not be calculated easily so FISTA has to rely on back-tracking to find the right step-size for each iteration. This can slow down the algorithm significantly.

Many of the proposed approaches are versions of the *operator splitting methods* or their dual versions, *alternating direction methods* (see, e.g., Boyd et al. [2010], Combettes and Pesquet [2010], and the references therein). The alternating direction method of multipliers (ADMM) considers the problem:

$$\min_{x,z} f(x) + g(z) \text{ s.t}: Ax + Bz = c$$

In the context of the *Generalized Lasso* problem, the formulation is:

$$\min_x \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|z\|_1 \text{ s.t : } Rx - z = 0.$$

The update rule is :

$$\begin{aligned} x^{k+1} &= (A^TA+\rho F^tF)^{-1}(A^Tb+\rho F^tz^k - F^Ty^k). \\ z^{k+1} &= S_{\lambda/\rho}(Fx^{k+1}+y^k/\rho). \\ y^{k+1} &= y^k+\rho(Fx^{k+1}-z^{k+1}). \end{aligned}$$

for ρ is a parameter, $S_{\lambda/\rho}(x) = sign(x)(|x|-\lambda/\rho)_+$. In every iteration of ADMM, the Augmented Lagrangian is minimized with respect to each of the two primal variables x and z. Following these is a update step for the dual variable. This strategy allows ADMM to exploit the structures of functions f and g which are very common in many learning problems. With proper structures, ADMM can be implemented for distributed system to deal with data sets of high dimensions. ADMM and many related methods have been applied to succesfully solve a wide variety of problems such as penalized least square, image denoising and deblurring, group Lasso, and regularized logistic regression [Boyd et al., 2010], [Afonso et al., 2010], [Goldstein and Osher, 2009]. Although fairly general and universal, they frequently suffer from slow tail convergence (see [He and Yuan, 2011] and the references therein).

2.2 Mathematical Notation

The set of real number is denoted by $\mathcal{R}$ and the $n-$dimensional Euclidean space by $\mathcal{R}^n$. $\mathcal{R}^{m\times n}$ denotes the set of $m \times n$ real matrices. Superscript A^T denotes the transpose of the matrix A. The inner product of two vectors x,y $\in \mathcal{R}^n$ is denoted by:$< x, y >= x^Ty = \sum_{j=1}^n x_jy_j$. The Euclidean norm of $x \in \mathcal{R}$ is denoted by $\|x\|_2$. $\|x\|_0$ denotes the ℓ_0 norm of x. $\|x\|_1$ denotes the ℓ_0 norm of x: $\|x\|_1 = \sum_{j=1}^n |x_j|$. $\|x\|_\infty$ denotes the ℓ_∞ norm of x or the largest component of x.

$Diag(s)$ denotes the matrix with the main diagonal being s or the diagonal of the matrix s. For $x \in \mathcal{R}$, $sign(x)$ is the sign function of x.

$$sign(x) = \begin{cases} +1 & \text{if } x > 0. \\ 0 & \text{if } x = 0. \\ -1 & \text{if } x < 0. \end{cases}$$

2.3 Selected optimization methods

In this section, we present several building blocks of our computation framework, they are essential components of our method.

2.3.1 Block coordinate descent method

Block coordinate descent method (BCD) is one of the oldest methods in optimization. The method has recently gained much attention to solve large scale problems in machine learning and image processing. BCD is used to solve the following problem:

$$\min_{x} \ f(x) \text{ subject to x} \in X \tag{2.2}$$

where $f(x)$ is a convex function and X is the product of closed convex sets $X_1, X_2, \cdots, X_m$. Decision variable vector x is partitioned into m blocks:

$$x = (x_1, x_2, \cdots, x_m) \tag{2.3}$$

The constraint $x \in X$ is is required to have a special separable structure:

$$x_i \in X_i \, , i = 1, 2, \cdots, m \tag{2.4}$$

We assume that $\forall x \in X$ and $\forall i = 1, 2, \cdots, m$, the following optimization subproblem has a solution :

$$\min_{\xi} \ f(x_1, \cdots, x_{i-1}, \xi, x_{i+1}, \cdots, x_m) \text{ subject to } \xi \in X_i \tag{2.5}$$

Block coordinate descent algorithm, given the estimate at iteration k, $x^k = (x_1^k, x_2^k, \cdot, x_m^k)$ provides the following estimate in iteration $k+1$:

$$x^{k+1} = \arg\min_{\xi} f(x_1^k, \cdots, x_{i-1}^k, \xi, x_{i+1}^k, \cdots, x_m^k) \text{ subject to } \xi \in X_i \tag{2.6}$$

The starting point x^0 is chosen in X. The blocks are processed in cyclical order or updated simultaneous. BCD works in practice with large scale problems only if minimization of the subproblem is fairly easy. This is usually the case when x_i is a low dimensional vector.

Theorem 1 *[Ruszczyński, 2006] Assume that the function f is continuously differentiable over the the set X. Moreover, assume that for every block i and $x \in X$ the minimization problem has a unique solution:*

$$\min_{\xi} \; f(x_1, \cdots, x_{i-1}, \xi, x_{i+1}, \cdots, x_m) \textit{ subject to } \xi \in X_i \tag{2.7}$$

Let x^k be the sequence generated by the block coordinate descent method. Then, every limit point of x^k is a stationary point.

In situations where the objective function and constraints have partially decomposable structure in terms of the decision variables, block coordinate descent method is extremely efficient. This will be demonstrated later with numerical examples.

2.3.2 Conjugate gradient method

Conjugate gradient method can be used to efficiently minimize a quadratic function:

$$f(x) = \frac{1}{2}x^T Q x + c^T x \tag{2.8}$$

where x $\in \mathcal{R}^n$, Q is a $n \times n$ positive definite matrix.

The idea of conjugate gradient method is successively minimize $f(x)$ along conjugate directions. These conjugate directions can be found with or without knowledge of the Hessian Q. The method can easily be applied to nonquadratic functions since any twice continuously differentiable function can be approximated by a quadratic model in some neighborhood of its minimum point.

Theorem 2 *Assume that $d^1, d^2, \cdots, d^n$ are conjugate directions and that the sequence $x^1, x^2, \cdots, x^{n+1}$ is obtained by successive minimization of function f(x) in directions d^k, $k = 1, 2, \cdots, n$:*

$$\begin{aligned} x^{k+1} &= x^k + \tau_k d^k, \quad \textit{and} \\ f(x^{k+1}) &= \min_{\tau \in \mathcal{R}} f(x^k + \tau d^k), \end{aligned}$$

then for every $k = 1, 2, \cdots, n$ the points x^{k+1} is the minimum of $f(x)$ in the linear manifold

$$L_k = x^1 + lin\{d^1, d^2, \cdots, d^n\}. \tag{2.9}$$

The minimum of f(x) can be found in no more than n steps.

Conjugate gradient method is described with details in the following.

- **Step 0:** *Set k=1.*
- **Step 1:** *Calculate* $\nabla f(x^k)$. *If* $\nabla f(x^k) = 0$ *then stop; otherwise continue.*
- **Step 2:** *Calculate*

  $$d^k = \begin{cases} -\nabla f(x^k) & k = 1 \\ -\nabla f(x^k) + \alpha_k d^{k-1} & k > 1 \end{cases}$$

  with

  $$\alpha_k = \frac{\langle \nabla f(x^k), \nabla f(x^k) - \nabla f(x^{k-1} \rangle}{\|\nabla f(x^{k-1}\|^2}.$$

- **Step3:** *Calculate the next point*

  $$x^{k+1} = x^k + \tau_k d^k \tag{2.10}$$

  such that

  $$f(x^{k+1}) = \min_{\tau \geq 0} f(x^k + \tau d^k). \tag{2.11}$$

- **Step 4:** *Increase k by 1 and go to Step 1.*

2.4 Problem background

In this thesis, I am proposing a unified framework to solve the problem:

$$\min_{\beta} \mathcal{L}(\beta) = f(\beta) + \lambda \|R\beta\|_1, \quad \lambda > 0$$

In a broader view, an more general problem is:

$$\min_{\beta} \mathcal{L}(\beta) = f(\beta) + \lambda \|R\beta\|_\Diamond, \quad \lambda > 0,$$

where $\|.\|_\Diamond$ is a norm in $\mathcal{R}^p$, and $f(\beta)$ is a convex loss function. In the context of the regression problem, we have a matrix $X \in \mathcal{R}^{n \times p}$ of independent variables, a vector $y \in \mathcal{R}^{n \times 1}$ is the vector of the responses. Note that y can be a binary response vector. For linear regression, the loss function $f(\beta)$ is the traditional least square loss function, $f(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$. If the response vector y is binary, we have a logistic regression problem with logistic loss function $f(\beta) = \sum_{i=1}^{n} \log(1 + e^{-y_i X_i^T \beta})$. In this thesis, the method is presented to work with smooth convex loss function but it can be extended to nonsmooth loss functions as well.

The matrix $R \in \mathcal{R}^{k \times p}$ may have many different forms. When $R = \mathbb{I}$, the identity matrix, we have the Lasso regularization. When R is the linear operator that takes the difference between neighboring coefficients of β, we have the fusion penalty. The fusion penalty can also be generalized to problems in $2 - d$ (image processing) or $3 - d$ (neuro-imaging). Other problems that can be posed in this format can be found in [Tibshirani and Taylor, 2011].

Chapter 3

Convex penalties

3.1 Introduction

Regularization techniques that encourage sparsity in parameter estimation have gained increasing popularity recently. The most widely used example is *lasso* [Tibshirani, 1996], where the loss function $f(\cdot)$ is penalized by the ℓ_1-norm of the unknown coefficients $\beta \in \mathbb{R}^p$, to form a modified objective function,

$$\mathcal{L}(\beta) = f(\beta) + \lambda \|\beta\|_1, \quad \lambda > 0, \tag{3.1}$$

in order to shrink irrelevant coefficients to zero. Many efficient algorithms have been proposed to solve this problem, including [Fu, 1998, Daubechies et al., 2004, Efron et al., 2004] and [Friedman et al., 2007]. Some of them are capable of handling massive data sets with tens of thousands of variables and observations.

For many practical applications, physical constraints and domain knowledge may mandate additional structural constraints on the parameters. For example, in cancer research, it may be important to consider groups of interacting genes in each pathway rather than individual genes. In image analysis, it is natural to regulate the differences between neighboring pixels in order to achieve smoothness and reduce noise. In light of these popular demands, a variety of structured penalties have been proposed to incorporate prior information regarding model parameters. One of the most important structural penalties is the *fused lasso* proposed in [Tibshirani et al., 2005]. It utilizes the natural ordering of input variables to achieve parsimonious parameter estimation on neighboring coefficients. Chen et al. [2010] developed the *graph induced fused lasso* that penalizes differences between coefficients associated with nodes in a graph that are connected. Beck and Teboulle [2009] proposed the *total variation penalty* for image denoising and deblurring, in a similar fashion to the two-dimensional fused lasso. Similar penalty functions have been successfully applied to several neuroimaging studies [Michel et al., 2011, Grosenick et al., 2011, 2013]. More recently, Zhang et al. [2012] applied a generalized version of fused lasso to reconstruct gene copy number variant regions. A general structural lasso framework was proposed in [Tibshirani and Taylor, 2011], with the following form:

$$\mathcal{L}(\beta) = f(\beta) + \lambda\|R\beta\|_1, \quad \lambda > 0, \tag{3.2}$$

where R is an $m \times p$ matrix that defines the structural constraints one wants to impose on the coefficients. Many regularization problems, including high dimensional fused lasso and graph induced fused lasso, can be cast in this framework.

When the structural matrix R is relatively simple, as in the original lasso case with $R = I$, traditional path algorithms and coordinate descent techniques can be used to solve the optimization problem efficiently [Friedman et al., 2007]. For more complex structural regularization, these methods cannot be directly applied. One of the key difficulties is the non-separability of the nonsmooth penalty function. Coordinate descent methods fail to converge under this circumstances [Tseng, 2001]. Generic solvers, such as interior point methods, can sometimes be used; unfortunately they become increasingly inefficient for large size problems, particularly when the design matrix is ill-conditioned [Chen et al., 2011].

In the past two years, many efforts have been devoted to developing efficient optimization techniques for solving regularization problems using structured penalties. Liu et al. [2010a] and Ye and Xie [2011] developed a first-order and a split Bregman scheme, respectively, for solving similar class of problems. In many practical studies, the convergence of these two methods can not be guaranteed. Chen et al. [2011] proposed a modified proximal technique for the general structurally penalized problems. It is based on a first order approximation of the nonsmooth penalty function, which can become unstable when dimension is high. Meanwhile, several path algorithms have also been proposed to compute the whole regularization path for the general fused lasso problem. Hoefling [2010] developed a path algorithm for solving (4.1) when the matrix X^TX is nonsingular. This technique is not applicable to cases with large dimension of β and small number of observations, such as gene expression and brain imaging analysis. Tibshirani and Taylor [2011] extended the path algorithm to include all design matrices X, by computing the regularization path of the dual problem. Although fairly general, this version of the path algorithm does not scale well with data dimension, as the knots of the piecewise linear solution path become very dense. Many of the proposed approaches are versions of the *operator splitting methods* or their dual versions, *alternating direction methods* (see, e.g., Boyd et al. [2010], Combettes and Pesquet [2010], and the references therein). Although fairly general and universal, they frequently suffer from slow tail convergence (see [He and Yuan, 2011] and the references therein).

Thus, a need arises to develop a general approach that can solve large scale structured regularization problem efficiently. For such an approach to be successful in practice, it should guarantee to converge at a fast rate, be able to handle massive data sets, and should not rely on approximating the penalty function. In this dissertation, we propose a framework based on the alternating linearization algorithm of [Kiwiel et al., 1999], that satisfies all these requirements.

We consider the following generalization of (4.1):

$$\mathcal{L}(\beta) = f(\beta) + \lambda \|R\beta\|_{\Diamond}, \quad \lambda > 0, \tag{3.3}$$

where $\|\cdot\|_{\Diamond}$ is a norm in $\mathbb{R}^m$. Our considerations and techniques will apply to several possible choices of this norm, in particular, to the ℓ_1 norm $\|\cdot\|_1$, and to the total variation norm $\|\cdot\|_{\text{TV}}$ used in image processing.

Formally, we write the objective function as a sum of two convex functions,

$$\mathcal{L}(\beta) = f(\beta) + h(\beta), \tag{3.4}$$

where $f(\beta)$ is a loss function, which is assumed to be convex with respect to β, and $h(\cdot)$ is a convex penalty function. Any of the functions (or both) may be nonsmooth, but an essential requirement of our framework is that each of them can be easily minimized with respect to β, when augmented by a linear-quadratic term $\sum_{i=1}^{p} \left(s_i\beta_i + d_i\beta_i^2 \right)$, with some vectors $s, d \in \mathbb{R}^p$, $d > 0$. Our method bears resemblance to operator splitting and alternating direction approaches, but differs from them in the fact that it is *monotonic* with respect to the values of (3.5). We discuss these relations and differences later in section **??**, but roughly speaking, a special test applied at every iteration of the method decides which of the operator splitting iterations is the most beneficial one.

In our applications, we focus on the quadratic loss function $f(\cdot)$ and the penalty function in the form of generalized lasso (3.3), as the most important case, where comparison with other approaches is available. This case satisfies the requirement specified above, and allows for substantial specialization and acceleration of the general framework of alternating linearization. In fact, it will be clear from our presentation that any convex loss function $f(\cdot)$ can be handled in exactly the same way.

An important feature of our approach is that problems with the identity design matrix are solved exactly in one iteration, even for very large dimension.

The remainder of the chapter is organized as follows. In Section 3.3, we introduce the alternating linearization method and we discuss its relations to other approaches. Section 3.4 briefly discusses the application to lasso problems. In section 3.5 we describe the application to generalized lasso problems. Section 3.6 presents simulation results and real data examples, which illustrate the efficacy, accuracy, and scalability of the alternating linearization method. Concluding remarks are presented in section 4.4. The appendix contains details about the algorithms used to solve the subproblems of the alternating linearization method.

3.2 Optimality condition

In this section, we consider the optimality condition of the problem:

$$\min_{\beta} \mathcal{L}(\beta) = \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|R\beta\|_1, \quad \lambda > 0.$$

The analysis can be easily extended to the case where the loss function $f(\beta)$ is the log-likehood function. In addition, a more general formulation of the problem above is:

$$\min_{\beta} \mathcal{L}(\beta) = \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|R\beta\|_\Diamond, \quad \lambda > 0.$$

where $\|\cdot\|_\Diamond$ is any norm $\in \mathcal{R}^p$. To construct the optimality condition of this problem, we will need the definition of subgradient and some elements of subdifferential calculus [Ruszczyński, 2006].

Definition 1 *Let $f : \mathcal{R}^n \to \mathcal{R}$ be a proper convex function and let $x \in dom f$. A vector $g \in \mathcal{R}^n$: $f(y) \geq f(x) + \langle g, y - x\rangle \forall y \in \mathcal{R}^n$ is called a subgradient of f at x. $\partial f(x)$ is the set of subgradients of f at x.*

Lemma 1 *Let $f : \mathcal{R}^n \to \mathcal{R}$ be a convex function, $A \in \mathcal{R}^{m\times n}$, and $h(x) = f(Ax)$, then $\partial h(x) = A^T \partial f(Ax), \forall x$.*

Lemma 2 *Let $\|\cdot\|_\Diamond$ is any norm $\in \mathcal{R}^n$. Define the dual norm as:*

$$\|g\|_* = \sup_{d\neq 0} \frac{\langle g, d\rangle}{\|d\|_\Diamond} = \sup_{\|d\|_\Diamond = 1} \langle g, d\rangle.$$

then

$$\partial\|x\|_\Diamond = \{g \in \mathcal{R}^n : \|g\|_* \leq 1, \langle g, x\rangle = \|x\|_\Diamond\}.$$

When the loss function $f(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$ then we have:

$$\partial\mathcal{L}(\beta) = X^TX\beta - X^Ty + \partial h(\beta).$$

where $h(\beta) = \lambda\|R\beta\|_\Diamond$. In particular, if $\|\cdot\|_\Diamond = \|\cdot\|_1$, then the dual norm $\|g\|_* = \|g\|_\infty$, we have:

$$\partial h(\beta) = \{\lambda R^T g : \|g\|_\infty = 1, \langle g, R\beta\rangle = \|R\beta\|_1\}.$$

when $R\beta \neq 0$. At the optimal β, zero is an element of the subdifferential, we have:

$$0 = X^TX\beta - X^Ty + \lambda R^T\hat{g}.$$

for a $\hat{g}$ that satisfies the conditions above . Let $z = X^Ty - X^TX\beta$, then we have:

$$\begin{cases} z = & \lambda R^T\hat{g}. \\ \|g\|_\infty = & 1. \\ \langle\hat{g}, R\beta\rangle = & \|R\beta\|_1. \end{cases}$$

Let R_j be the jth row of matrix R, denote:

$$\begin{aligned} J^+ = &\ \{j : R_j\beta > 0\}. \\ J^- = &\ \{j : R_j\beta < 0\}. \\ J^0 = &\ \{j : R_j\beta = 0\}. \end{aligned}$$

From the last two conditions we get:

$$\hat{g}_j = \begin{cases} 1 & \text{if } j \in J^+. \\ -1 & \text{if } j \in J^-. \\ p_j & \text{if } j \in J^0. \end{cases}$$

for $p_j \in [-1, 1]$.

3.3 Alternating linearization method

3.3.1 Outline of the method

In this section, we describe the alternating linearization (ALIN) approach to minimize:

$$\mathcal{L}(\beta) = f(\beta) + h(\beta), \tag{3.5}$$

It is an iterative method, which generates a sequence of approximations $\{\hat{\beta}^k\}$ converging to a solution of the original problem (3.5), and two auxiliary sequences: $\{\tilde{\beta}_h^k\}$ and $\{\tilde{\beta}_f^k\}$, where k is the iteration number. Each iteration of the ALIN algorithm consists of solving two subproblems: the *h-subproblem* and the *f-subproblem*, and of an *update step*, applied after any of the subproblems, or after each of them. At the beginning we set $\tilde{\beta}_f^0 = \hat{\beta}^0$, where $\hat{\beta}^0$ is the starting point of the method. In the description below, we suppress the superscript k denoting the iteration number, to simplify notation.

The h-subproblem

We linearize $f(\cdot)$ at $\tilde{\beta}_f$, and approximate it by the function

$$\tilde{f}(\beta) = f(\tilde{\beta}_f) + s_f^T(\beta - \tilde{\beta}_f).$$

If $f(\cdot)$ is differentiable, then $s_f = \nabla f(\tilde{\beta}_f)$; for a general convex $f(\cdot)$, we select a subgradient $s_f \in \partial f(\tilde{\beta}_f)$. In the first iteration, this may be an arbitrary subgradient; at later iterations special selection rules apply, as described in (3.9) below.

The approximation is used in the optimization problem

$$\min_{\beta} \; \tilde{f}(\beta) + h(\beta) + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2, \tag{3.6}$$

in which the last term is defined as follows:

$$\|\beta - \hat{\beta}\|_D^2 = (\beta - \hat{\beta})^T D(\beta - \hat{\beta}),$$

with a diagonal matrix $D = \text{diag}\{d_j,\, j = 1, \dots, p\}$, $d_j > 0$, $j = 1, \dots, p$. The solution of the h-subproblem (3.6) is denoted by $\tilde{\beta}_h$.

We complete this stage by calculating the subgradient of $h(\cdot)$ at $\tilde{\beta}_h$, which features in the optimality condition for the minimum in (3.6):

$$0 \in s_f + \partial h(\tilde{\beta}_h) + D(\tilde{\beta}_h - \hat{\beta}).$$

Elementary calculation yields the right subgradient $s_h \in \partial h(\tilde{\beta}_h)$:

$$s_h = -s_f - D(\tilde{\beta}_h - \hat{\beta}). \tag{3.7}$$

The f-subproblem

Using the subgradient s_h we construct a linear minorant of the penalty function $h(\cdot)$ as follows:

$$\tilde{h}(\beta) = h(\tilde{\beta}_h) + s_h^T(\beta - \tilde{\beta}_h).$$

This approximation is employed in the optimization problem

$$\min_{\beta} \ f(\beta) + \tilde{h}(\beta) + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2. \tag{3.8}$$

The optimal solution of this problem is denoted by $\tilde{\beta}_f$. It will be used in the next iteration as the point at which the new linearization of $f(\cdot)$ will be constructed. The next subgradient of $f(\cdot)$ to be used in the h-subproblem will be

$$s_f = -s_h - D(\tilde{\beta}_f - \hat{\beta}). \tag{3.9}$$

The update step

The update step can be applied after any of the subproblems, or after both of them. It changes the current best approximation of the solution $\hat{\beta}$, if certain improvement conditions are satisfied. It uses a parameter $\gamma \in (0, 1)$. We describe it here for the case of applying the update step after the f-subproblem; analogous operations are carried out if the update step is applied after the h-subproblem.

At the beginning of the update step the stopping criterion is verified. If

$$f(\tilde{\beta}_f) + \tilde{h}(\tilde{\beta}_f) \geq f(\hat{\beta}) + h(\hat{\beta}) - \varepsilon, \tag{3.10}$$

the algorithm terminates. Here $\varepsilon > 0$ is the stopping test parameter.

If the the stopping test is not satisfied, we check the inequality

$$f(\tilde{\beta}_f) + h(\tilde{\beta}_f) \le (1-\gamma)\big[f(\hat{\beta}) + h(\hat{\beta})\big] + \gamma\big[f(\tilde{\beta}_f) + \tilde{h}(\tilde{\beta}_f)\big]. \qquad (3.11)$$

If it is satisfied, then we update $\hat{\beta} \leftarrow \tilde{\beta}_f$; otherwise $\hat{\beta}$ remains unchanged.

If the update step is applied after the h-subproblem, we use $\tilde{\beta}_h$ instead if $\tilde{\beta}_f$ in the inequalities (3.10) and (3.11).

The update step is a crucial component of the alternating linearization algorithm; it guarantees that the sequence $\{\mathcal{L}(\hat{\beta}^k)\}$ is monotonic, and it stabilizes the entire algorithm (see the remarks at the end of section 3.6.2).

3.3.2 Convergence

Convergence properties of the alternating linearization method follow from the general theory developed in [Kiwiel et al., 1999]. Indeed, after the change of variables $\xi = D^{1/2}\beta$ we see that the method is identical to Algorithm 3.1 of [Kiwiel et al., 1999], with $\rho_k = 1$. The following statement is a direct consequence of [Kiwiel et al., 1999, Theorem 4.8].

Theorem 3 *Suppose that the set of minima of the function* (3.5) *is nonempty. Then the sequence* $\{\hat{\beta}^k\}$ *generated by the algorithm is convergent to a minimum point* β^* *of the function* (3.5). *Moreover, every accumulation point* (s_f^*, s_h^*) *of the sequence* $\{(s_f^k, s_h^k)\}$ *satisfies the relations:* $s_f^* \in \partial f(\beta^*)$, $s_h^* \in \partial h(\beta^*)$, *and* $s_f^* + s_h^* = 0$.

For structured regularization problems the assumption of the theorem is satisfied, because both the loss function $f(\cdot)$ and the regularizing function $h(\cdot)$ are bounded from below, and one of the purposes of the regularization term is to make the set of minima of the function $\mathcal{L}(\cdot)$ nonempty and bounded.

3.4 Application to lasso regression

First, we demonstrate the alternating linearization algorithm (ALIN) on the classical lasso regression problem. Due to the separable nature of the penalty function, very efficient coordinate descent methods are applicable to this problem as well [Tseng, 2001], but we wish to illustrate our approach on the simplest case first.

In the lasso regression problem we have

$$f(\beta) = \tfrac{1}{2}\|y - X\beta\|_2^2, \qquad h(\beta) = \lambda\|\beta\|_1,$$

where X is the $n \times p$ design matrix, $y \in \mathbb{R}^n$ is the vector of response variables, $\beta \in \mathbb{R}^p$ is the vector of regression coefficients, and $\lambda > 0$ is a parameter of the model.

We found it essential to use $D = \text{diag}(X^T X)$, that is, $d_j = X_j^T X_j$, $j = 1, \ldots, p$. This choice is related to the *diagonal quadratic approximation* of the function $f(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$, which was employed (for similar objectives in the context of augmented Lagrangian minimization) by Ruszczyński [1995]. Indeed, in the h-subproblem in the formula (3.12) below, the quadratic regularization term is a quadratic form built on the diagonal of the Hessian of $f(\cdot)$.

The h-subproblem

The problem (3.6), after skipping constants, simplifies to the following form

$$\min_{\beta}\ s_f^T \beta + \lambda\|\beta\|_1 + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2, \tag{3.12}$$

with $s_f = X^T(X\tilde{\beta}_f - y)$. Writing $\tau_j = \hat{\beta} - \tilde{s}_{fj}/d_j$, we obtain the following closed form solutions of (3.12), which can be calculated component-wise:

$$\tilde{\beta}_{hj} = \text{sgn}(\tau_j)\max\left(0, |\tau_j| - \frac{\lambda}{d_j}\right), \quad j = 1, \ldots, p. \tag{3.13}$$

The subgradient s_h of $h(\cdot)$ at $\tilde{\beta}_h$ is calculated by (3.7).

The f-subproblem

The problem (3.8), after skipping constants, simplifies to the unconstrained quadratic programming problem

$$\min_{\beta}\ s_h^T \beta + \tfrac{1}{2}\|y - X\beta\|_2^2 + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2. \tag{3.14}$$

Its solution can be obtained by solving the following symmetric linear system in $\delta = \beta - \hat{\beta}$:

$$(X^T X + D)\delta = X^T(y - X\hat{\beta}) - s_h. \tag{3.15}$$

This system can be efficiently solved by the preconditioned conjugate gradient method (see, e.g., [Golub and Van Loan, 1996]), with the diagonal preconditioner $2D = 2\,\mathrm{diag}(X^TX)$. Its application does not require the explicit form of the matrix X^TX; only matrix-vector multiplications with X and X^T are employed, and they can be implemented with sparse data structures.

3.5 Application to general structured regularization problems

In the following we apply the alternating linearization algorithm to solve more general structured regularization problems including the generalized Lasso (3.3). Here we assume the least square loss, as in the previous subsection. The objective function can be written as follows:

$$\mathcal{L}(\beta) = f(\beta) + h(\beta) = \tfrac{1}{2}\|y - X\beta\|_2^2 + \lambda\|R\beta\|_\Diamond. \tag{3.16}$$

For example, for the one-dimensional fused lasso, R is the following $(p-1)\times p$ matrix:

$$R = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots\dots} \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix},$$

and the norm $\|\cdot\|_\Diamond$ is the ℓ_1-norm $\|\cdot\|_1$, but our derivations are valid for any form of R, and any norm $\|\cdot\|_\Diamond$.

The h-subproblem

The h-subproblem can be equivalently formulated as follows:

$$\min_{\beta,z}\; s_f^T\beta + \lambda\|z\|_\Diamond + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2 \quad \text{subject to} \quad R\beta = z. \tag{3.17}$$

Owing to the use of $D = \mathrm{diag}(X^TX)$, and with $s_f = X^T(X\hat{\beta} - y)$, it is a quite accurate approximation of the original problem, especially for sparse X [Ruszczyński, 1995].

The Lagrangian of problem (3.17) has the form

$$L(\beta, z, \mu) = s_f^T\beta + \lambda\|z\|_\Diamond + \mu^T(R\beta - z) + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2,$$

where μ is the dual variable. Consider the dual norm $\|\cdot\|_*$, defined as follows:

$$\|\mu\|_* = \max_{\|z\|_\Diamond \le 1} \mu^T z, \qquad \|z\|_\Diamond = \max_{\|\mu\|_* \le 1} \mu^T z. \tag{3.18}$$

We see that the minimum of the Lagrangian with respect to z is finite if and only if $\|\mu\|_* \le \lambda$ [Ruszczyński, 2006, Example 2.94]. Under this condition, the minimum value of the z-terms is zero and we can eliminate them from the Lagrangian. We arrive to its reduced form,

$$\hat{L}(\beta, \mu) = s_f^T \beta + \mu^T R\beta + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2. \tag{3.19}$$

To calculate the dual function, we minimize $\hat{L}(\beta, \mu)$ over $\beta \in \mathbb{R}^p$. After elementary calculations, we obtain the solution

$$\tilde{\beta}_h = \hat{\beta} - D^{-1}(s_f + R^T \mu). \tag{3.20}$$

Substituting it back to (3.19), we arrive to the following dual problem:

$$\max_{\mu} \; -\tfrac{1}{2}\mu^T R D^{-1} R^T \mu + \mu^T R(\hat{\beta} - D^{-1} s_f) \quad \text{subject to} \quad \|\mu\|_* \le \lambda. \tag{3.21}$$

This is a norm-constrained optimization problem. Its objective function is quadratic, and the specific form of the constraints depends on the norm $\|\cdot\|_\Diamond$ used in the regularizing term of (3.3).

The case of the ℓ_1-norm

If the norm $\|\cdot\|_\Diamond$ is the ℓ_1-norm $\|\cdot\|_1$, then the dual norm is the ℓ_∞-norm:

$$\|\mu\|_* = \|\mu\|_\infty = \max_{1 \le j \le m} |\mu_j|.$$

In this case (3.21) becomes a box-constrained quadratic programming problem, for which many efficient algorithms are available. One possibility is the active-set box-constrained preconditioned conjugate gradient algorithm with spectral projected gradients, as described in [Birgin and Martínez, 2002, Friedlander and Martínez, 1994]. It should be stressed that its application does not require the explicit form of the matrix $RD^{-1}R^T$; only matrix-vector multiplications with R and R^T are employed, and they can be implemented with sparse data structures.

An even better possibility, due to the separable form of the constraints, is coordinate-wise optimization (see, e.g., [Ruszczyński, 2006, Sec. 5.8.2]) in the dual problem (3.21). In our experiments, the dual coordinate-wise optimization method strictly outperforms the box-constrained algorithm, in terms of the solution time.

The solution $\tilde{\mu}$ of the dual problem can be substituted into (3.20) to obtain the primal solution.

The case of a sum of ℓ_2-norms

Another important case arises when the vector $z = R\beta$ is split into I subvectors $z^1, z^2, \ldots, z^I$, and

$$\|z\|_\Diamond = \sum_{i=1}^{I} \|z^i\|_2. \tag{3.22}$$

A special case of it is the *total variation norm* discussed in section 3.6.4.

We can directly verify that the dual norm has the following form:

$$\|\mu\|_* = \max_{1\le i\le I} \|\mu^i\|_2.$$

It follows that problem (3.21) is a block-quadratically constrained quadratic optimization problem:

$$\begin{aligned} &\max_{\mu} \; -\tfrac{1}{2}\mu^T R D^{-1} R^T \mu + \mu^T R(\hat{\beta} - D^{-1} s_f) \\ &\text{s. t. } \|\mu^i\|_2^2 \le \lambda^2, \quad i = 1, \ldots, I. \end{aligned} \tag{3.23}$$

This problem can be very efficiently solved by a cyclical block-wise optimization with respect to the subvectors $\mu^1, \mu^2, \ldots, \mu^I$. At each iteration of the method, optimization with respect to the corresponding subvector μ^j is performed, subject to one constraint $\|\mu^j\|_2^2 \le \lambda^2$. The other subvectors, μ^i, $i \neq j$ are kept fixed on their last values. After that, j is incremented (if $j < I$) or reset to 1 (if $j = I$), and the iteration continues. The method stops when no significant improvements over I steps can be observed. The dual block optimization method performs well in the applications we are interested in.

Again, the solution $\tilde{\mu}$ of the dual problem is substituted into (3.20) to obtain the primal solution.

The f-subproblem

We obtain the update $\tilde{\beta}_f$ by solving the linear equation system (3.15), exactly as in the lasso case.

The special case of $X = I$

If the design matrix $X = I$ in (3.16), then our method solves the problem in one iteration, when started from $\hat{\beta} = y$. Indeed, in this case we have $s_f = 0$, $D = I$, and the first h-subproblem becomes equivalent to the original problem (3.16):

$$\min_{\beta,z} \ \lambda\|z\|_\Diamond + \tfrac{1}{2}\|\beta - y\|_2^2 \quad \text{subject to} \quad R\beta = z. \tag{3.24}$$

The dual problem (3.21) simplifies as follows:

$$\max_{\mu} \ -\tfrac{1}{2}\mu^T RR^T\mu + \mu^T Ry \quad \text{subject to} \quad \|\mu\|_* \le \lambda. \tag{3.25}$$

It can be solved by the same block-wise optimization method, as in the general case. The optimal primal solution is then $\tilde{\beta}_h = y - R^T\mu$.

3.6 Numerical experiments

In this section, we present results on a number of simulations and real data studies involving a variety of non-differentiable penalty functions. We compare the alternating linearization algorithm (ALIN) with competing approaches in terms of iteration steps, computation time, and estimation accuracy. All these studies are performed on an AMD 2.6GHZ, 4GB RAM computer using MATLAB.

3.6.1 ℓ_1 regularization

In this section, we compare ALIN with some competing methods for solving the ℓ_1 regularization problem:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1, \quad \lambda > 0. \tag{3.26}$$

The methods that we are comparing with are: SpaRSA, a type of iterative thresholding method, considered the best method among its variations; FISTA, a variation of the Nesterov method, considered to be state-of-the-art among the first order methods; and SPG, a spectral gradient method. We follow the procedure described by [Wright et al., 2009] to generate a data set for comparisons. The elements of the matrix X are generated independently using a Gaussian distribution with mean zero and variance 10^{-2}. The dimension of X is $n = 2^{10}$ by $p = 2^{12}$, $p = 2^{13}$, and $p = 2^{14}$. The true signal, β, is a vector with 160 randomly placed ± 1 spikes and zeros elsewhere. The dependent variables are $y = X\beta + \epsilon$, where ϵ is Gaussian noise with variance 10^{-4}.

To make a fair comparison between the methods, we run FISTA on each instance of the problem. FISTA is set to run to "tol" $= 10^{-5}$ or $5,000$ iterations, whichever comes first. Then ALIN, SpaRSA, and SPG are set to run until the objective function values obtained are as good as that of FISTA. We set a parameter $\tau = 0.1\|X^T y\|_\infty$ and chose values of $\lambda =\tau, 10^{-1}\tau$, $5 \times 10^{-2}\tau$, $10^{-2}\tau$, and $10^{-3}\tau$. We allow SpaRSA to run its *monotone* and *continuation* feature. Continuation is a special feature of SpaRSA for cases when the parameter λ is small. With this feature, SpaRSA computes the solutions for bigger values of λ and uses them to find solutions for smaller values of λ. We did not let SpaRSA use its special feature *de-bias* since it involves removing zero coefficients to reduce the size of the data set. This feature makes it unfair for the other competing methods. In Table 3.1, we report the average time elapsed (in seconds) and the standard deviation after 20 runs.

Table 3.1: Average run time (in CPU seconds) and standard deviation (in parenthesis) comparison for combinations of dimension p and tuning parameter λ.

			$p = 2^{12}$		$p = 2^{13}$		$p = 2^{14}$
$\lambda = \tau$	*ALIN*	17.99	(10.68)	36.60	(15.08)	105.14	(40.88)
	FISTA	8.58	(4.00)	18.63	(9.35)	58.73	(42.01)
	SPARSA	8.18	(2.34)	8.18	(3.80)	34.23	(49.55)
	SPG	160.72	(27.48)	160.72	(47.82)	404.59	(79.62)
$\lambda = 10^{-1}\tau$	*ALIN*	9.35	(2.99)	34.01	(15.18)	74.06	(34.27)
	FISTA	16.91	(4.57)	36.43	(16.66)	131.91	(33.94)
	SPARSA	20.55	(10.08)	36.81	(19.29)	169.88	(73.01)
	SPG	136.26	(23.30)	186.94	(46.56)	460.71	(38.93)
$\lambda = 5 \times 10^{-2}\tau$	*ALIN*	6.30	(2.73)	21.83	(10.17)	65.79	(39.49)
	FISTA	18.88	(2.95)	48.37	(15.77)	158.73	(21.86)
	SPARSA	35.56	(11.32)	74.66	(24.49)	234.75	(64.67)
	SPG	140.00	(23.15)	190.43	(45.48)	473.78	(6.41)
$\lambda = 10^{-2}\tau$	*ALIN*	4.58	(2.05)	16.96	(10.99)	28.88	(16.03)
	FISTA	18.85	(3.04)	46.58	(14.91)	169.94	(19.76)
	SPARSA	33.52	(16.10)	76.69	(28.18)	214.85	(124.56)
	SPG	140.63	(22.43)	196.54	(43.96)	483.71	(4.51)
$\lambda = 10^{-3}\tau$	*ALIN*	3.68	(1.20)	6.67	(2.43)	20.16	(4.31)
	FISTA	18.88	(2.84)	45.40	(14.28)	162.85	(36.76)
	SPARSA	19.94	(12.10)	39.55	(27.45)	92.76	(102.53)
	SPG	138.73	(19.56)	201.74	(48.09)	467.91	(101.32)

We can see that the performance of ALIN is comparable to the other methods. In terms of running time, ALIN does better than all competing methods for the range of middle and small values of λ. For large values of λ, ALIN does worse than FISTA and SPARSA. For large values of λ, the solution is fairly close to the starting point 0, therefore the overhead cost of the update steps and the f-subproblem would slow down ALIN. When the value of λ reduces, the benefits of these steps become more evident, when ALIN outperforms other methods in terms of running time, by factors of two to three. We should also note that the implementation of FISTA was in C, and SpaRSA is a very efficient method specially designed for separable regularization. From our numerical studies, for medium and small values of λ, FISTA makes very small improvement over $5,000$ iterations and SPARSA has to go through many previous values of λ to reach the desired level of objective function values.

3.6.2 Fused Lasso regularization

In this experiment, we compare the ALIN algorithm with two different approaches using data sets generated from a linear regression model $y = \sum_{j=1}^{p} x_j\beta_j + \epsilon$, with pre-specified coefficients β_j, and varying dimension p. The values of x_j are drawn from the normal distribution with zero mean and unit variance. The noise ϵ is generated from the normal distribution with zero mean and variance equal to 0.01. Among the coefficients β_j, 10% equal 1, 20% equal 2, and the rest are zero. For instance, with $p = 100$, we may have

$$\beta_j = \begin{cases} 1 & \text{for } j = 11, 12, \dots, 20, \\ 2 & \text{for } j = 21, \dots, 40, \\ 0 & \text{otherwise.} \end{cases}$$

Table 3.2 reports the run times of ALIN and three competing algorithms: the generic quadratic programming solver (SQOPT), an implementation of Nesterov's method, SLEP, of [Liu et al., 2011, Nesterov, 2007], and the split Bregman method of [**?**] (BREGMAN). We fix the sample size to n=1000 and vary the dimension p of the problem from 1000 to 50000. Each method is repeated 10 times over different values of turning parameter λ and the average running time is reported. SLEP's stopping parameter "tol" was set to 10^{-5}. BREGMAN also uses stopping parameter "tol"= 10^{-5}. We stop ALIN runs when the objective function value attained is as good as the last value attained by SLEP. Judging from these results, ALIN clearly outperforms the other methods in terms of speed for most cases. The relative improvements on run time can be as much as 8 fold, depending on the experimental setting, and become more

Table 3.2: Run time (in CPU seconds) comparison for combinations of dimension p and tuning parameter λ

		$p = 1000$	$p = 5000$	p=10,000	p=20,000	p=50,000
$\lambda = 10^{-4}$	*SQOPT*	10	1076	NA	NA	NA
	SLEP	3	1	2	3	5
	ALIN	6	0.5	1	2	5
	BREGMAN	111	24	25	38	52
$\lambda = 10^{-3}$	*SQOPT*	9	1025	NA	NA	NA
	SLEP	4	99	921	2661	4150
	ALIN	9	31	248	665	1278
	BREGMAN	114	21	23	30	49
$\lambda = 10^{-2}$	*SQOPT*	11	1019	NA	NA	NA
	SLEP	2	109	400	1571	6441
	ALIN	6	17	77	313	815
	BREGMAN	114	23	57	96	106
$\lambda = 0.1$	*SQOPT*	11	956	NA	NA	NA
	SLEP	0.4	42	145	508	1358
	ALIN	4	22	80	280	387
	BREGMAN	103	471	879	2133	3633
$\lambda = 0.2$	*SQOPT*	11	1015	NA	NA	NA
	SLEP	1	47	121	387	2251
	ALIN	4	52	100	284	1360
	BREGMAN	87	492	833	1543	3541
$\lambda = 0.5$	*SQOPT*	11	1029	NA	NA	NA
	SLEP	0.9	36	111	386	1584
	ALIN	3	47	144	371	1730
	BREGMAN	60	503	924	1633	3543

significant, when the data dimension grows. This is particularly significant in view of the fact that ALIN was implemented as a MATLAB code, as opposed to the executables in the other cases. Figure 4.2 presents the solutions obtained by ALIN and SLEP compared to the known parameters. Both ALIN and SLEP achieve results that are very close to the original β, and the objective function values are very similar. BREGMAN performs well when the number of parameters p is not significantly larger than the number of observations n. When $p \gg n$, although BREGMAN has a very good running time, it tends to terminate early and does not provide accurate results. In Fig 4.2, we can see that the solution obtained by BREGMAN is not as good as those of SLEP and ALIN.

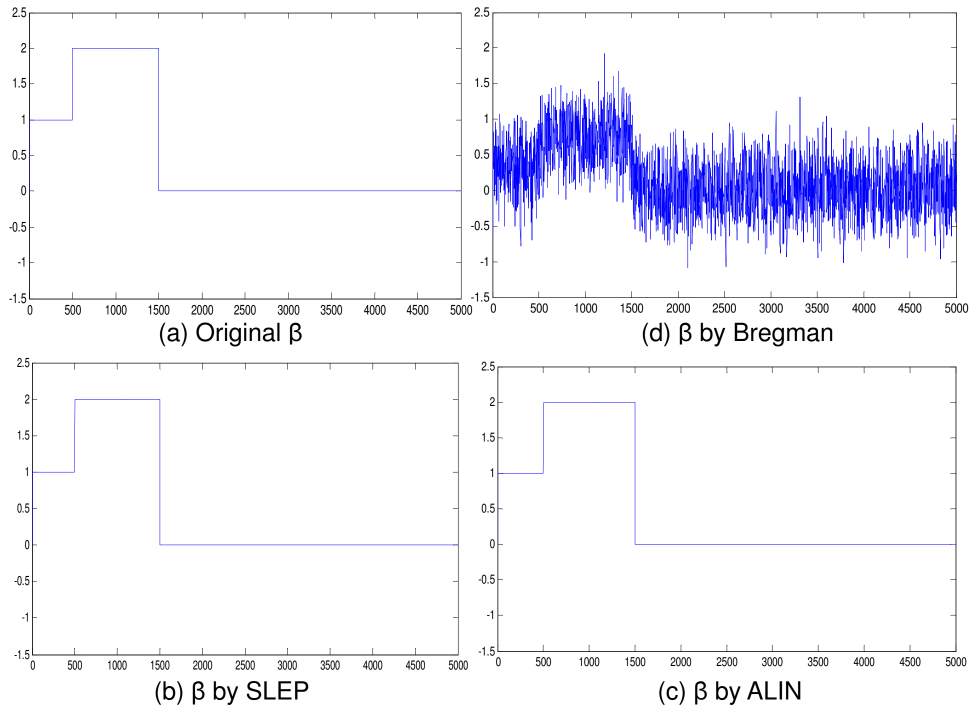
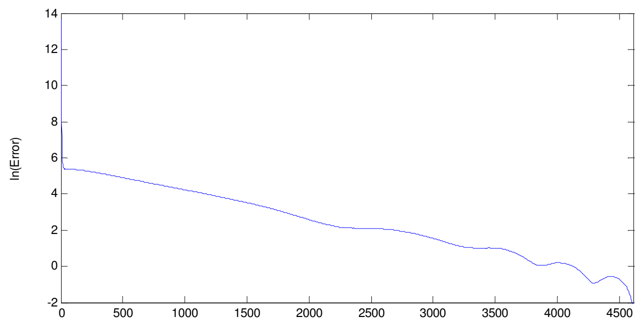


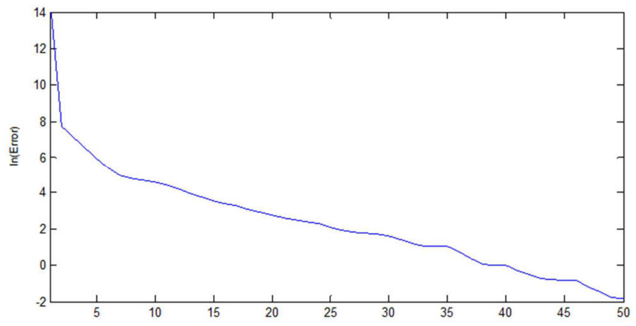
Figure 3.1: Results of using fused lasso penalty on a simulated data set with $n = 1000, p = 5000$, and $\lambda = 0.2$. Plots (a), (b), (c), and (d) correspond to the original β, results from BREGMAN, SLEP, and ALIN, respectively.

We also investigated how our method approaches the optimal objective function value compared to other methods. Using the above simulated data set with $n = 1000$, $p = 5000$, and $\lambda = 0.1$, we run ALIN and SLEP to convergence. At each iteration, we calculated the difference between the optimal value $\mathcal{L}^*$ (obtained by SQOPT) and the current function value for each method. Figure 3.2 displays (in a logarithmic scale) the progress of both methods. It is clear that ALIN achieves the same accuracy as SLEP in a much smaller number of iterations. Furthermore, the convergence of ALIN is monotonic, whereas that of SLEP is not.

In Figure 3.3 we provide the dependence of the running time of ALIN on the dimensions of the problem, to illustrate its scalability. The efficiency of the method is due mainly to its good convergence properties, but also to the efficiency of the preconditioned conjugate gradient method for solving the subproblems. It employs sparse data structures and converges rapidly. Usually, between 10 and 20 iterations of the conjugate gradient method are sufficient to find the



(a) SLEP



(b) ALIN

Figure 3.2: Simulated data set with $n = 1000$, $p = 5000$, λ=0.1. Plots (a) and (b): $\ln(\text{Error})$ versus iteration number of SLEP and ALIN, respectively. Error is defined as the difference between the optimal value $\mathcal{L}^*$ (obtained by SQOPT) and those obtained by SLEP and ALIN respectively.

solution of a subproblem.

The update test (3.11) is an essential element of the ALIN method. For example, in a case with $n = 1000$, $p = 5000$, and $\lambda = 0.1$, the update of $\hat{\beta}$ occurred in about 80% of the total of 70 iterations, while other iterations consisted only of improving alternating linearizations. If we allow updates of $\hat{\beta}$ at every step, the algorithm takes more than 5000 iterations to converge in this case. Similar behavior was observed in all other cases. These observations clearly demonstrate the difference between the alternating linearization method and the operator splitting methods.

3.6.3 CGH data example

In this study we present the results on analyzing the CGH data using fused lasso penalty. CGH is a technique for measuring DNA copy numbers of selected genes on the genome. The CGH array experiments return the log ratio between the number of DNA copies of the gene in the tumor cells and the number of DNA copies in the reference cells. A value greater than zero indicates a

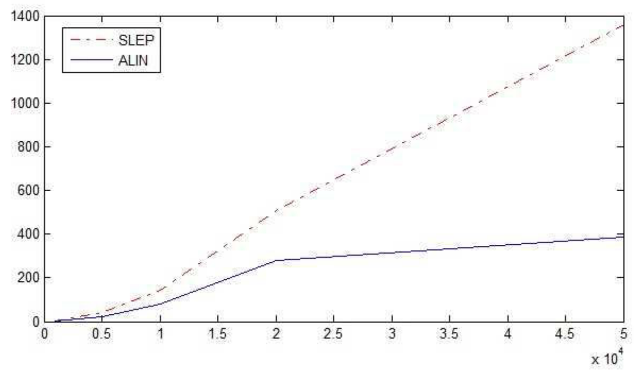


Figure 3.3: Running time of SLEP and ALIN as dimension changes.

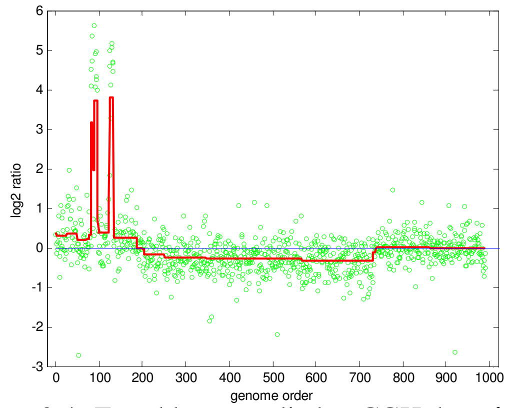


Figure 3.4: Fused lasso applied to CGH data, $\lambda = 3$.

possible gain, while a value less than zero suggests possible losses. Tibshirani and Wang [2008] applied the fused lasso signal approximator for detecting such copy number variations. This is a simple one-dimensional signal approximation problem with the design matrix X being the identity matrix. Thus the advantage of ALIN over the other three methods is not significant, due to the overhead that ALIN has during the conjugate gradient method implemented in MATLAB. Indeed the solution time of ALIN is comparable to that of Bregman and SLEP.

Figure 3.4 presents the estimation results obtained by our ALIN method. The green dots shows the original CNV number, and the red line presents the fused lasso penalized estimates.

3.6.4 Wavelet based and Total variation based image reconstruction

In image recovery literature, two classes of regularizers are well known. One is the Tikhonov type of operators, where the regularizing term is quadratic, and the other is the discrete total variation (TV) regularizer. The resulting objective function from the first type is relatively easy to minimize, but it tends to over-smooth the image, thus failing to preserve its sharpness [Wang et al., 2008]. In the following experiment, we demonstrate the effectiveness of ALIN in solving TV-based image deblurring problems, with discrete TV, as well as a comparison to the Tikhonov regularizer.

Although of similar form, higher-order fused lasso models are fundamentally different from the one-dimensional fused lasso, as the structural matrix R appearing in eq. (3.3) is not full-rank and $R^T R$ is ill-conditioned. This additional complication introduces considerable challenges in the path type algorithms [**?**], and additional computational steps need to be implemented to guarantee convergence. The ALIN algorithm does not suffer from complications due to the singularity of R, because the dual problem (3.21) is always well-defined and has a solution. Even if the solution is not unique, (3.20) is still an optimal solution of the h-subproblem, and the algorithm proceeds unaffected.

Let y be an $m \times n$ observed noisy image; one attempts to minimize the following objective function:

$$\mathcal{L}(\beta) = \tfrac{1}{2}\|y - \mathcal{A}(\beta)\|_2^2 + \lambda h(\beta), \tag{3.27}$$

where $h(\beta)$ is an image variation penalty, and $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is a linear transformation. When $\mathcal{A}$ is the identity transformation, the problem is to *denoise* the image y, but we are rather interested in a significantly more challenging problem of *deblurring*, where $\mathcal{A}$ replaces each pixel with the average of its neighbors and itself (typically, a 3 by 3 block, except for the border).

The penalty can be defined as the ℓ_1-norm of the differences between neighboring pixels (ℓ_1-*TV*),

$$h(\beta) = \sum_{i=1}^{m-1}\sum_{j=1}^{n-1} \big(|\beta_{i,j} - \beta_{i+1,j}| + |\beta_{i,j} - \beta_{i,j+1}|\big) + \sum_{i=1}^{m-1} |\beta_{i,n} - \beta_{i+1,n}| + \sum_{j=1}^{n-1} |\beta_{m,j} - \beta_{m,j+1}|, \tag{3.28}$$

or as follows (ℓ_2-*TV*):

$$h(\beta) = \sum_{i=1}^{m-1}\sum_{j=1}^{n-1} \big(|\beta_{i,j}-\beta_{i+1,j}|^2+|\beta_{i,j}-\beta_{i,j+1}|^2\big)^{1/2}+\sum_{i=1}^{m-1} |\beta_{i,n}-\beta_{i+1,n}|+\sum_{j=1}^{n-1} |\beta_{m,j}-\beta_{m,j+1}|. \tag{3.29}$$

It is clear that both cases can be cast into the general form (3.3), with the operator R representing the evaluation of the differences $\beta_{i,j} - \beta_{i+1,j}$ and $\beta_{i,j} - \beta_{i,j+1}$. The regularizing function (3.28) corresponds to the ℓ_1-norm of $R\beta$, while the function (3.29) corresponds to a norm of form

(3.22). In the latter case, we have mn blocks, each of dimension two, except for the border blocks, which are one-dimensional.

In the following experiments, we apply the ℓ_1-TV to recover noisy and blurred images to their original forms. The resulting regularization problems are rather complex. Deblurring a 256 by 256 image results in solving a very large generalized lasso problem (the matrix R has dimensions of about 262000×66000). The f-subproblem is solved using the block coordinate descent method and the h-subproblem is solved using the conjugate gradient method with "tol" $= 10^{-5}$, as discussed previously. The fact that A and R are sparse matrices makes the implementation very efficient, as demonstrated in the numerical study.

First, we *blur* the image, by replacing each pixel with the average of its neighbors and itself. This operation defines the kernel operator $\mathcal{A}$ used in the loss function $\frac{1}{2}\|y - \mathcal{A}(\beta)\|_2^2$. Then we add $N(0, 0.02)$ noise to each pixel. Clearly, for image deblurring, the design matrix is no longer the identity matrix, thus the problem is more complicated than the image denoising problem. The deblurring results on a standard example ("Lena") are shown in Figure 3.5; similar deblurring results from ALIN and FISTA are observed.

Next, we run the image deblurring on a 1 Megapixel image. We compare the result of image deblurring using the ℓ_1-TV and a quadratic Tiknonov regularization approach, which corresponds to formula (3.29) without the square root operations:

$$h(\beta) = \sum_{i=1}^{m-1}\sum_{j=1}^{n-1} |\beta_{i,j} - \beta_{i+1,j}|^2 + |\beta_{i,j} - \beta_{i,j+1}|^2 + \sum_{i=1}^{m-1} |\beta_{i,n} - \beta_{i+1,n}|^2 + \sum_{j=1}^{n-1} |\beta_{m,j} - \beta_{m,j+1}|^2. \tag{3.30}$$

The results are shown in Figure 3.6. It is seen that the ℓ_1-TV recovers a sharper image than the quadratic penalty. Deblurring with the two-dimensional fused lasso penalty yields an MSE of 7.2 with respect to the original image, while that of Tikhonov regularization is 9.3. Deblurring with the regularizer (3.28) has an almost identical effect as with (3.29).

Another method that is considered for the image deblurring problem is wavelet based deblurring. The basic setting is very similar to the previous one. However, wavelet based deblurring uses a different penalty function:

$$\min_{\beta} \frac{1}{2}\|y - \mathcal{A}(\beta)\|_2^2 + \lambda\|\mathbb{W}\beta\|_1, \quad \lambda > 0,$$

where $\mathbb{W}$ is a wavelet transformation, and $\mathcal{A}$ is the linear blurring operator. The philosophy

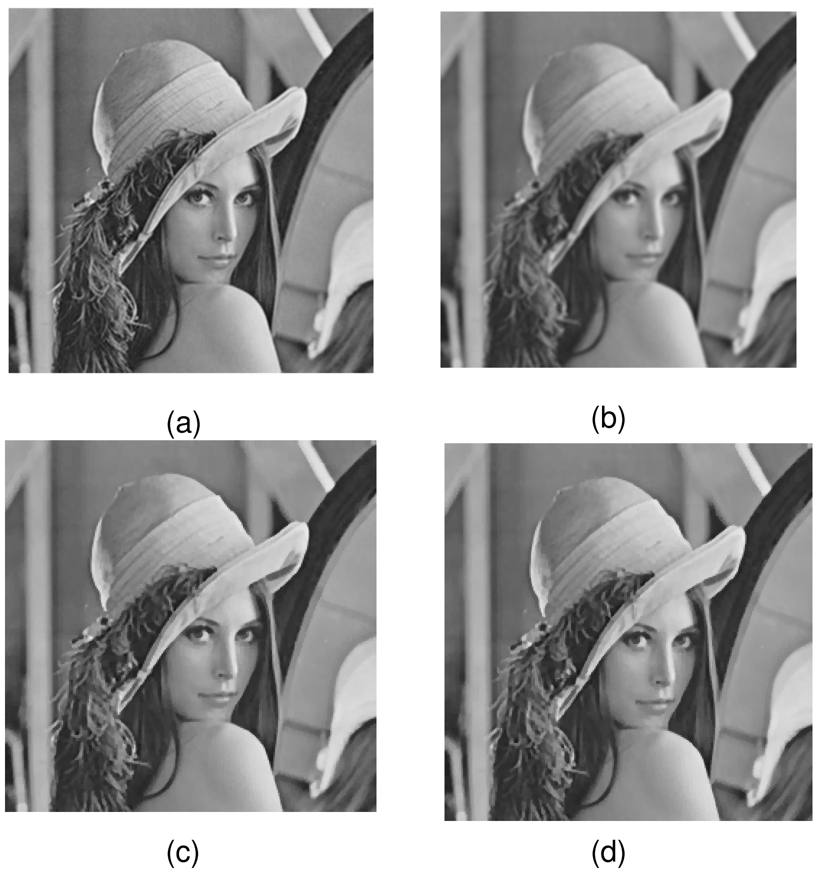
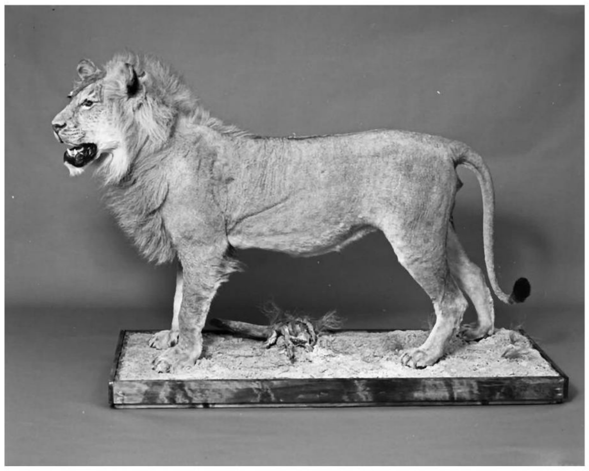


Figure 3.5: Results of deblurring using fused lasso penalty. Plots (a), (b), (c), and (d) correspond to the original image, the blurred image, the ALIN de-blurred image, and the FISTA de-blurred image, respectively.



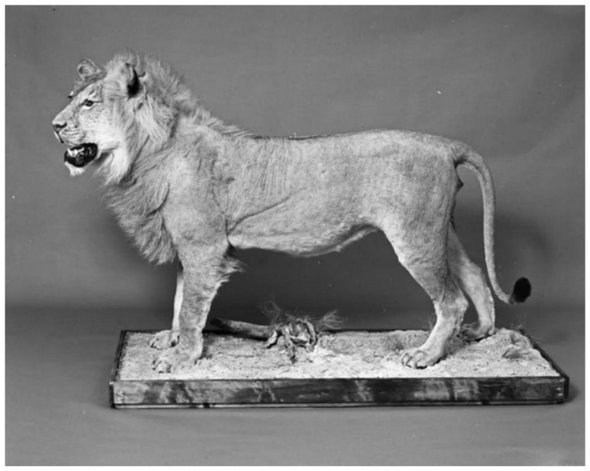


Figure 3.6: Image deblurring on the "lion" data. The left plot is the result from the ℓ_1-TV penalty; the right plot is from the Tikhonov penalty.

behind this formulation is that most image have a sparse representation in the wavelet domain. We can see that this problem is also an instance of the *Generalized Lasso* problem. In this experiment, we attempt to compare the two approach: total variation based and wavelet based. We use a *Haar* wavelet transformation for the columns of an image as the Wavelet basis matrix and ℓ_1-TV to make comparision. Deblurring was done on a set of testing images of small scale, about $40,000$ pixels. To compare the quality of the restored image, we use the signal-to-noise (SNR) ratio defined as

$$SNR = 10 \log 10 \frac{\|u^0 - \tilde{u}\|^2}{\|u^0 - u\|^2}, \tag{3.31}$$

where u^0 is the original image, $\tilde{u}$ is the mean intensity of the original image, and u is the restored image.

Table 3.3: Run time comparison on image deblurring - small sized images.

Method	CPU time (secs)	SNR	MSE
TV based	6.85	11.09	4.21
Wavelet based	6.83	8.94	6.47

We can see that the two methods take up similar amount of time for the deblurring problem. However, using discrete total variation penalty, we can restore images with higher quality than using wavelet basis.

There have been many efficient iterative methods proposed to solve this problem. Two outstanding general frameworks are a variation of Nesterov's gradient method [Nesterov, 2007] and the method of alternating direction (ADMM). SLEP is a variation of Nesterov method like FISTA, although it was specifically implemented for fused-lasso penalty. It is not directly applicable for total variation deblurring problem. We pick two algorithms to compare with ALIN in this numerical study: FISTA of Beck and Teboulle [2009], a very efficient first-order method for discrete total variation based image processing; and TVAL, a method based on Augmented Lagrangian and Alternating Direction algorithm. TVAL solves a model equivalent to (3.27), but with a coefficient μ in front of the least-squares term, instead of λ at the regularization.

In the first comparison, we pick 10 random grayscale images with small size, typically 205×205, or approximately $40,000$ pixels. Following the same procedure as described in [Beck and Teboulle, 2009] the image is blurred using a 3×3 kernel and a Gaussian noise with variance 10^{-2} is added. The deblurring procedure is run with a few different values for λ. We let FISTA runs 100 iterations with the $monotone$ feature, which keeps the objective function decrease monotonically, and the tolerance is set to 10^{-5}. Then we run ALIN to the same objective function value. For TVAL, unfortunately, we cannot proceed similarly. Thus we let TVAL run $10,000$ iterations or to "tol" $= 10^{-5}$, whichever comes first. In Table 3.4, we report the running time to produce the best quality restored image, where the regularization parameter $\lambda = 10^{-4}$, similar to what was suggested by [C. Li and Zhang, 2013]. We also report SNR and the mean squared error (MSE).

Table 3.4: Run time comparison on image deblurring - small size images.

Method	CPU time (secs)	SNR	MSE
FISTA	9.19	11.00	4.21
ALIN	6.85	11.03	4.18
TVAL	3.03	10.56	4.57

Although TVAL has superior performance in terms of running time, when compared to FISTA and ALIN, it produces an image of lower quality. With the same value of parameter λ, TVAL was not able to obtain the same objective function value as FISTA and ALIN. This makes the SNR of the TVAL-restored image lower and the error higher than those of ALIN and FISTA. ALIN and FISTA have similar performance in terms of image quality, but ALIN is more efficient than FISTA. In Figure 3.7, we plot the progression in terms of objective function values for all three methods. TVAL takes only 52 iterations to terminate. In this plot, ALIN and FISTA are set to terminate in 52 iterations.

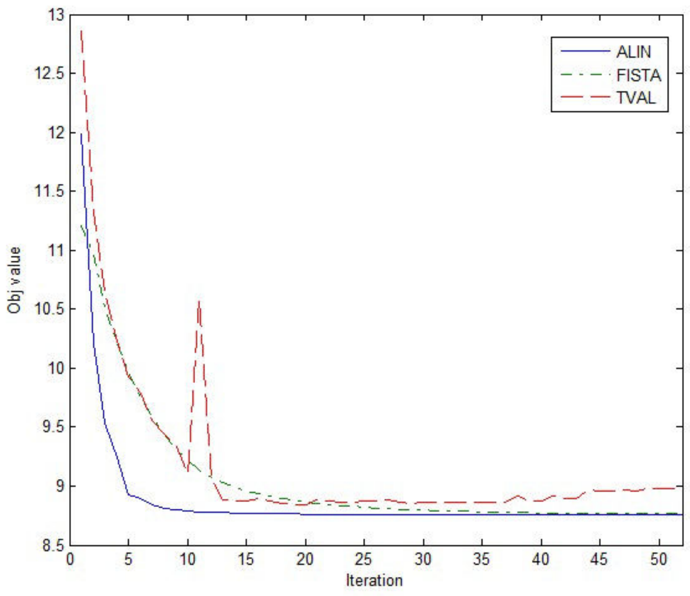


Figure 3.7: Progression in terms of objective function values of ALIN, FISTA, and TVAL

In the second comparison, we pick 10 random grayscale images with medium size, ranging from $200,000$ to $500,000$ pixels. The experiment is carried out in the same manner as the previous one. The results are reported in Table 3.5, and we observe the same pattern as in the previous comparison.

Table 3.5: Run time comparison on image deblurring - medium size images.

Method	CPU time (secs)	SNR	MSE
FISTA	65.41	12.14	5.98
ALIN	41.28	12.14	5.97
TVAL	7.18	8.30	14.36

3.6.5 Application to a narrative comprehension study for children

With high dimensional fused lasso penalty, the constrained optimization problem with identity design matrix is already difficult to solve, and a large body of literature has been devoted to solving this problem. When the design matrix is not full rank, the problem becomes much more difficult. In this section, we apply the three-dimensional fused lasso penalty to an regression problem where the design matrix X contains many more columns than rows.

Specifically, we perform regularized regression between the measurement of children's language ability (the response y) and voxel level brain activity during a narrative comprehension task (the design matrix X). Children develop a variety of skills and strategies for narrative comprehension during early childhood years [Karunanayaka et al., 2010]. This is a complex brain function that involves various cognitive processes in multiple brain regions. We are not attempting to solve the challenging neurological problem of identifying all such brain regions for this cognitive task. Instead, the goal of this study is to demonstrate ALIN's ability for solving constrained optimization problems of this type and magnitude.

The functional MRI data are collected from 313 children with ages 5 to 18 [Schmithorst et al., 2006]. The experimental paradigm is a 30-second block design with alternating stimulus and control. Children are listening to a story read by adult female speaker in each stimulus period, and pure tones of 1-second duration in each resting period. The subjects are instructed to answer ten story-related multiple-choice questions upon the completion of the MRI scan (two questions per story). The fMRI data were preprocessed and transformed into the Talairach stereotaxic space by linear affine transformation. A uniform mask is applied to all the subjects so that they have measurements on the same set of voxels.

The response variable y is the oral and written language scale (OWLS). The matrix X records the activity level for all the 8000 voxels measured. The objective function is the following:

$$\mathcal{L}(\beta) = \tfrac{1}{2}\|y - X\beta\|_2^2 + \lambda_1 h_1(\beta) + \lambda_2 h_2(\beta),$$

where

$$h_1(\beta) = \sum_{i=1}^{m-1}\sum_{j=1}^{n-1}\sum_{k=1}^{p-1}\{|\beta_{i,j,k} - \beta_{i+1,j,k}| + |\beta_{i,j,k} - \beta_{i,j+1,k}| + |\beta_{i,j,k} - \beta_{i,j,k+1}|\}$$
$$+ \sum_{i=1}^{m-1}\sum_{k=1}^{p-1}\{|\beta_{i,n,k} - \beta_{i+1,n,k}| + |\beta_{i,n,k} - \beta_{i,n,k+1}|\} + \sum_{j=1}^{n-1}\{|\beta_{m,j,p} - \beta_{m,j+1,p}|\}$$
$$+ \sum_{j=1}^{n-1}\sum_{k=1}^{p-1}\{|\beta_{m,j,k} - \beta_{n,j+1,k}| + |\beta_{m,j,k} - \beta_{m,j,k+1}|\} + \sum_{i=1}^{m-1}\{|\beta_{i,n,p} - \beta_{i+1,n,p}|\}$$
$$+ \sum_{i=1}^{m-1}\sum_{j=1}^{n-1}\{|\beta_{i,j,p} - \beta_{i+1,j,p}| + |\beta_{i,j,p} - \beta_{i,j+1,p}|\} + \sum_{k=1}^{p-1}\{|\beta_{m,n,k} - \beta_{m,n,k+1}|\},$$
$$h_2(\beta) = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p}|\beta_{i,j,k}|,$$

and $m = 31$, $n = 35$, and $p = 15$.

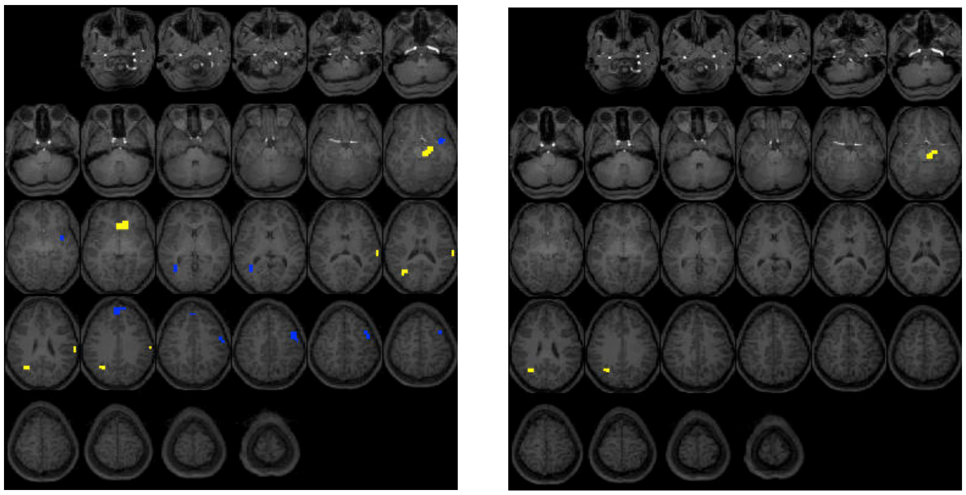


Figure 3.8: Results of regularization regression with combined lasso and 3-d fused lasso penalty. The tuning parameters of fused lasso is 0.2 for both figures. The tuning parameter for lasso is 0.2 for the left and 0.6 for the right.

While the main purpose of this study is to demonstrate the capability of the ALIN algorithm for solving penalized regression problems with 3-d fused lasso, there are also some interesting neurological observations. One objective of this study is to identify the voxels that are significant for explaining the performance score y. These voxels constitute active brain regions that are closely related to the OWLS. Figure 3.8 presents the results of fitted coefficients using combined lasso and fused lasso penalty. The highlighted regions shown in the maps are areas with more than 10 voxels (representing clusters of size 10 and above). The left plot in the figure is the optimal solution obtained using ten-fold cross validation. The optimal tuning parameters are 0.2 for both fused lasso and lasso penalties. Roughly speaking, five brain regions have been identified. The yellow area to the rightmost side of the brain is situated in the wernicke area, which is one of the two parts of the cerebral cortex linked to speech. It is involved in the understanding of written and spoken language. The only difference between the left and right plots is the value of the tuning parameter for the lasso penalty, which is 0.2 and 0.6 respectively. Clearly, the right plot shrinks more coefficients to zero, which results in a reduced number of significant regions, as compared to the left plot.

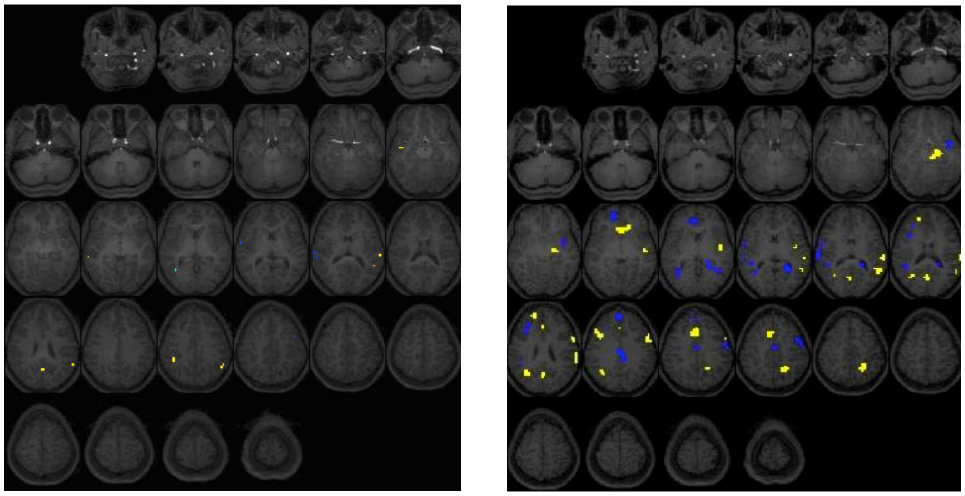


Figure 3.9: Results of regularization regression with lasso penalty (left plot) and Tiknonov type penalty (right plot).

We further study this regularization problem using only lasso penalty and Tiknonov type penalty similar to (3.30). Figure 3.9 shows the fitted maps. The left plot is the case where only lasso penalty is applied. Comparing this with Figure 3.8, we see that the 3-d fused lasso penalty imposes smoothing constraints on the neighboring coefficients, thus allowing to identify larger areas significant for the response variable y. The simple lasso penalty imposes shrinkage on the coefficients individually, resulting in rather disjoint significant voxels. Such scatterness is much less informative for neurologists than larger areas identified by the three-dimensional fused lasso penalty. Meanwhile, the Tiknonov type regularization generates too many significant regions as shown in the right plot. This is partially due to the over smoothing of the image as discussed in the previous section.

For comparison, we have considered a couple of methods designed to solve this particular problem. *Genlasso* is the path algorithm designed for the Generalized Lasso in the original paper. However, it was unable to handle an instance of this magnitude.

Another method is the Augmented Lagrangian and Alternating Direction method. The problem of interest

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|R\beta\|_1, \quad \lambda > 0 \tag{3.32}$$

can be reformulated as

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|z\|_1 \quad s.t : R\beta - z = 0. \tag{3.33}$$

The Augmented Lagrangian is:

$$\mathcal{L}(\beta, z, u) = \frac{1}{2}\|y - A\beta\|_2^2 + \lambda\|z\|_1 + \rho u^T(R\beta - z) + \frac{\rho}{2}\|R\beta - z\|_2^2, \tag{3.34}$$

where $\rho > 0$ is the penalty coefficient. This formulation can be solved by the Alternating Direction method. We implemented this method in Matlab. The update step for β requires minimizing a quadratic function. The iterative method of choice to solve the sub-problems is the conjugate gradient method built in Matlab. The performance of this method strongly depends on the choice of the penalty parameter ρ. As suggested in [B. Wahlberg, 2012], we choose $\rho = \lambda$ to keep the algorithm stable for the implementation. It is known that the method has a nice decrease in the function values but slow tail convergence and an iteration of ADMM for this particular problem is rather expensive so we let it run for 30 iterations and let ALIN run until it reached the same objective function value. The running times for different values of λ are reported in Table 3.6.

Table 3.6: Run time comparison on 3D fused lasso.

Method	$\lambda = 0.001$	$\lambda = 0.01$	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 1$
	CPU time (secs)	CPU time (secs)	CPU time (secs)	CPU time (secs)	CPU time (secs)
ALIN	20.68	12.19	18.58	23.45	129.81
ADMM	72.86	94.69	68.13	74.43	267.01

We also implemented FISTA for this particular problem. For each iteration k of FISTA, the following optimization problem is solved:

$$\min_{\beta} \mathcal{Q}_L(\beta, \beta^k) = f(\beta^k) + \langle \nabla f(\beta^k), \beta - \beta^k \rangle + \frac{L}{2}\|\beta - \beta^k\|^2 + g(\beta), \tag{3.35}$$

where f is Gaussian loss function and $g(\beta) = \|R\beta\|_1$. The parameter L is the Lipschitz constant of ∇f. This problem is similar to the f-subproblem of ALIN, so we utilize our own block-coordinate descent method to solve this problem. Since the Lipschitz constant L cannot be computed efficiently, we use back-tracking to find the proper L. Back-tracking is a popular method to find the right step size for FISTA iterations, however it can slow down the algorithm significantly. We observe that in each iteration, back-tracking will have to solve the sub-problem 20 to 30 times to find a good approximation to the Lipschitz constant of ∇f. Normally, this quantity is approximated by the maximum eigenvalue of $X^T X$. However, in the $p \gg n$ setting, it is not computationally efficient to estimate eigenvalues. When FISTA is applied to this data set with 3-d fused lasso penalty, it needs around $10,000$ iterations to reach the same objective function value as ADMM, and the running time is more or less an hour. This is also in line with what we observe from the implementation of FISTA for 1-d fused lasso penalty in the package SPAMS.

Chapter 4

Nonconvex penalties

4.1 Introduction

Consider the linear regression setting:

$$y = X\beta$$

where $X \in \mathcal{R}^{n\times p}$ is the matrix of covariates, $y \in \mathcal{R}^{n\times 1}$ is the vector response variables, and $\beta \in \mathcal{R}^{p\times 1}$ is the vector of coefficients. When $n \geq p$, i.e. the number of observations is larger than the number of independent variables, there is a closed form solution to β. However, when $p \gg n$, the problem has multiple solutions and it can lead to over-fitting or picking too many variables into the model. In many applications, the true coefficient vector β is sparse, as in there are only a few influential predictors while many other predictors are zero. We would like to obtain a sparse model with only a small number of useful predictors which still closely estimate the true underlying coefficients and facilitate interpretation of the model. Selecting the right variables among many thousands is a combinatorially hard problem. It can be formulated as an optimization problem as follows:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_0, \quad \lambda > 0.$$

Parameter $\lambda > 0$ controls the sparsity of the coefficient estimation, $\|\beta\|_0 = \sum_{i=1}^{p} \mathbb{I}(|\beta_i| > 0)$. Finding the solution to this problem requires enumerating all possible subsets of coefficients and it is computationally intractable to do so. Many methods have been proposed to find an approximate solution. The penalized least square with ℓ_1 penalty:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1, \quad \lambda > 0.$$

is a convex surrogate to the formulation above. The ℓ_1 penalty tends to shrink coefficients to zero. This method has many attractive statistical and computational properties. With certain conditions on the matrix X and the sparsity of the underlying model, it can provide estimates with good accuracy.

A general structural lasso framework was proposed in [Tibshirani and Taylor, 2011], with the following form:

$$\mathcal{L}(\beta) = f(\beta) + \lambda \|R\beta\|_1, \quad \lambda > 0, \tag{4.1}$$

where R is an $m \times p$ matrix that defines the structural constraints one wants to impose on the coefficients. Many regularization problems, including high dimensional fused lasso, graph induced fused lasso, and discrete total variation can be cast in this framework. Fused Lasso penalty and its higher-order variants have been applied with much success in the problem of *hot-spot detection* of CGH data in cancer research [Tibshirani et al., 2005] [Rapaport et al., 2008]. ArrayCGH is a popular technique in bioinformatics to detect chromosomal abberations of the genes along the genome. These abberations come in the forms of gains or losses in the number of DNA copies of a gene. Studies have shown that large structural chromosomal abberations might be associated with increased risk of cancer. CGH data fit in the $p \gg n$ settings because the number of genes are usually in high order magnitude of the number of collected samples. However, variable selection techniques such the *Lasso* fail to detect the influence of interacting group of genes. With the assumption that neighboring genes should have similar effects, utilizing structural penalties such as Fused Lasso is more appropriate.

Although the $\ell_1 - norm$ and *Generalized Lasso* have been successfully in many applications, they have critical limitations. The most evident one is the biase introduced by the penalty on large coefficients [Zhang and Huang, 2008]. A good penalty function should have three desirable properties: *Unbiasedness* when the true unknown coefficient is large; *Sparsity* that results in a sparse solution to reduce model complexity; *Continuity* of the estimator in terms of the data to obtain model stability [Fan and Li, 2001]. Many different penalty functions have been proposed towards these criteria. Among those, we consider two penalties: smoothly clipped absolute deviation (SCAD) penalty [Fan and Li, 2001] and the minimax concave penalty (MCP) [Zhang, 2010]. Under mild conditions, these penalties are shown to have the oracle property, i.e: the penalty can choose the submodel correctly as the size of the problem grows and the nonzero coefficients can be estimated as if the true model was known in advance. However, these penalties are nonconvex therefore finding a solution to the optimization problem is computationally challenging. With these nonconvex penalties, the general structural lasso becomes:

$$\mathcal{L}(\beta) = f(\beta) + \mathcal{P}(|R\beta|; \lambda; \gamma), \quad \lambda > 0. \tag{4.2}$$

There have been several iterative methods proposed to solve these nonconvex penalty problem. When $R = \mathbb{I}$, Local Quadratic Approximation (LQA) is the first method proposed for SCAD penalty [Fan and Li, 2001]. At every iteration, it computes a quadratic approximation to the penalty function and solves the optimization subproblem using Newton-Raphson method. LQA requires a starting point being carefully chosen. [Zou and Li, 2008] proposed a Local Linear Approximation method that utilizes linear approximation instead of quadratic. Starting from an initial least square solution, it iteratively solves a weighted lasso problem where the weights are obtained from the previous iterations. Coordinate descent method can also be used to solve this problem efficiently [Breheny and Huang, 2011], [Mazumder et al., 2011]. Both methods require to solve the problem on a grid for tuning parameter λ. For a single value of parameter λ, coordinate descent does not work very well. Its efficiency relies on having a good starting point, which is the solution of the previous value of λ on the grid, and its descent property. No function evaluation is needed. However, it is difficult to determine the density of the grid.

When the structural matrix R is more complicated, which makes the penalty function nonseparable, these methods are not directly applicable. Recently [Zhou and Wu, 2012] propsed a path algorithm for a general structure penalty. However, this method requires certain conditions on the rank of the matrix R.

In this chapter, we present a system of methods based on the alternating linearization framework to solve this nonconvex structured penalty problem. We will describe our algorithm and show how it can be applied to linear regression and logistic regression. The method has been tested on several simulated data sets and two arrayCGH data sets on melanoma and bladder tumours. The experiments have shown that solutions obtained by using the nonconvex structured penalties are superior to those obtained by structured penalties.

4.2 Alternating linearization method for nonconvex structured penalties

4.2.1 Structured nonconvex penalties

In the context of traditional linear regression, variable selection is a very important problem. Penalized least square methods receive much attention since it can automatically and simultaneously select variables unlike classical variable selection methods. Penalized methods often have the form:

$$\min_{\beta} f(\beta) + \sum_{i} \mathcal{P}_{\lambda}(|\beta_i|). \tag{4.3}$$

In linear reression problem, the loss function (β) is the least square loss and it can be extended to the Generalized Linear Models (GLM) by using different types of loss functions such as log-likelihood function. One of the most well-known penalty functions is the ℓ_1 penalty (LASSO). The LASSO has been studied intensively in both computational methods and theoretical properties. The ℓ_1 penalty, however, is biased even for large coefficients β since LASSO penalizes the coefficients with the same rate λ [Fan and Li, 2001], [Zhang and Huang, 2008].

[Fan and Li, 2001] suggests that a good penalty function should be approximately unbiased when the true unknown coefficient is large. It also suggests that in order to reduce biasedness in the estimator, the derivative of the penalty function should be 0 for large coefficients. Following these criteria, two nonconvex penalty functions have been proposed: SCAD and MCAP [Fan and Li, 2001],[Zhang, 2010]. In this section, we consider these two popular penalties, although the method is fairly general and can be applied for other penalty functions.

- The SCAD penalty is defined as the follows:

$$\mathcal{P}(|\beta|;\lambda;\gamma) = \begin{cases} \lambda|\beta| & if \quad |\beta| \le \lambda. \\ \frac{\gamma\lambda|\beta| - 0.5(|\beta|^2+\lambda^2)}{\gamma-1} & if \quad \lambda < |\beta| \le \lambda\gamma. \\ \frac{\lambda^2(\gamma+1)}{2} & if \quad |\beta| > \lambda\gamma. \end{cases}$$

$$\frac{\partial \mathcal{P}(|\beta|;\lambda;\gamma)}{\partial|\beta|} = \begin{cases} \lambda & if \quad |\beta| \le \lambda. \\ \frac{\lambda\gamma - |\beta|}{\gamma-1} & if \quad \lambda < |\beta| \le \lambda\gamma. \\ 0 & if \quad |\beta| > \lambda\gamma. \end{cases}$$

- MCP penalty is defined as follows:

$$\mathcal{P}(|\beta|;\lambda;\gamma) = \begin{cases} \lambda|\beta| - \frac{|\beta|^2}{2\gamma} & if \quad |\beta| \le \lambda\gamma. \\ \frac{\lambda^2\gamma}{2} & if \quad |\beta| > \lambda\gamma. \end{cases}$$

$$\frac{\partial \mathcal{P}(|\beta|;\lambda;\gamma)}{\partial |\beta|} = \begin{cases} \lambda - \frac{|\beta|}{\gamma} & if \quad |\beta| \le \lambda\gamma. \\ 0 & if \quad |\beta| > \lambda\gamma. \end{cases}$$

The two penalty functions are controlled by two parameters λ and γ. When coefficient β is not large, the derivative of the penalty functions are the same as in the LASSO. However, when the coefficient is large enough ($> \gamma\lambda$) for both penalty functions, penalty functions are flattened out and the corresponding coefficients are not excessively penalized.

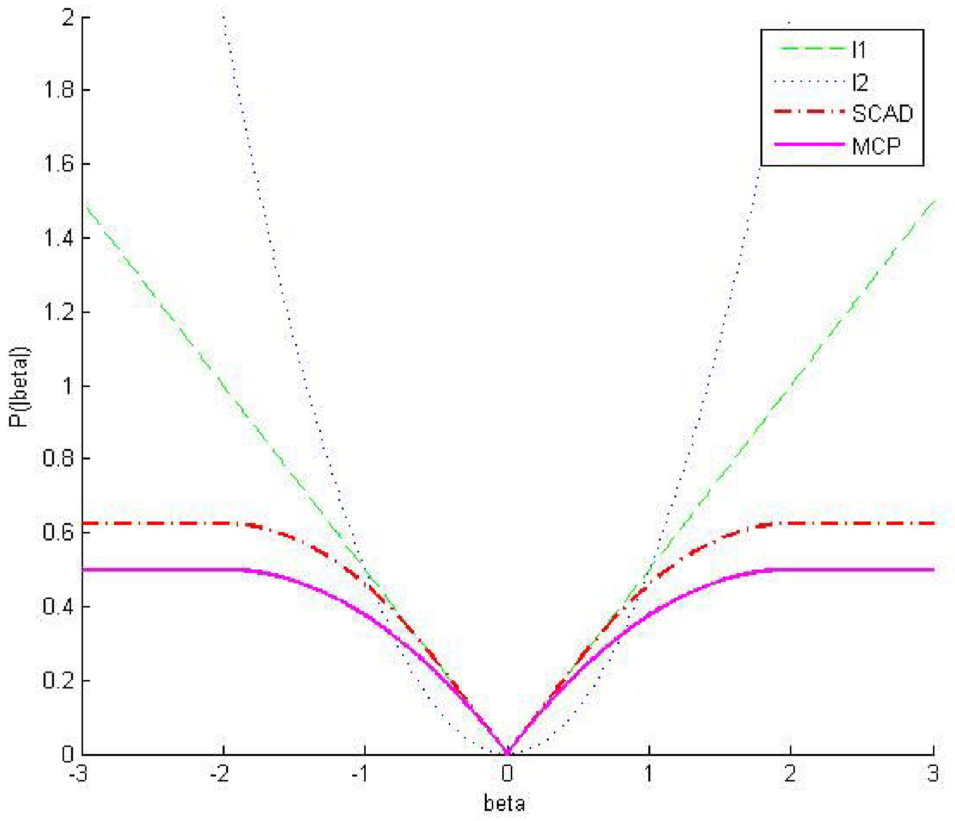


Figure 4.1: Plot of different penalty functions including ℓ_1, ℓ_2, SCAD, and MCP

We propose a new formulation combining the structured penalty and the nonconvex penalty. As mentioned in the previous chapters, if there was prior knowledge or domain constraints on the parameters of the model, this information can be incorporated in the penalty function. These structured penalty functions, however, are variations of the ℓ_1 norm penalty which suffer from similar estimation bias. The formulation we propose is the following:

$$\min_{\beta} \mathcal{L}(\beta) = f(\beta) + \sum_{i=1}^{k} \mathcal{P}(|R_i\beta|; \lambda; \gamma), \tag{4.4}$$

where R_i is the ith row of matrix $R \in \mathcal{R}^{k\times p}$. When $R = \mathbf{I}$, the function (4.4) becomes the original nonconvex penalties.

4.2.2 Logistic regression with nonconvex penalties

Logistic regression is a very popular tool in statistics that has seen tremendous applications acrosss disciplines. As opposed to linear regression, the response vector y is a binary vector, $y \in \{0,1\}^n$. The goal is to model the probability that a point $x_i \in \mathbb{R}^p$ belong to class 1 as a function of $x_i^T\beta + c$. This is done via a logit transformation of the probability: $logit(p_i) = \frac{p_i}{1-p_i}$. The resulting model is: $logit(p_i) = x_i^T\beta + c$. We have:

$$P(y_i = 1|x_{i1}, x_{i2}, \cdots, x_{ip}) = p_i = \frac{e^{x_i^T\beta+c}}{1+e^{x_i^T\beta+c}}, \tag{4.5}$$

where c is the intercept. The hyperplane $x^T\beta + c = 0$ corresponds to $P(y = 1|x) = 0.5$. Response variable y_i follows Bernoulli distribution with probability p_i. The likelihood function with n observations is:

$$L(y;\beta) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}. \tag{4.6}$$

Consider the negative log-likelihood function and substitute p_i as above we obtain:

$$\begin{aligned} l(y;\beta;c) &= -\sum_{i=1}^{n}\{y_i \log p_i + (1-y_i)\log(1-p_i)\} \\ &= -\sum_{i=1}^{n}\{y_i \log \frac{e^{x_i^T\beta+c}}{1+e^{x_i^T\beta+c}} + (1-y_i)\log\frac{1}{1+e^{x_i^T\beta+c}}\} \\ &= -\sum_{i=1}^{n}\{y_i(x_i^T\beta+c) - y_i\log(1+e^{x_i^T\beta+c}) - (1-y_i)\log(1+e^{x_i^T\beta+c})\} \\ &= \sum_{i=1}^{n}\{-y_i(x_i^T\beta+c) + \log(1+e^{x_i^T\beta+c})\}. \end{aligned} \tag{4.7}$$

The derivatives of $l(y;\beta;c)$ with respect to c and β_j are:

$$\begin{aligned}\frac{\partial l}{\partial c} &= \sum_{i=1}^{n} \frac{e^{x_i^T\beta+c}}{1+e^{x_i^T\beta+c}} - y_i. \\ \frac{\partial l}{\partial \beta_j} &= \sum_{i=1}^{n} \frac{e^{x_i^T\beta+c}x_{ij}}{1+e^{x_i^T\beta+c}} - y_i x_{ij}.\end{aligned} \tag{4.8}$$

We need to compute the parameter estimates β and c that minimize the negative log-likelihood function. When $p >> n$, this function $l(:;\beta;c)$ has multiple minima. In this section, we consider this loss function with nonconvex and structured nonconvex penalty function:

$$Q(\beta;c) = \sum_{i=1}^{n}\{-y_i(x_i^T\beta + c) + \log(1 + e^{x_i^T\beta+c})\} + \sum_{j=1}^{p}\mathcal{P}(|\beta_j|;\lambda;\gamma). \tag{4.9}$$

and

$$Q(\beta;c) = \sum_{i=1}^{n}\{-y_i(x_i^T\beta + c) + \log(1 + e^{x_i^T\beta+c})\} + \mathcal{P}(|R\beta|;\lambda;\gamma). \tag{4.10}$$

where $\mathcal{P}$ is a nonconvex penalty function.

4.2.3 ALIN for nonconvex penalties with Gaussian loss and Logistic loss

When the structure matrix $R = \mathbb{I}$, we have the original regularization problem with nonconvex penalty:

$$\min_{\beta} \mathcal{L}(\beta) = f(\beta) + \sum_{i=1}^{p}\mathcal{P}(|\beta_j|;\lambda;\gamma)$$

Loss function $f(\beta)$ can be the Gaussian loss $\|y - X\beta\|_2^2$ or the logistic loss defined above. At each iteration k, the current estimate is β^k. The objective function is approximated by:

$$f(\beta) + \sum_{i=1}^{p}\partial\mathcal{P}(|\beta_i^k|;\lambda;\gamma)(|\beta_i| - |\beta_i^k|).$$

Simplifying the constant and denote $\partial\mathcal{P}(|\beta_i^k|;\lambda;\gamma) = \lambda_i^k$, at iteration k we have, the following sub-problem:

$$\tilde{\mathcal{L}}^k = f(\beta) + \sum_{i=1}^{p}\lambda_i^k|\beta_i|.$$

This sub-problem can also be solved via alternating linearization method described in Chapter 3 as shown below:

- Step 1: The f-subproblem is as follows:

$$\min_{\beta} \tilde{f}(\beta) + \sum_{i=1}^{p}\lambda_i^k|\beta_i| + \frac{1}{2}\|\beta - \hat{\beta}^k\|_D^2.$$

where $\tilde{f}(.)$ is a linear approximation of $f(.)$ and $\hat{\beta}^k$ is the current estimate of the sub-problem. The f-subproblem is separable and the solution has closed form formula.

- Step 2: Update step and calculate the subgradient of function $g(\beta) = \sum_{i=1}^{p} \lambda_i^k |\beta_i|$ as described in Chapter 3.
- Step 3: The h-subproblem is as follows:

$$\min_{\beta} f(\beta) + \tilde{g}(\beta) + \frac{1}{2}\|\beta - \hat{\beta}^k\|_D^2.$$

where $\tilde{g}(\beta)$ is the linear approximation $g(\beta)$. The h-subproblem is solved using the conjugate gradient method. In the case when $f(\beta)$ is the logistic loss function, conjugate gradient method for nonlinear functions are used.

4.2.4 ALIN for structured nonconvex penalties

In the case of a generic sparse structure matrix R, we have the problem:

$$\min_{\beta} \mathcal{L}(\beta) = f(\beta) + \mathcal{P}(|R\beta|; \lambda; \gamma)$$

At each iteration k, the current estimate is β^k. The objective function is approximated by:

$$f(\beta) + \mathcal{P}(|R\beta^k|; \lambda; \gamma)(|R\beta| - |R\beta^k|).$$

Simplifying the constant and denote $\partial\mathcal{P}(|R\beta^k|; \lambda; \gamma) = \lambda^k$, at iteration k, the following sub-problem is solved:

$$\min_{\beta} \tilde{\mathcal{L}}^k = f(\beta) + \langle \lambda^k, |R\beta| \rangle.$$

This sub-problem can be solved via alternating linearization method described in Chapter 3. Here we provide a brief description.

- Step 1: The f-subproblem is as follows:

$$\min_{\beta} \tilde{f}(\beta) + \langle \lambda^k, |R\beta| \rangle + \frac{1}{2}\|\beta - \hat{\beta}^k\|_D^2.$$

where $\tilde{f}(.)$ is a linear approximation of $f(.)$ and $\hat{\beta}^k$ is the current estimate of the sub-problem. The dual of the f-subproblem is the minimization of a quadratic function subjected to a box-constraint. Block coordinate descent method can be used to solve this problem.

- Step 2: Update step and calculate the subgradient of function $g(\beta) = \langle \lambda^k, |R\beta| \rangle$ as described in Chapter 3.

- Step 3: The h-subproblem is as follows:

$$\min_{\beta} \mathcal{G}(\beta) = f(\beta) + \tilde{g}(\beta) + \frac{1}{2}\|\beta - \hat{\beta}^k\|_D^2.$$

  where $\tilde{g}(\beta)$ is the linear approximation $g(\beta)$. The h-subproblem is solved using the conjugate gradient method. In the case when $f(\beta)$ is the logistic loss function, conjugate gradient method for nonlinear functions are used.

4.2.5 Nonlinear conjugate gradient method

In the previous section, we are interested in solving the following problem:

$$\min_{b} \mathcal{G}(\beta) = \log(1 + e^{X^T\beta}) - y^T X^T \beta + s^T \beta + \frac{1}{2}\|\beta - \hat{\beta}\|_D^2.$$

In this section, we present the application of the nonlinear conjugate gradient method to solve this problem.

1. Step 0: Set k=1.

2. Step 1: Caculate $\nabla \mathcal{G}(\beta^k) = X^T(\frac{e^{X^T\beta^k}}{1+e^{X^T\beta^k}} - y) + s + (\beta^k - \hat{\beta})$. If $\nabla \mathcal{G}(\beta^k) = 0$ then stop.

3. Step 2: Compute

$$d^k = \begin{cases} -\nabla \mathcal{G}(\beta^k) & \text{if } k = 1. \\ -\nabla \mathcal{G}(\beta^k) + \alpha_k d^{k-1} & \text{if } k > 1. \end{cases}$$

   where

$$\alpha_k = \frac{\langle \nabla \mathcal{G}\beta^k, \nabla \mathcal{G}\beta^k - \nabla \mathcal{G}\beta^{k-1} \rangle}{\|\nabla \mathcal{G}\beta^{k-1}\|^2}.$$

4. Step 3: Find the next point $\beta^{k+1} = \beta^k + \tau_k d^k$ such that

$$\mathcal{G}(\beta^{k+1}) = \min_{\tau \geq 0} \mathcal{G}(\beta^k + \tau d^k).$$

   where τ_k can be found using methods such as binary search or secant method.

5. Step 4: Increase k by 1 and go to Step 1.

Notice that, in each sub-problem of ALIN, we can use the result of the previous iteration as the starting point of the next one and this significantly accelerates the whole algorithm.

4.3 Numerical examples

In this section we demonstrate the ability of ALIN with several competing algorithms for nonconvex penalized regression including NCVREG (by Breheny and Huang), another implementation of coordinate descent method SparseNet (Mazumder), and path following algorithm SparseReg (Zhou). The criteria that we use to make comparision are prediction error, number of non-zero coefficients in the model, and misclassification error. We also study the effectiveness of our method on the problem with structured nonconvex penalties, in particular the fused lasso type penalty.

4.3.1 Simulation studies - Linear regression

In the simulation studies, we assume a linear model $Y = X\beta + \epsilon$ with multivariate Gaussian predictors X and Gaussian noise. We generate datasets with the number of observations n=400 and the number of covariates $p = 500$ and $p = 1000$. 100 observations in the dataset are used as training dataset and the other 300 observations are used in the testing dataset. The pre-determined β has block structures as following:

$$\beta_j = \begin{cases} 1 & \text{for } j = 2, 3 \ldots, 11, \\ 5 & \text{for } j = 102, 103, \ldots, 111, \\ 10 & \text{for } j = 202, 203, \ldots, 211, \\ 50 & \text{for } j = 302, 303, \ldots, 311, \\ 100 & \text{for } j = 402, 403, \ldots, 49, \\ 0 & \text{otherwise.} \end{cases}$$

The covariate matrix X is generated from a multivariate normal distribution with pairwise correlation of ρ between covariates in each block. Two different settings are used in our experiments: high correlation with $\rho = 0.9$ and low correlation with $\rho = 0.1$.

In the first experiment, we compare the performance of NCVREG, SparseNet, SparseReg, and ALIN on these two data sets using SCAD and MCP penalties. With SCAD penalty and $\rho = 0.1$, the solutions provided by NCVREG, SparseReg, and ALIN are very similar. However, in the high correlation setting, we can see that the solution by ALIN is much better than those by CD and SparseReg (Fig 1). NCVREG and SparseReg were able to recover many underlying coefficients, however the values of these coefficients are not as accurate as ALIN.

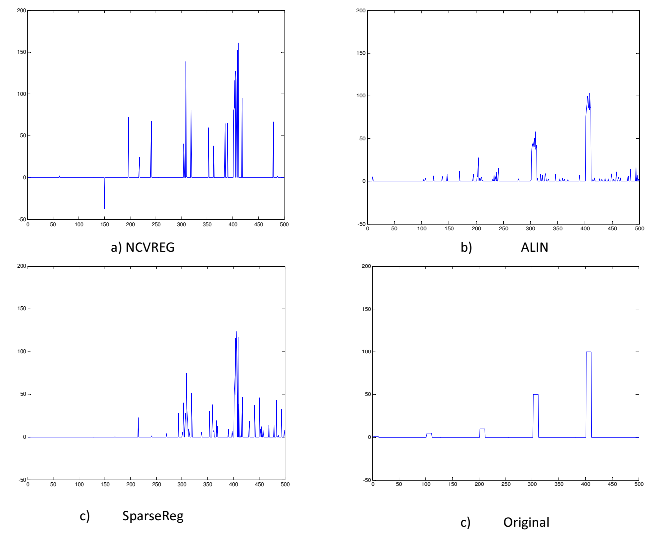


Figure 4.2: Results of using SCAD penalty on data set with $n = 100$, $p = 500$, and $\gamma = 3.7$, $\rho = 0.9$. Plots (a), (b), (c), and (d) correspond to results from NCVREG, ALIN, SparseReg, and original solution.

We then compare these three algorithms on the testing data set. The comparisons are made with regards to three criteria: prediction error, the number of non-zero coefficients in the model, and misclassification error. The discovered models are used to make prediction on the testing data set and standardized prediction errors are recorded. The number of true non-zero variables in the original model is 50. In Fig 2., we can see the comparisons between the three algorithms with regards to those criteria. Although, the model by ALIN is not as sparse as NCVREG and SparseReg for some values of γ, ALIN was able to recover more parameters and achieve better accuracy. This explains why the prediction error by ALIN is much better than those of NCVREG and SparseReg. We further extended this experiment to the data set in higher dimension, n=100, p=1000, generated in a similar manner. We observed the same results when

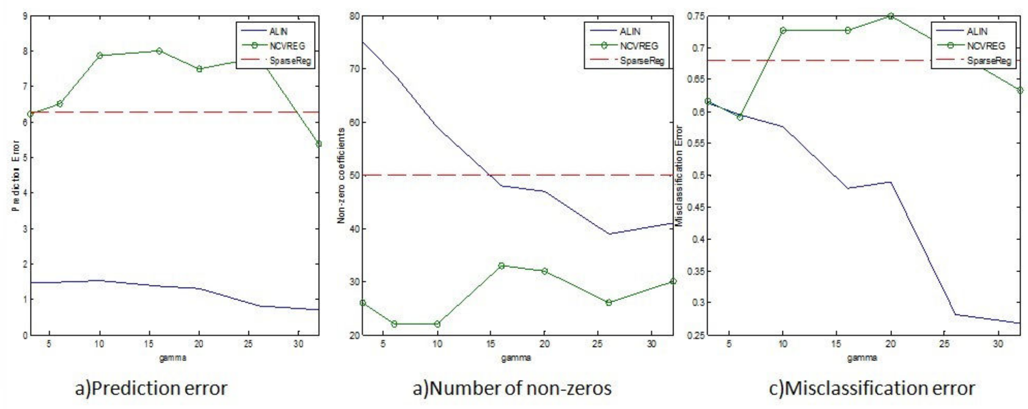


Figure 4.3: Three plots represented prediction error, model sparsity, and misclassification loss. Plots show solutions for different values of γ of SCAD penalty.

$\rho = 0.1$. In low correlation setting, NCVREG, SparseReg, and ALIN have similar solutions but in high correlation case, ALIN performs better than the other two algorithms.

Now we turn to the performance of three agorithms SparseNet, NCVREG, and ALIN on the same data sets with MCP penalty. The parameters λ and γ are estimated on the a grid of values via cross-validation. We observe the same phenomena where all three algorithms provide very similar solutions in low correlation setting, however when $\rho = 0.9$, ALIN computed a better solution as demonstrated in the Fig 3. The prediction error by ALIN solution is also much lower than those by SparseNet and CD.

The following table summarizes the performance of all four algorithms with regards to standardized prediction error (SPE), % of non-zeros in the recovered models, and misclassification error. Although ALIN does not provide as sparse models as NCVREG, it is able to recover more underlying coefficients. As the result, in terms of SPE, ALIN is better than the other three methods in all simulated datasets, especially in high correlation and high dimension settings.

n=100, p=500, ρ=0.1	ALIN	NCVREG	SparseNet	SparseReg
SPE	0.24	0.28	1.55	3.55
% of non-zeros [10]	19.20	14.00	9.00	15.00
0-1 error	0.49	0.30	0.20	0.56
n=100, p=500, ρ=0.1	ALIN	NCVREG	SparseNet	SparseReg

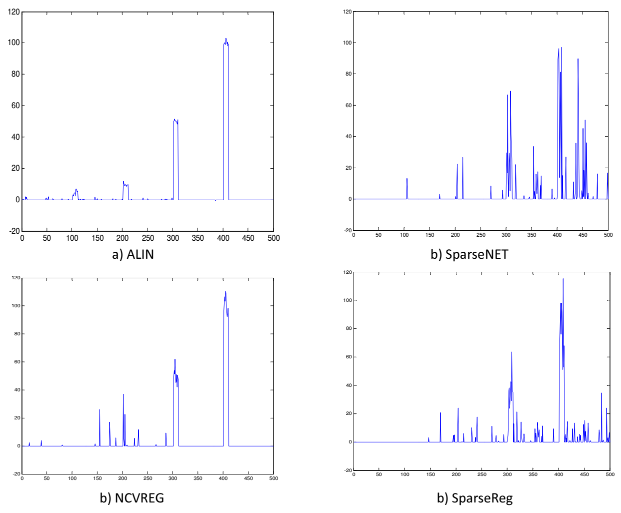


Figure 4.4

n=100, p=500, ρ=0.9	ALIN	NCVREG	SparseNet	SparseReg
SPE	0.19	1.10	6.13	2.90
% of non-zeros [10]	18.80	9.00	12.20	15.80
0-1 error	0.51	0.38	0.67	0.73

n=100, p=1000, ρ=0.1	ALIN	NCVREG	SparseNet	SparseReg
SPE	0.56	2.40	10.77	9.10
% of non-zeros [5]	13.40	7.70	11.10	9.30
0-1 error	0.35	0.58	0.75	0.58

n=100, p=1000, ρ=0.9	ALIN	NCVREG	SparseNet	SparseReg
SPE	0.83	2.52	6.48	5.08
% of non-zeros [5]	9.60	4.70	9.00	6.70
0-1 error	0.33	0.49	0.79	0.71

In this section we demonstrate a simulated example using the nonconvex structured penalty and compare it with the structured penalty. We generate a data set in a similar manner with a block-structured coefficient vector β. Specifically, the number of observations $n = 400$ and the number of attributes $p = 500$. The coefficient vector β has block structure with 10 blocks and each

Table 4.1: Comparison between fusion penalty, fusion SCAD, and fusion MCP

	$\sum(\hat{Y}-Y)^2$	$\|\hat{\beta}-\beta\|_2$	Standard error
Fusion	3.1830	2.2448	0.4490
Fusion SCAD	1.6654	1.5044	0.2554
Fusion MCP	1.8361	1.5657	0.5019

block has 10 consecutive coefficients. The coefficients in each block are randomly generated by an uniform distribution in [-5,5]. These blocks are randomly positioned between 1 and 500. The first 100 observations are used as the training data set and the other 300 observations are for testing. In our experiments, we use the fused lasso penalty. The problem of interest is :

$$\min_{\beta} \mathcal{L}(\beta) = \tfrac{1}{2}\|y - X\beta\|_2^2 + P(|R\beta|; \lambda; \gamma),$$

where R is a $(p-1) \times p$ matrix to enforce structure on coefficient estimate β, $P(:; \lambda; \gamma)$ is a nonconvex penalty function. In our experiment, we use the fused lasso structure so matrix R is:

$$R = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix},$$

We need to solve the problem:

$$\min_{\beta} \mathcal{L}(\beta) = \tfrac{1}{2}\|y - X\beta\|_2^2 + \textstyle\sum_{i=2}^{p} P(|\beta_{i-1} - \beta_i|; \lambda; \gamma),$$

where $P(.; \lambda; \gamma)$ is the SCAD or MCP penalty. The parameter γ is fixed at 3.7, and λ is chosen via cross-validation. To make comparisons, we use the traditional fused penalty:

$$\min_{\beta} \mathcal{L}(\beta) = \tfrac{1}{2}\|y - X\beta\|_2^2 + \lambda \textstyle\sum_{i=2}^{p} |\beta_{i-1} - \beta_i|.$$

Parameter λ is also chosen by using cross-validation. As we can see from the illustration, using the SCAD and MCP structured penalty, ALIN was able to recover the estimate β almost as good as the original solution while using fused penalty failed to do this. SCAD penalty alone works very poorly on this simulated data set. The following table reports the result of mean squared error of predictors and β for fusion penalty, fusion SCAD, and fusion MCP after 10 simulated experiments. We can see that structured nonconvex penalty such as fusion SCAD and fusion MCP outperformed traditional fusion penalty in terms of both prediction accuracy and squared error.

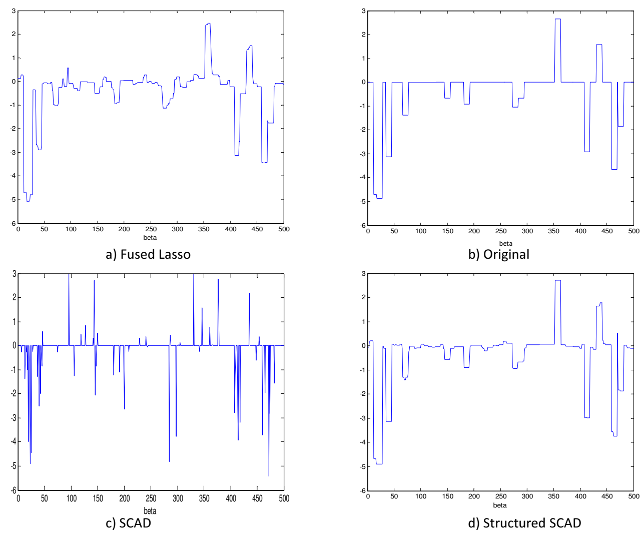


Figure 4.5: Result of SCAD and MCP structured penalty (a) and (b) compared to Fused penalty (c), and original solution

4.3.2 Simulation studies - Logistic regression

In this section, we reports the results of our method for solving the fused logistic regression problem with nonconvex penalties in comparison with fused lasso logistic regression. Using simulated datasets, we show that when features have natural ordering, structured nonconvex penalties perform better than fused lasso penalty. Regression coefficients $\beta \in \mathbb{R}^{500}$ with block structure is pre-determined. β has 7 blocks; the level of each block is drawn from an uniform random distribution between $[-10, 10]$; each level has length 20. A typical β is similar to the following:

$$\beta_j = \begin{cases} l_1, & \text{for } j = 5, 6, \cdots, 25 \\ l_2, & \text{for } j = 25, 26, \cdots, 45 \\ l_3, & \text{for } j = 100, 101, \cdots, 120 \\ \cdots & \cdots \end{cases}$$

where l_1, l_2, l_3 are drawn from a uniform random distribution between $[-10, 10]$. The design matrix $X \in \mathbb{R}^{n \times p}$ with $n = 200$ and $p = 500$ is drawn from a multivariate Gaussian distribution $\mathcal{N}(0, 1)$ with pairwise correlation between covariates $\rho = 0.3$. The intercept c is also drawn from

Gaussian distribution $\mathcal{N}(0,1)$. The success probability p_i for each observation x_i is calculated as folllowed: $p_i = \frac{e^{x_i^T\beta+c}}{1+e^{x_i^T\beta+c}}$. Response variable $y_i \in \{0,1\}$ is then generated using Bernoulli distribution with success probability p_i. The objective function that we are considering is :

$$\mathcal{L}(\beta) = \sum_{i=1}^{n} -y_i(x_i^T\beta + c) + \log(1 + e^{x_i^T\beta+c}) + \sum_{j=1}^{p-1} \mathcal{P}(|\beta_{j+1} - \beta_j|; \lambda; \gamma). \tag{4.11}$$

where $\mathcal{P}$ is a nonconvex penalty SCAD or MCP. The strutural matrix $R \in \mathbb{R}^{499\times500}$ is :

$$R = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix},$$

In comparison, we use a routine for fused lasso logistic regression in the SLEP package by [Liu et al., 2010b]. The fusion logistic regression objective function is:

$$\mathcal{F}(\beta) = \sum_{i=1}^{n} -y_i(x_i^T\beta + c) + \log(1 + e^{x_i^T\beta+c}) + \lambda \sum_{j=1}^{p-1} |\beta_{j+1} - \beta_j|. \tag{4.12}$$

The experiment is repeated 10 times with different levels and non-zero areas for β. For fusion

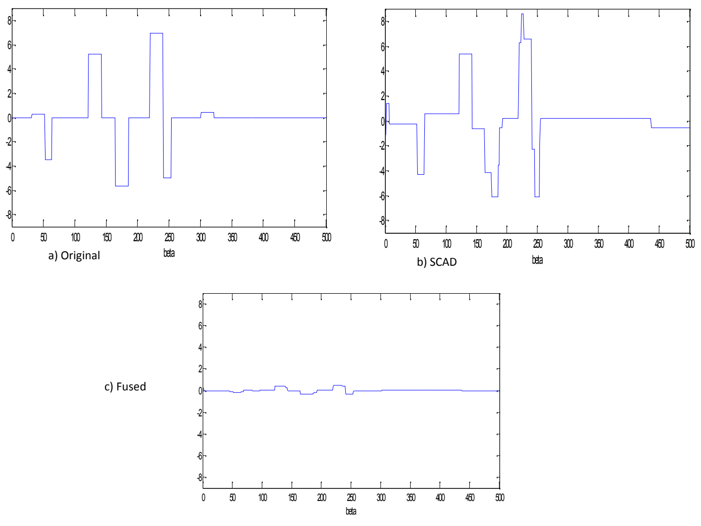


Figure 4.6: Result of SCAD structured penalty (a) and (b) compared to fusion penalty (c), and original solution

penalty, the average mean squared error (MSE) is 2.57. For structured SCAD penalty, MSE is

2.05. We have similar results as the number of blocks and block levels are varied. The structured penalty did well in detecting non-zero areas and recovering true levels. Illustration showed that although fusion penalty did decently when it comes to detecting non-zero areas in estimating parameter β, it failed to recover the levels of these areas. This phenomena is also observed when the ℓ_1 penalty is used with log-likelihood loss function. We generated a toy dataset for logistic regression to demonstrate the fact. In the figure, the left handside is a simulated parameter and the right handside is the coefficient estimate using ℓ_1 penalty. We can see that the ℓ_1 coefficient estimates do not correctly recover the accurate levels of the parameters.

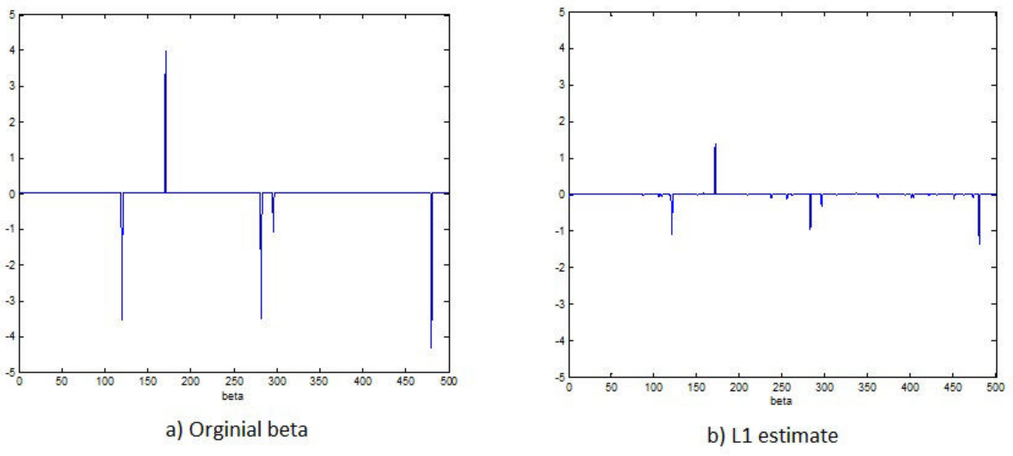


Figure 4.7: a) Original β and b) ℓ_1 estimates.

4.3.3 Cancer research data

Fused lasso penalty has been used successfully for hot-spot detection. In particular, detection of regions with gain or loss in DNA copies in the array-based comparative genome hybridization data (CGH). ArrayCGH is a very popular technique in bioinformatics to detect chromosomal abberations of the genes along the genome. These abberations come in the forms of gains or losses in the number of DNA copies of a gene. Array CGH data record the $\log_2$ ratio between the number of DNA copies of the genes in a tumor cell and reference cell. A positive value represents an amplification in the number of DNA copies and a negative value represents loss. These deletion or amplication in DNA copies are the hallmarks of cancer.

ArrayCGH data usually have a large number of features while the number of samples is small. Moreover, there is a particular structure of correlations between neighboring genes due to natural orderings. Most regression/classification methods for this type of high-dimensional data base on selecting a number of distinguished variables and might not be the right tool. Tibshirani proposed to use the fused lasso penalty for spatial smoothing and hot-spot detection for CGH data. [Rapaport et al., 2008] proposed a support vector machine method with fused lasso penalty to do classification on arrayCGH data and showed that with this proper penalty, classification methods can enhance their performance as opposed to the traditional lasso penalty. Fused lasso is devised for situations where the variables have some type of natural ordering. It not only encourages the sparsity of parameter estimates but also sparsity in the difference of neighboring coefficients.

In our study, we consider two publicly available data sets for cancer search: the bladder tumor and melanoma tumor data. The bladder tumor data contains arrayCGH profiles of 57 bladder tumor samples [Stransky et al., 2006]. Each profile contains the relative DNA counts for 2215 spots. We consider the classification problem of the tumor grade: with 12 tumors of Grade 1 and 45 tumors of higher grades (2 or 3). Tumor grade is a ranking system for how abnormal a tumor and its tissues appear to be. It is also an indicator of how fast the tumor will grow and spread. The ranking tends to differ for different types of cancer. In general, Grade 1 tumors appear to be very close to normal cells and tissues. These tumors tend to grow and spread slowly. Higher grade tumors are more abnormal and more likely to grow and spread fast.

The melanoma data set contains the arrayCGH profiles along 3750 spots of 78 melanoma tumors.35 of these tumors spread within 24 months and the other 43 tumors did not [Trolet et al., 2008]. We consider the problem of classifying based on arrayCGH data to determine whether a tumor will spread within 24 months. For the study, we use logistic regression. The two data sets are randomly splitted with two third of the data used for training and one third used for testing purpose. The objective function for fused lasso logistic regression with nonconvex penalty is:

$$\mathcal{Q}(\beta) = \sum_{i=1}^{n} -y_i(x_i^T\beta + c) + \log(1 + e^{x_i^T\beta + c}) + \mathcal{P}(|R\beta|; \lambda; \gamma), \tag{4.13}$$

Table 4.2: Comparison between fused lasso, lasso, SCAD, MCP, and structured SCAD on Melanoma data

Melanoma	Classification accuracy (%)	Standard error
Fused Lasso	51.92	11.07
Lasso	51.92	10.92
SCAD	51.53	6.58
MCP	51.15	8.7
Structured SCAD	74.23	7.70

where $\mathcal{P}$ is a nonconvex penalty SCAD or MCP. The strutural matrix $R \in \mathbb{R}^{7299\times 3650}$ is :

$$R = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots} \\ 0 & 1 & 0 & \dots & 1 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots} \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix}.$$

We compare the structured nonconvex penalty with several other penalties: fused lasso, lasso, SCAD, and MCP. The objective function for fused lasso logistic regression is:

$$\mathcal{L}(\beta) = \sum_{i=1}^{n} -y_i(x_i^T\beta + c) + \log(1 + e^{x_i^T\beta + c}) + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p-1} |\beta_{j+1} - \beta_j|. \quad (4.14)$$

For fused lasso, we use the procedure provided in [Liu et al., 2010a]. For lasso, we use GLMNET in Matlab. This function implements the coordinate descent method for penalized Generalized Linear Models (GLM) [Friedman et al., 2007]. For SCAD and MCP, we used NCVREG packge by [Breheny and Huang, 2011] which implements the coordinate descent method for nonconvex penalties. The experiments are repeated 10 times and the results are reported in the Table 3. Best parameters $\lambda_1, \lambda_2, \lambda$ are found using cross-validation. For nonconvex penalty, parameter $\gamma = 3.7$ is used as suggested by [Fan and Li, 2001]. We can see that non-convex fused lasso penalty outperforms other penalties in terms of prediction.

Table 4.3: Comparison between fused lasso, lasso, SCAD, MCP, and structured SCAD on Bladder data

Bladder	Classification accuracy (%)	Standard error
Fused Lasso	72.63	7.63
Lasso	76.32	7.94
SCAD	78.94	7.44
MCP	80.52	9.62
Structured SCAD	90.53	12.36

4.4 Conclusion

The alternating linearization method is a specialized nonsmooth optimization method for solving structured nonsmooth optimization problems. It combines the ideas of bundle methods and operator splitting methods, to define a descent algorithm in terms of the values of the function that is minimized. We have adapted the alternating linearization method to structured regularization problems by introducing the idea of diagonal quadratic approximation and developing specialized methods for solving subproblems. As a result, a new general method for a variety of regularization problems has been obtained, which has the following theoretical features:

- It deals with nonsmoothness directly, not via approximations,
- It is monotonic with respect to the values of the function that is minimized,
- Its convergence is guaranteed theoretically.

Our numerical experiments on a variety of structured regularization problems illustrate the applicability of the alternating linearization method and indicate its practically important virtues: speed, scalability, and accuracy. It clearly outperforms extant methods, and it can solve problems which were unsolvable otherwise.

Its efficacy and accuracy follow from the use of the diagonal quadratic approximation and from a special test, which chooses in an implicit way the best operator splitting step to be performed. The current approximate solution is updated only if it leads to a significant decrease of the value of the objective function. Its scalability is due to the use of highly specialized algorithms for solving its two subproblems. The algorithms do not require any explicit matrix formation or inversion, but only matrix–vector multiplications, and can be efficiently implemented with sparse data structures. Our study of image denoising and deblurring in section

3.6.4, as well as the narrative comprehension for children in section 3.6.5 are illustrations of broad applicability of the alternating linearization method.

In Chapter 4, we propose the structured nonconvex penalties and present an extension of the alternating linearization method to solve the problem. This problem was also considered in [Zhou and Wu, 2012], however the suggested algorithm requires the structured matrix to be full rank. Our method does not have this constraint. The extended version of ALIN is very simple to implement with robust performance. Our studies using synthetic and real data sets have shown that the structured nonconvex penalties are superior to their convex counterparts previously studied in the literature.

The simplicity and efficiency of the alternating linearization framework suggests that it can be extended to solve many problems that are currently of interests for the machine learning and data mining communities. One potential research direction is the problem of recovering and deblurring of f-MRI image. An f-MRI image is obtained from a partial $Fourier$ transform. Recently, a model is suggested to obtain a high quality restored image using total variation penalty and wavelet based penalty. The objective function is composed of three parts: a loss function, total variation penalty, and the wavelet based penalty. The total variation penalty has been used extensively in the image deblurring problem. Combined with the wavelet based penalty, the solution of the problem is not only a high quality restored image but also sparse in the wavelet domain. This formulation has been shown to have great advantages compared to previous models. However, with high dimensional images, the formulation is very challenging. In this dissertation, the framework is used for the problem of minimizing the composite function as sum of two convex functions. However, it is possible to extend the framework to deal with this problem with three component objective function.

Another interesting direction is the matrix regression problem with low rank conditions. This problem is currently studied rigorously due to its application in medical imaging. The study of children's brain imaging in chapter 3 is an instance of this problem. To make inference of a response variable y, the coefficient matrix β is assumed to have some intrinsic structures. The $3-$d fused lasso is a good penalty for this purpose since it assumes coefficients nearby should exhibit similar characteristics. However, in many cases, the structure does not have to be smooth, thus using a smoothing penalty like fused lasso will not yield satisfactory results.

We can assume the matrix of coefficients β to have low rank representation, thus it can be represented by the outer product of only a small number of components. This not only reduces the complexity of the model but also gives good insights into which parts of the coefficient matrix are important to building a good model. The objective function will be more complicated than the previous ones since it includes a term involving the rank of the matrix coefficient, which can be approximated by a convex surrogate.

The alternating linearization framework is a powerful tool in nonsmooth optimization. It is widely applicable in many problems in statistical machine learning and signal processing. Via this dissertation, it has been shown to be competitive against many state-of-the-art methods for these problems. Moreover, it has the potential to be extended to many problems that these methods will have trouble solving especially in high dimensional settings.

Bibliography

Afonso, Bioucas-Dias, and Figueiredo. Fast image recovery using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, 19(29, PAGES = 2345-2356,), 2010.

M. Annergren Y. Wang B. Wahlberg, S. Boyd. An admm algorithm for a class of total variation regularized estimation problems. *arXiv:1203.1828*, 2012.

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.

S Becker, J Bobin, and EJ CandÃ¨s. Nesta: a fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences*, 4(1), 2011.

E. G. Birgin and J. M. Martínez. Large-scale active-set box-constrained optimization method with spectral projected gradients. *Comput. Optim. Appl.*, 23(1):101–125, 2002.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

Patrick Breheny and Jian Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, page 232âŁ“253, 2011.

H. Jiang C. Li, W. Yin and Y. Zhang. An efficient augmented lagrangian method with applications to total variation minimization. *Computational Optimization and Applications*, 56(03), 2013.

X. Chen, S. Kim, Q. Lin, J. Carbonell, and E. Xing. Graph-structured multi-task regression and an efficient optimization method for general fused lasso. technical report 1005.3579v1, arXiv, 2010.

X. Chen, S. Kim, Q. Lin, J. Carbonell, and E. Xing. An efficient proximal gradient method for general structured sparse learning. technical report 1005.4717v3, arXiv, 2011.

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. technical report 0912.3522v4, arXiv, 2010.

S. Ma D. Goldfarb and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming.*, 141:349–382., 2013.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math*, 57(11):1413–1457, 2004.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–451, 2004.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Associations*, pages 1348–1360, 2001.

A. Friedlander and J. M. Martínez. On the maximization of a concave quadratic function with box constraints. *SIAM J. Optim.*, 4(1):177–192, 1994.

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

W. Fu. Penalized regressions: the bridge vs. the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.

T. Goldstein and S. Osher. The split Bregman method for $L1$-regularized problems. *SIAM J. Imaging Sci.*, 2(2):323–343, 2009.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

L. Grosenick, B. Klingenberg, B. Knutson, and J. Taylor. A family of interpretable multivariate models for regression and classification of whole-brain fmri data. 2011.

L. Grosenick, B. Klingenberg, K. Katovich, B. Knutson, and J. Taylor. Interpretable whole-brain prediction analysis with graphnet. *Neuroimage*, 2013.

B. He and X. Yuan. On the $O(1/t)$ convergence rate of alternating direction method. technical report, Optimization On-Line, 2011.

H. Hoefling. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4), 2010.

P. Karunanayaka, V. J. Schmithorst, J. Vannest, J. P. Szaflarski, E. Plante, and S. K. Holland. A group independent component analysis of covert verb generation in children: A functional magnetic resonance imaging study. *Neuroimage*, 51:472–487, 2010.

Seung-Jean Kim, M. Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING*, 1(4), 2007.

K. Kiwiel, C. Rosa, and A. Ruszczyński. Proximal decomposition via alternating linearization. *SIAM Journal on Optimization*, 9:153–172, 1999.

Kwangmoo Koh, Seung-Jean Kim, and Stephen Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research,*, 1(4), 2007.

J. Liu, J. Chen, and J. Ye. Large-scale sparse logistic regression. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 547–556, 2009.

J. Liu, L. Yuan, and J. Ye. An efficient algorithm for a class of fused lasso problems. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2010a.

J. Liu, L. Yuan, and J. Ye. An efficient algorithm for a class of fused lasso problems. *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2010b.

J. Liu, L. Yuan, and J. Ye. SLEP: Sparse learning with efficient projections. techincal report, Computer Science Center, Arizona State University, 2011.

Rahul Mazumder, Jerome H. Friedman, and Trevor Hastie. Sparsenet : Coordinate descent with non-convex penalties. *Journal of the American Statistical Association*, pages 1125–1138, 2011.

V. Michel, A. Gramfort, G. Varoquaux, E. Eger, and B. Thirion. Total variation regularization for fmri-based prediction of behavior. *IEEE Transactions on Medical Imaging*, 30(7):1328–1340, 2011.

Yu. Nesterov. Gradient methods for minimizing composite objective function. discussion paper 2007/76, CORE, 2007.

F. Rapaport, E. Barillot, and J. Vert. Classification of arraycgh data using fused svm. *Bioinformatics*, pages 375–382, 2008.

Saharon Rosset and Ji Zhu. Piecewise linear regularized solution paths. *Annals of Statistics*, pages 1012–1030, 2007.

A. Ruszczyński. On convergence of an augmented Lagrangian decomposition method for sparse convex optimization. *Math. Oper. Res.*, 20(3):634–656, 1995.

A. Ruszczyński. *Nonlinear optimization*. Princeton University Press, Princeton, NJ, 2006.

V. Schmithorst, S. Holland, and E. Plante. Cognitive modules utilized for narrative comprehension in children: a functional magnetic resonance imaging study. *Neuroimage*, 29:254–266, 2006.

Stransky, Vallot, Reyal, and Bernard-Pierrot. Regional copy number-independent deregulation of transcription in cancer. *Nature Genetics*, pages 1386–96, 2006.

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

R. Tibshirani and J. Taylor. The solution path of the generalized lasso. *Annals of Statistics*, 2011.

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

R. Tibshirani, M. Saunders, J. Zhu, and S. Rosset. Sparsity and smoothness via the fused lasso. *Journal of The Royal Statistical Society Series B*, 67:91–108, 2005.

Trolet, Hupe, Lebigot, and Decraene. Genomic proï¬'ling and identiï¬'cation of high-risk uveal melanoma by array cgh analysis of primary tumors and liver metastases. *Acta Ophthalmologica*, page 0, 2008.

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

P. Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117:387–423, 2007.

Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 57(07), 2009.

Tong Tong Wu and Kenneth Lange. Coordinate descent algorithms for lasso penalized regression. *Annals of Applied Statistics*, 2(205), 2008.

G.-B. Ye and X. Xie. Split Bregman method for large scale fused Lasso. *Comput. Statist. Data Anal.*, 55(4):1552–1569, 2011.

Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010.

D. Goldfarb Z. Qin and S. Ma. An alternating direction method for total variation denoising. *arXiv:1108.1587*.

Zhang. Tnearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, page 894âŁ“942, 2010.

Zhang and Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, page 1567âŁ“1594, 2008.

Z. Zhang, K. Lange, and C. Sabatti. Reconstructing dna copy number by joint segmentation of multiple sequences. *BMC Bioinformatics*, 13(205), 2012.

Peng Zhao and Bin Yu. Stagewise lasso. *Journal of Machine Learning Research*, page 2701âŁ“2726, 2007.

H. Zhou and Y. Wu. A generic path algorithm for regularized statistical estimation. *Journal of American Statistical Association*, 2012.

H. Zou and R. Li. One-step sparse estimates in nonconcave penalized likelihood models. *The Annals of Statistics*, page 1509âŁ“1533, 2008.