Convergence Analysis of the Approximate Proximal Splitting Method for Non-Smooth Convex Optimization

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Mojtaba Kadkhodaie Elyaderani

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Master of Science

Zhi-Quan Luo

May, 2014

Acknowledgements

I want to thank my advisor, Professor Luo, who provided me with the necessary guidance to allow me to complete this research project. I also thank my great friends, Meisam, Maziar and Morteza, who helped me during this period at grad school.

Abstract

Consider a class of convex minimization problems for which the objective function is the sum of a smooth convex function and a non-smooth convex regularity term. This class of problems includes several popular applications such as compressive sensing and sparse group LASSO. In this thesis, we introduce a general class of approximate proximal splitting (APS) methods for solving such minimization problems. Methods in the APS class include many well-known algorithms such as the proximal splitting method (PSM), the block coordinate descent method (BCD) and the approximate gradient projection methods for smooth convex optimization. We establish the linear convergence of APS methods under a local error bound assumption. Since the latter is known to hold for compressive sensing and sparse group LASSO problems, our analysis implies the linear convergence of the BCD method for these problems without strong convexity assumption.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Formulation

In this thesis, we study a class of algorithms for solving the constrained convex minimization problems of the form

$$\min_{\boldsymbol{x}\in X} \quad F(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}), \tag{1.1}$$

where $X \subseteq \mathbb{R}^n$ is a convex closed set, f_1 is a convex function (may be non-smooth) and f_2 is a smooth convex function with Lipschitz continuous gradient on X

$$\|\nabla f_2(\boldsymbol{x}) - \nabla f_2(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \ \ \forall \boldsymbol{x} \in X, \boldsymbol{y} \in X, \tag{1.2}$$

where L is a positive scalar and $\|\cdot\|$ denotes the usual Euclidean norm.

Non-smooth convex optimization problems of the form (1.1) arise in many contemporary statistical and signal processing applications including compressive sensing, signal denoising and sparse logistic regression. In the sequel, we outline some of the most recent applications of problem (1.1).

1.1.1 LASSO Problem

Suppose that we have a noisy observation vector $\boldsymbol{b} \in \mathbb{R}^m$ about an unknown sparse vector $\boldsymbol{x} \in \mathbb{R}^n$, where the signal model is linear and given by

$$\boldsymbol{b} \approx \boldsymbol{A}\boldsymbol{x},$$

for some given matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. One of the most popular techniques to estimate the sparse vector $\boldsymbol{x}$ is called LASSO [1]. LASSO can be viewed as an ℓ_1-norm regularized linear least squares problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2 + \lambda\|\boldsymbol{x}\|_1, \tag{1.3}$$

where the first term $\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2$ reduces the estimation error, and the second term $\lambda\|\boldsymbol{x}\|_1$ promotes the sparsity of the solution. The parameter λ controls the sparsity level of the solution. The higher λ is, the fewer non-zero entries would be in the LASSO solution. Clearly, by setting $f_2(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2$, $f_1(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1$ and $X = \mathbb{R}^n$, problem (1.3) becomes a special case of problem (1.1).

1.1.2 Group LASSO Problem

In many regression problems, the goal is to find important explanatory factors in predicting the response variable, where each explanatory factor may be represented by a (predefined) group of input variables [2]. This idea extends LASSO which is designed to select individual input variables as the explanatory factors.

Consider the linear regression problem

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}, \tag{1.4}$$

where $\boldsymbol{b} \in \mathbb{R}^m$ is the vector of response variables, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 I_m)$ is the error vector, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is the design matrix and $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of regression coefficients. Assume that $\boldsymbol{A}$ has a block structure, i.e. $\boldsymbol{A} = (\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_J)$ with each $\boldsymbol{A}_j \in \mathbb{R}^{m\times n_j}$, $j = 1, \cdots, J$, and $\sum_{j=1}^J n_j = n$. Then the coefficients vector $\boldsymbol{x}$ can be respectively factorized as $\boldsymbol{x} = (\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \cdots, \boldsymbol{x}_J^T)^T$ with each $\boldsymbol{x}_j \in \mathbb{R}^{n_j}$, $j = 1, \cdots, J$.

The group LASSO problem is to find the best representation of $\boldsymbol{b}$ in terms of the factors $\boldsymbol{A}_j$ and can be formulated as the following optimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2 + \sum_{j=1}^{J} w_j\|\boldsymbol{x}_j\|, \tag{1.5}$$

where w_j is the sparsity weight of block j. Notice that the second term in the objective induces sparsity at the factor level. Setting $f_2(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2$ and $f_1(\boldsymbol{x}) = \sum_{J\in\mathcal{J}} w_J\|\boldsymbol{x}_J\|$, the Group LASSO problem (1.5) follows the structure of problem (1.1).

1.1.3 Group LASSO for Logistic Regression

Given a set of n-dimensional feature vectors $\boldsymbol{a}_i$, $i = 1,\cdots,m$, and the corresponding class labels $b_i \in \{0,1\}$, $i = 1,\cdots,m$, our task is to find a linear classifier for the vectors $\boldsymbol{a}_i$. Assume the probability distribution of the class label b, given a feature vector $\boldsymbol{a}$ is given by

$$p(b=1|\boldsymbol{a};\boldsymbol{x}) = \frac{\exp(\boldsymbol{a}^T\boldsymbol{x})}{1+\exp(\boldsymbol{a}^T\boldsymbol{x})},$$

where $\boldsymbol{x}$ is the logistic coefficient vector. The logistic Group LASSO problem [3] can be written as

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \sum_{i=1}^{m}(\log(1+\exp(\boldsymbol{a}_i^T\boldsymbol{x})) - b_i\boldsymbol{a}_i^T\boldsymbol{x}) + \sum_{J\in\mathcal{J}} w_J\|\boldsymbol{x}_J\|, \tag{1.6}$$

where w_J is the sparsity weight for the corresponding block $\boldsymbol{x}_J$. Problem (1.6) can also be interpreted as a special form of problem (1.1). We refer the readers to [4, 5] for further applications of Group LASSO, and to [3, 6–10] for further studies on Group LASSO type of techniques in statistical problems.

These three examples, among many others, enjoy the composite objective structure of problem (1.1). Since the arising of these important problems, there has been a lot of effort to design algorithms that can solve them. In the next chapter, we will review some of the well-known first-order algorithms for solving problem (1.1) (or some special

cases of it). Later in chapter 3, we will define a broad class of first order algorithms which includes all of the algorithms reviewed in chapter 2 as special cases. We will then analyze the convergence rate of this algorithm and prove that under some special conditions of the problem, one can expect a linear rate of convergence for this algorithm. The important feature of our analysis is that it does not require any non-degeneracy assumption on the problem, i.e. no strong convexity assumption of the objective is needed.

Chapter 2

Proximal Splitting Methods

In this chapter, we review some of the well-known algorithms for solving problem (1.1) (or some of its specific examples). The reviewed algorithms all lie within the general framework of first order algorithms. Generally speaking, in each iteration of a first order algorithm, only gradients (or sub-gradients) of the objective function, evaluated at the current and past iterates, are available. First order algorithms benefit from having cheap iterative updates and thus are very popular for solving large scale optimization problems recently.

2.1 Gradient Projection Method

If we assume $X = \mathbb{R}^n$ and $f_1(\cdot)$ to be the indicator function, $\iota_{\boldsymbol{C}}(\cdot)$, of a closed convex set $\boldsymbol{C}$,

$$\iota_{\boldsymbol{C}}(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \boldsymbol{C} \\ +\infty & \text{otherwise,} \end{cases}$$

then the problem in (1.1) turns out to be the smooth minimization of $F(\cdot) = f_2(\cdot)$ over the set $\boldsymbol{C}$

$$\min_{\boldsymbol{x} \in \boldsymbol{C}} f_2(\boldsymbol{x}). \tag{2.1}$$

The optimal points of this convex problem should satisfy the following equation

$$\boldsymbol{x}^* = \mathrm{Proj}_{\boldsymbol{C}}[\boldsymbol{x}^* - \nabla f_2(\boldsymbol{x}^*)], \tag{2.2}$$

where $\text{Proj}_C[\cdot]$ denotes the orthogonal projection into the set C and operates, on an arbitrary point $\boldsymbol{x} \in \mathbb{R}^n$, as

$$\text{Proj}_{\boldsymbol{C}}[\boldsymbol{x}] = \arg\min_{\boldsymbol{y} \in \boldsymbol{C}} \ \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2.$$

It is easy to see that equation (2.2) is a compact way of writing the first-order optimality condition of problem (2.1) at the point $\boldsymbol{x}^*$.

The well-known approach to solve (2.1) is called Gradient Projection (GP) [11, 12]. In every iteration k of the GP method, we take a gradient step of size α_k and then project the point back into the feasible set $\boldsymbol{C}$,

$$\boldsymbol{x}^{k+1} = \text{Proj}_{\boldsymbol{C}}[\boldsymbol{x}^k - \alpha_k \nabla f_2(\boldsymbol{x}^k)]. \tag{2.3}$$

This update rule is naturally suggested to solve the optimality condition (2.2). In general, projection to a convex set is not an easy problem. Hence, the efficiency of the GP algorithm highly depends on the simplicity of projection into the set $\boldsymbol{C}$, and therefore, it relies on the structure of $\boldsymbol{C}$. For instance, if $\boldsymbol{C}$ is the non-negative orthant, then projection to $\boldsymbol{C}$ decomposes over the elements of $\boldsymbol{x}$, and thus is easy to handle.

2.2 Proximal Splitting Method

The counterpart of the GP algorithm for the general non-smooth problem (1.1) is the so called *Proximal Splitting Method* (PSM). In order to introduce this method, we first need to define the proximity operator.

2.2.1 Proximity Operator

Definition 1 *For any convex function $\varphi(\cdot)$ (possibly non-smooth), the Moreau-Yashida proximity operator* $\text{prox}_{\varphi}(\cdot, X) : \ \mathbb{R}^n \to \mathbb{R}^n$ *is defined as*

$$\text{prox}_{\varphi}(\boldsymbol{x}, X) = \arg\min_{\boldsymbol{y} \in X} \varphi(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2. \tag{2.4}$$

Note that since $\frac{1}{2}\|\cdot - \boldsymbol{x}\|^2$ is strongly convex and $\varphi(\cdot)$ is convex, the minimizer of (2.4) is unique. Furthermore, if the function φ is chosen to be the indicator function,

$\iota_{\boldsymbol{C}}$, of the closed convex set C and $X = \mathbb{R}^n$, then proximity operator reduces to the projection operator into the set C since

$$\begin{aligned}\mathrm{prox}_{\varphi}(\boldsymbol{x}, X) &= \arg\min_{\boldsymbol{y}\in X} \ \iota_{\boldsymbol{C}}(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{x}\|^2 \\ &= \arg\min_{\boldsymbol{y}\in \mathbf{C}} \ \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{x}\|^2 \\ &= \mathrm{proj}_{\boldsymbol{C}}[\boldsymbol{x}].\end{aligned}$$

Thus, the proximity operator can be viewed as a natural extension of the projection operator. In the sequel, we will denote the proximity operator by $\mathrm{prox}_{\varphi}(\cdot)$ for the sake of conciseness and assume that its dependence on the set X is understood from the context.

The proximity operator inherits many useful properties of the projection operator into convex sets. As an instance, it is known to be non-expansive and therefore Lipschitz.

Proposition 1 *Assume $\phi(\cdot)$ is a convex function. Then we have*

$$\|\mathrm{prox}_{\varphi}(\boldsymbol{x}_1) - \mathrm{prox}_{\varphi}(\boldsymbol{x}_2)\| \leq \|\boldsymbol{x}-\boldsymbol{y}\|, \ \forall \ \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n.$$

Proof Writing the optimality condition of the problem

$$\min_{\boldsymbol{y}\in\boldsymbol{C}} \ \phi(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{x}\|^2,$$

for $\boldsymbol{x} = \boldsymbol{x}_1$ and $\boldsymbol{x}_2$ we obtain

$$\begin{aligned}\mathrm{prox}_{\varphi}(\boldsymbol{x}_1) - \boldsymbol{x}_1 + \boldsymbol{g}_1 = \mathbf{0}, \ \text{ for some } \boldsymbol{g}_1 \in \partial f(\mathrm{prox}_{\varphi}(\boldsymbol{x}_1)), \\ \mathrm{prox}_{\varphi}(\boldsymbol{x}_2) - \boldsymbol{x}_2 + \boldsymbol{g}_2 = \mathbf{0}, \ \text{ for some } \boldsymbol{g}_2 \in \partial f(\mathrm{prox}_{\varphi}(\boldsymbol{x}_2)).\end{aligned}$$

Therefore, we have

$$\begin{aligned}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 &= \|\mathrm{prox}_{\varphi}(\boldsymbol{x}_1) - \mathrm{prox}_{\varphi}(\boldsymbol{x}_2) + \boldsymbol{g}_1 - \boldsymbol{g}_2\|^2 \\ &= \|\mathrm{prox}_{\varphi}(\boldsymbol{x}_1) - \mathrm{prox}_{\varphi}(\boldsymbol{x}_2)\|^2 \\ &+ 2\langle \mathrm{prox}_{\varphi}(\boldsymbol{x}_1) - \mathrm{prox}_{\varphi}(\boldsymbol{x}_2), \boldsymbol{g}_1 - \boldsymbol{g}_2\rangle + \|\boldsymbol{g}_1 - \boldsymbol{g}_2\|^2 \\ &\geq \|\mathrm{prox}_{\varphi}(\boldsymbol{x}_1) - \mathrm{prox}_{\varphi}(\boldsymbol{x}_2)\|^2,\end{aligned}$$

where the last step is due to $\langle \mathrm{prox}_{\varphi}(\boldsymbol{x}_1) - \mathrm{prox}_{\varphi}(\boldsymbol{x}_2), \boldsymbol{g}_1 - \boldsymbol{g}_2\rangle \geq 0$ (by the convexity of ϕ). This completes the proof.

In large scale problems, it is not always easy to compute the proximity operator, unless the function φ has some special structure, such as separability. In those cases the proximity operator is efficiently computable (or has closed form). For instance, if the function φ is the ℓ_1-norm, the proximity operator has a closed form solution, also known as the Shrinkage and thresholding operator [13].

The optimality condition of problem (1.1) can be formulated using the proximity operator. This fact is proved in the following proposition.

Proposition 2 *The point $\boldsymbol{x}^*$ is a minimizer of the problem* (1.1) *if and only if*

$$\boldsymbol{x}^* = \mathrm{prox}_{\alpha f_1}(\boldsymbol{x}^* - \alpha \nabla f_2(\boldsymbol{x}^*)), \tag{2.5}$$

for some $\alpha > 0$.

Proof Due to the convexity of the problem, $\boldsymbol{x}^*$ is an optimal point if and only if

$$\nabla f_2(\boldsymbol{x}^*) \in -\partial f_1(\boldsymbol{x}^*), \tag{2.6}$$

where $\partial f_1(\boldsymbol{x})$ is the sub-differential set of the function f_1 evaluated at the point $\boldsymbol{x}$. By the definition of the proximity operator, (2.5) is equivalent to

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{y}\in X} \ \alpha f_1(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{y} - (\boldsymbol{x}^* - \alpha \nabla f_2(\boldsymbol{x}^*)\|^2.$$

Differentiating with respect to $\boldsymbol{y}$ yields the optimality condition

$$\boldsymbol{x}^* - (\boldsymbol{x}^* - \alpha \nabla f_2(\boldsymbol{x}^*)) \in -\alpha \partial f_1(\boldsymbol{x}^*),$$

which is equivalent to (2.6). This completes the proof.

The Proximal Splitting Method (PSM) can be viewed as an iterative approach to solve the fixed point equation (2.5)

$$\boldsymbol{x}^{k+1} = \text{prox}_{\alpha_k f_1}(\boldsymbol{x}^k - \alpha_k \nabla f_2(\boldsymbol{x}^k)), \tag{2.7}$$

where $\alpha_k > 0$ determines the step size at iteration k. Note that PSM is identical to the GP algorithm if $f_1 = \iota_C$ for some convex closed set C.

2.2.2 Proximal Gradient Vector

Another basic concept which is often useful in analyzing PSM (or its variants) is the concept of proximal gradient.

Definition 2 *For any $\alpha > 0$, we define proximal gradient vector as*

$$\tilde{\nabla} F(\boldsymbol{x}, \alpha) = \frac{1}{\alpha}[\boldsymbol{x} - \text{prox}_{\alpha f_1}(\boldsymbol{x} - \alpha \nabla f_2(\boldsymbol{x}))]. \tag{2.8}$$

When $\alpha = 1$, we will use the short notation

$$\tilde{\nabla} F(\boldsymbol{x}) = \tilde{\nabla} F(\boldsymbol{x}, \alpha). \tag{2.9}$$

Note that in the special case of $f_1 = 0$ and $X = \mathbb{R}^n$, the proximal gradient reduces to the standard gradient, namely, $\tilde{\nabla} F(\boldsymbol{x}, \alpha) = \nabla f_2(\boldsymbol{x}) = \nabla F(\boldsymbol{x})$. In another special case where $f_1 = \iota_C$ (the indicator function of a convex set C), we have

$$\tilde{\nabla} F(\boldsymbol{x}, \alpha) = \frac{1}{\alpha}[\boldsymbol{x} - \text{proj}_C(\boldsymbol{x} - \alpha \nabla f_2(\boldsymbol{x}))], \tag{2.10}$$

which is the residual of the optimality condition for the following problem

$$\min_{\boldsymbol{x} \in C} f_2(\boldsymbol{x}). \tag{2.11}$$

Hence, $\tilde{\nabla} F(\boldsymbol{x}, \alpha)$ can be viewed as a generalized notion of gradient for the constrained non-smooth minimization. In addition, it inherits many useful properties of gradient. For instance, $\tilde{\nabla} F(\boldsymbol{x}^*, \alpha) = 0$ for some $\alpha > 0$ iff $\boldsymbol{x}^*$ is an optimal solution of (1.1) as shown in proposition 2.

The optimality condition for (1.1), given by (2.5), suggests that we can define a local measure for the distance to optimality by

$$\psi(\boldsymbol{x}) = \| \tilde{\nabla} F(\boldsymbol{x}) \| = \|\boldsymbol{x} - \text{prox}_{f_1}(\boldsymbol{x} - \nabla f_2(\boldsymbol{x}))\|. \tag{2.12}$$

It is easy to see that $\psi(\boldsymbol{x}) = 0$ iff $\boldsymbol{x}$ belongs to the set of optimal solutions of (1.1), which we denote by X^*.

2.3 Convergence Analysis

In this section, we briefly review the established convergence analysis of the GP and PSM methods.

2.3.1 Convergence Analysis of GP

The convergence analysis of the GP method has been studied before [14]. It has been shown that such analysis can be generalized to approximate versions of the GP method [14–18] which are also known as Approximate Gradient Projection (AGP) methods. In the framework of AGP, an error is allowed in the computation of gradient, as far as the size of the error vector is sufficiently small. Therefore, the update in such an algorithm can be formulated as

$$\boldsymbol{x}^{k+1} = \text{Proj}_{\boldsymbol{C}}[\boldsymbol{x}^k - \alpha_k \nabla f_2(\boldsymbol{x}^k) + \boldsymbol{e}_k], \tag{2.13}$$

where $\boldsymbol{e}_k \in \mathbb{R}^n$ is the error vector. It has been shown [14] that many well-known algorithms such as Matrix Splitting Method [19] and Extragradient Method [20] lie within this framework (with their size of error to be bounded by $\|\boldsymbol{e}_k\| \leq \kappa \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|$ for some $\kappa > 0$).

The key to the convergence analysis of AGP in [14] lies in a certain error bound which estimates the distance to the optimal solution set $\boldsymbol{C}^*$ from an $\boldsymbol{x} \in \boldsymbol{C}$ near $\boldsymbol{C}^*$ by

the norm of the residual

$$\boldsymbol{x} - \text{proj}_{\boldsymbol{C}}[\boldsymbol{x} - \nabla f_2(\boldsymbol{x})].$$

By using this error bound, it can be shown that the sequence of AGP iterates (2.13) converges linearly to an optimal point. As we are going to extend this convergence analysis approach for non-smooth optimization, we will revisit the AGP formulation (2.13) again (see Section 3.1).

2.3.2 Convergence Analysis of PSM

It is known that if the step size α_k satisfies

$$0 < \underline{\alpha} \leq \alpha_k \leq \bar{\alpha} < 1/L, \ k = 0, 1, \cdots$$

for L being the Lipschitz constant of ∇f_2, then every sequence generated by PSM converges to a solution of (1.1) (see [21]).

In spite of this convergence result, the rate of convergence for PSM is not known, except in some specific cases. For instance, if $f_1 = \iota_C$ and f_2 has a composite structure ($f_2(x) = h(\boldsymbol{A}\boldsymbol{x})$, where h is strongly convex and $\boldsymbol{A}$ is an $m \times n$ matrix which is not necessarily full column rank), then it is proved by Luo and Tseng [15] that the PSM algorithm (which coincides with GP in this case) converges linearly to an optimal solution of (1.1). This result is significant due to the fact that it establishes linear convergence in the absence of strong convexity. This result has been recently extended to the case where, $f_1(x) = \sum_{J \in \mathcal{J}} w_J \|\boldsymbol{x}_J\|_2$ or $f_1(x) = \sum_{J \in \mathcal{J}} w_J \|\boldsymbol{x}_J\|_2 + \lambda \|\boldsymbol{x}\|$ and $f_2 = h(\boldsymbol{A}\boldsymbol{x})$ is still a composite function (see [13, 22]). The analysis is again based on the notion of local error bound.

2.3.3 Error Bounds

In this section we formally introduce the notion of error bound. As we will see it is a vital property in obtaining linear convergence rate for solving a problem via first-order methods.

For any $x \in X$, we can define

$$\varphi(\boldsymbol{x}) = \min_{\boldsymbol{y} \in \bar{X}^*} \|\boldsymbol{x} - \boldsymbol{y}\|, \tag{2.14}$$

where $\bar{X}^*$ is the closure of X^* (the set of optimal solutions of (1.1)). It is straightforward to see that $\varphi(\boldsymbol{x})$ can be used as a measure for distance to optimality, and $\varphi(\boldsymbol{x}) = 0$ iff $\boldsymbol{x} \in \bar{X}^*$. However, in practice it is impossible to compute $\varphi(\boldsymbol{x})$, due to the requirement of knowing the set of optimal solutions, $\bar{X}^*$. This is where the error bound comes into the picture. It serves as an approximated measure of the distance to optimality. The error bound is simply a bound on $\varphi(\boldsymbol{x})$, based on another measure of optimality that can be computed easily (in this case, the size of the residual $\psi(\boldsymbol{x})$ defined by (2.12)).

Definition 3 *Consider the optimality distance measures defined by* (2.12) *and* (2.14). *We say that problem* (1.1) *satisfies the local error bound property if for every* $\nu \geq \inf_{\boldsymbol{x} \in X} F(\boldsymbol{x})$, *there exist scalars* $\delta > 0$ *and* $\tau > 0$ *such that*

$$\varphi(\boldsymbol{x}) \leq \tau \psi(\boldsymbol{x}), \tag{2.15}$$

for all $\boldsymbol{x} \in X$ *with* $F(\boldsymbol{x}) \leq \nu$ *and* $\psi(\boldsymbol{x}) \leq \delta$.

In other words, (2.15) says that $\varphi(\boldsymbol{x})$ is bounded above by the norm of the residual ψ at x, whenever $F(\boldsymbol{x})$ is bounded above and this residual is small enough. In order to gain some insight on when (2.15) holds, consider the case where $X = \mathbb{R}^n$ and $F(x) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x}$ for some Positive definite matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b} \in \mathbb{R}^n$. Then, (2.15) is equivalent to

$$\varphi(\boldsymbol{x}) \leq \tau \|\nabla F(\boldsymbol{x})\| = \tau \|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\|,$$

which can be easily checked to be true (using elementary linear algebra). Furthermore, it holds for strongly convex smooth $F(\boldsymbol{x})$, when $X = \mathbb{R}^n$. Notice that for any strongly convex smooth function $F(\boldsymbol{x})$, there exists a $\tau > 0$ such that

$$\|\boldsymbol{x} - \boldsymbol{y}\|^2 \leq \tau \langle \nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \ \forall \ \boldsymbol{x}, \boldsymbol{y}.$$

Let $\hat{\boldsymbol{x}}$ to be a stationary point in $\bar{X}^*$ satisfying $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\| = \varphi(\boldsymbol{x})$, then

$$\varphi(\boldsymbol{x})^2 \leq \tau \langle \nabla F(\boldsymbol{x}), \boldsymbol{x} - \hat{\boldsymbol{x}} \rangle \leq \tau \|\nabla F(\boldsymbol{x})\| \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|.$$

Canceling the "$\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|$" term on both sides obtains the bound (2.15).

Proving error bound for different problems has a long history in the literature. It was first considered by Demb and Tulowizki [23] for strongly convex quadratic functions and by Pang [24] in the context of Linear Complementarity Problems (LCP) satisfying a certain regularity condition.

In the case of smooth minimization, there are various results on different cases in which (2.15) holds true. For instance, it has been proven for strongly convex functions in [25], and for quadratic functions with polyhedral constraint in [26], [15]. The error bound condition also holds if f_2 has a composite structure as

$$f_2(\boldsymbol{x}) = h(\boldsymbol{A}\boldsymbol{x}) + \langle \boldsymbol{q}, \boldsymbol{x} \rangle \ \ \forall \boldsymbol{x},$$

where $\boldsymbol{A}$ is an $m \times n$ matrix with no zero column, $\boldsymbol{q}$ is a vector in $\mathbb{R}^n$, h is a strongly convex differentiable function in $\mathbb{R}^m$ with ∇h Lipschitz continuous in $\mathbb{R}^m$ and X is a polyhedral set [15].

In the case of non-smooth optimization, the results are more restricted. It is mainly due to the difficulties which arise in dealing with the non-smooth part. Hence, these results can only handle structured non-smooth parts. For instance, in the recent works [13, 22], it has been proved that error bounds holds for special type of non-smooth problems (Group LASSO type of problems). As we are going to use these results, we summarize them in the following theorem which is taken from [22].

Theorem 1 *In problem* (1.1) *let* $X = \mathbb{R}^n$, $f_1(\boldsymbol{x}) = \sum_{J \in \mathcal{J}} w_J \|\boldsymbol{x}_J\| + \lambda \|\boldsymbol{x}\|$ *for non-negative* w_J*'s and* λ *and* $f_2(\boldsymbol{x}) = h(\boldsymbol{A}\boldsymbol{x})$ *for some strongly convex smooth function* $h : \mathbb{R}^m \to \mathbb{R}$ *and an* $m \times n$ *matrix* $\boldsymbol{A}$*. In addition if the function* F *is coercive, then error bound condition* (2.15) *holds for problem* (1.1)*.*

The direct consequence of this theorem is that the error bound condition holds true for all the examples mentioned in Chapter 1.

Corollary 1 *The error bound condition* (2.15) *holds for LASSO problem* (1.3)*, Group LASSO problem* (1.5) *and logistic Group LASSO problem* (1.6)*.*

Proof It is clear that LASSO and Group LASSO problems both have the structure assumed in Theorem 1 (with $h(\boldsymbol{u}) = \|\boldsymbol{u} - \boldsymbol{b}\|^2$). Therefore, the error bound condition holds true in their cases. For the logistic regression problem, we should set

$$h(\boldsymbol{u}) = \sum_{i=1}^{m} (\log(1 + \exp(u_i)) - b_i u_i)$$

which is strongly convex in $\boldsymbol{u}$. Therefore, this problem also satisfies the local error bound condition.

Later in chapter 3, we will utilize this result to establish the linear convergence rate of the APS class of methods when applied to solving these problems .

Chapter 3

Approximate Proximal Splitting Method

In this chapter we formally introduce the Approximate Proximal Splitting (APS) class of methods. As we will see in the sequel, APS includes the algorithms we reviewed in chapter 2, among many others, as special cases. After introducing this class of algorithms, we will analyze its convergence rate using the concept of local error bound.

3.1 Approximate Proximal Splitting Method

Definition 4 *An algorithm is considered in the class of APS methods if it generates a sequence of iterates* $\boldsymbol{x}^0, \boldsymbol{x}^1, \cdots$ *in* X *such that*

$$\boldsymbol{x}^{r+1} = \text{prox}_{\alpha^r f_1}(\boldsymbol{x}^r - \alpha^r(\nabla f_2(\boldsymbol{x}^r) + \boldsymbol{e}^r)),\ r = 0, 1, \cdots\ , \tag{3.1}$$

where $\{\alpha^r\}$ *is a sequence of positive scalars with* $\liminf \alpha^r > 0$ *and* $\{\boldsymbol{e}^r\}$ *is a sequence in* $\mathbb{R}^n$ *with*

$$\|\boldsymbol{e}^r\| \leq \kappa \|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\|, \tag{3.2}$$

for some non-negative scalar κ*.*

In equation (3.1), α^r and $\boldsymbol{e}^r$ may depend on $\boldsymbol{x}^r$ and can be viewed as algorithm parameters. Hence, different choices of α^r and $\boldsymbol{e}^r$ lead into different algorithms. For instance, the PSM algorithm whose update rule is given by (2.7), is a special case of the APS algorithm with $\boldsymbol{e}^r = \mathbf{0}$. In fact, the condition (3.2) ensures that the algorithm does not deviate too much from the PSM update.

For smooth minimization which is a special case of problem (1.1) with $f_1 = \iota_C$, the AGP class of algorithms is very common. Since the proximity operator reduces to the projection operator in this case, the APS algorithm contains the APG method as a special case. Later in chapter 4 we will see how the Block Coordinate Decent (BCD) algorithm is also a special case of the APS method.

3.2 Linear Convergence of APS

In this section we prove that any sequence generated by the iterations (3.1)-(3.2), converges at least linearly to an optimal point of problem (1.1), if the following properties hold true.

- **Sufficient Decrease:** There exists a constant $c_1 > 0$ such that,

$$F(\boldsymbol{x}^r) - F(\boldsymbol{x}^{r+1}) \geq c_1 \|\boldsymbol{x}^{r+1} - \boldsymbol{x}^r\|^2, \ \forall \ r. \tag{3.3}$$

- **Local Error Bound**: For every $\nu \geq \inf_{\boldsymbol{x} \in X} F(\boldsymbol{x})$, there exist scalars $\delta > 0$ and $\tau > 0$ such that

$$\varphi(\boldsymbol{x}) \leq \tau \psi(\boldsymbol{x}), \tag{3.4}$$

  for all $\boldsymbol{x} \in X$ with $F(\boldsymbol{x}) \leq \nu$ and $\psi(\boldsymbol{x}) \leq \delta$.

- **Cost-to-go:** There exists a $c_2 > 0$, such that

$$F(\boldsymbol{x}^r) - F^* \leq c_2(\varphi(\boldsymbol{x}^r)^2 + \|\boldsymbol{x}^{r+1} - \boldsymbol{x}^r\|^2), \ \forall \ r, \tag{3.5}$$

  where F^* is the optimal objective value of (1.1).

Among these three conditions, the sufficient decrease property can be shown to hold for a sequence generated by (3.1)-(3.2) whenever the step size α_r is sufficiently small. Also, we will prove that the cost-to-go property is a direct consequence of the sufficient decrease condition.

On the other hand, the local error bound property solely depends on the optimization problem. Therefore the problem structure needs to be studied to ensure this property holds. As we discussed in the previous chapter, this condition has been established for certain classes of optimization problems, see [14], [22], [27] and references therein. Some of these existing results were summarized in Theorem 1 of chapter 2.

The rest of the section proceeds as follows. Assuming the sufficient decrease condition, we first prove that the cost-to-go will naturally follow for the APS class of algorithms. Then, the sufficient decrease is proved under some assumptions on the step size α^r and the error vector $\boldsymbol{e}^r$. Finally, the linear convergence rate of APS method is shown assuming the local error bound condition of the problem.

The following Lemma proves the cost-to-go property assuming the sufficient decrease condition.

Lemma 1 *If an APS method satisfies the sufficient decrease condition* (3.3)*, then the cost-to-go condition* (3.5) *will follow.*

Proof Set $\hat{\boldsymbol{x}}^r$ to be the point in $\bar{X}^*$, such that $\varphi(\boldsymbol{x}^r) = \|\boldsymbol{x}^r - \hat{\boldsymbol{x}}^r\|$. The optimality condition of $\boldsymbol{x}^{r+1}$ implies

$$\begin{aligned} &f_1(\hat{\boldsymbol{x}}^r) + \langle \nabla f_2(\boldsymbol{x}^r) + \boldsymbol{e}^r, \hat{\boldsymbol{x}}^r - \boldsymbol{x}^r \rangle + \frac{1}{2\alpha_r} \parallel \hat{\boldsymbol{x}}^r - \boldsymbol{x}^r \parallel^2 \quad \geq \\ &f_1(\boldsymbol{x}^{r+1}) + \langle \nabla f_2(\boldsymbol{x}^r) + \boldsymbol{e}^r, \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \rangle + \frac{1}{2\alpha_r} \parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel^2 \end{aligned}$$

This implies

$$\langle \nabla f_2(\boldsymbol{x}^r) + \boldsymbol{e}^r, \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \rangle + f_1(\boldsymbol{x}^{r+1}) - f_1(\hat{\boldsymbol{x}}^r) \leq \frac{1}{2\alpha_r} \varphi^2(\boldsymbol{x}^r). \tag{3.6}$$

Also, the mean value theorem implies

$$f_2(\boldsymbol{x}^{r+1}) - f_2(\hat{\boldsymbol{x}}^r) = \langle \nabla f_2(\boldsymbol{\xi}^r), \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \rangle, \tag{3.7}$$

for some $\boldsymbol{\xi}^r$ in the line segment joining $\boldsymbol{x}^{r+1}$ and $\hat{\boldsymbol{x}}^r$. Combining the above two relations yields

$$\begin{aligned}
F(\boldsymbol{x}^{r+1}) - F(\hat{\boldsymbol{x}}^r) &= f_1(\boldsymbol{x}^{r+1}) - f_1(\hat{\boldsymbol{x}}^r) + f_2(\boldsymbol{x}^{r+1}) - f_2(\hat{\boldsymbol{x}}^r) \\
&= \langle \nabla f_2(\boldsymbol{\xi}^r), \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \rangle + f_1(\boldsymbol{x}^{r+1}) - f_1(\hat{\boldsymbol{x}}^r) \\
&= \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \rangle + \langle \nabla f_2(\boldsymbol{\xi}^r) - \nabla f_2(\boldsymbol{x}^r), \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \rangle + f_1(\boldsymbol{x}^{r+1}) - f_1(\hat{\boldsymbol{x}}^r) \\
&\leq \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \rangle + L \parallel \boldsymbol{\xi}^r - \boldsymbol{x}^r \parallel \parallel \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \parallel + f_1(\boldsymbol{x}^{r+1}) - f_1(\hat{\boldsymbol{x}}^r) \\
&\leq \frac{1}{2\alpha_r}\varphi^2(\boldsymbol{x}^r) - \langle \boldsymbol{e}^r, \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \rangle + L \parallel \boldsymbol{\xi}^r - \boldsymbol{x}^r \parallel \parallel \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \parallel \\
&\leq \frac{1}{2\alpha_r}\varphi^2(\boldsymbol{x}^r) + L \parallel \boldsymbol{\xi}^r - \boldsymbol{x}^r \parallel \parallel \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \parallel + \kappa \parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel \parallel \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \parallel
\end{aligned} \tag{3.8}$$

where the first inequality is due to the Lipschitz continuity of ∇f_2, the second inequality is implied by (3.6) and the last inequality follows from the triangular inequality. It remains to bound the last two terms in (3.8). Using the fact that $\boldsymbol{\xi}^r$ lies in the line segment joining $\boldsymbol{x}^{r+1}$ and $\hat{\boldsymbol{x}}^r$, it follows that

$$\begin{aligned}
\parallel \boldsymbol{\xi}^r - \boldsymbol{x}^r \parallel \parallel \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \parallel &\leq (\parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel + \parallel \boldsymbol{x}^r - \hat{\boldsymbol{x}}^r \parallel)(\parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel + \parallel \boldsymbol{x}^r - \hat{\boldsymbol{x}}^r \parallel) \\
&= (\parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel + \varphi(\boldsymbol{x}^r))(\parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel + \varphi(\boldsymbol{x}^r)) \\
&\leq 2(\parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel^2 + \varphi^2(\boldsymbol{x}^r)).
\end{aligned}$$

For the last term in (3.8) we have,

$$\begin{aligned}
\parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel \parallel \boldsymbol{x}^{r+1} - \hat{\boldsymbol{x}}^r \parallel &\leq \parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel (\parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel + \varphi(\boldsymbol{x}^r))) \\
&\leq 2 \parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel^2 + 2\varphi^2(\boldsymbol{x}^r).
\end{aligned}$$

Substituting these upper bounds into the right hand side of inequality (3.8) yields

$$F(\boldsymbol{x}^{r+1}) - F(\hat{\boldsymbol{x}}^r) = O(\varphi^2(\boldsymbol{x}^r) + \parallel \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \parallel^2). \tag{3.9}$$

This proves the desired result.

The following result establishes the sufficient decrease condition for the APS algorithm under some conditions on the error sequence $\boldsymbol{e}^r$ and the step size sequence α^r

Lemma 2 *Consider an APS algorithm defined by* (3.1)-(3.2) *for some* $\kappa > 0$ *and step-sizes* α^r *satisfying*

$$0 < \underline{\alpha} \leq \alpha^r \leq \bar{\alpha} < \frac{2}{L + 2\kappa}, \ \ \textit{for some } \underline{\alpha} \textit{ and } \bar{\alpha}, \ \forall\, r, \tag{3.10}$$

then it satisfies the sufficient decrease property (3.3).

Proof By the optimality condition for $\boldsymbol{x}^{r+1}$, there exists a $\boldsymbol{g} \in \partial f_1(\boldsymbol{x}^{r+1})$ such that

$$\langle \alpha^r \boldsymbol{g} + \alpha^r \nabla f_2(\boldsymbol{x}^r) + \alpha^r \boldsymbol{e}^r + \boldsymbol{x}^{r+1} - \boldsymbol{x}^r, \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle \geq 0, \ \forall \boldsymbol{y} \in X.$$

Moreover, the convexity of f_1 in $\boldsymbol{x}^{r+1}$ implies that for the vectors $\boldsymbol{y} \in X$ and $\boldsymbol{g} \in \partial f_1(\boldsymbol{x}^{r+1})$, used in the above inequality, we have

$$f_1(\boldsymbol{y}) - \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle \geq f_1(\boldsymbol{x}^{r+1}).$$

Using the above two relations and the convexity of f_2, we obtain

$$\begin{aligned}
F(\boldsymbol{y}) &\geq f_1(\boldsymbol{y}) + f_2(\boldsymbol{x}^r) + \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{y} - \boldsymbol{x}^r \rangle \\
&= f_1(\boldsymbol{y}) + f_2(\boldsymbol{x}^r) + \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \rangle + \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle \\
&\geq f_1(\boldsymbol{y}) + f_2(\boldsymbol{x}^r) + \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \rangle - \langle \boldsymbol{e}^r + \frac{1}{\alpha^r}(\boldsymbol{x}^{r+1} - \boldsymbol{x}^r) + \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle \\
&\geq [f_1(\boldsymbol{x}^{r+1}) + \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle] + f_2(\boldsymbol{x}^r) + \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \rangle \\
&\quad - \langle \boldsymbol{e}^r + \frac{1}{\alpha^r}(\boldsymbol{x}^{r+1} - \boldsymbol{x}^r) + \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle \\
&= f_1(\boldsymbol{x}^{r+1}) + f_2(\boldsymbol{x}^r) + \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \rangle - \langle \boldsymbol{e}^r + \frac{1}{\alpha^r}(\boldsymbol{x}^{r+1} - \boldsymbol{x}^r), \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle.
\end{aligned} \tag{3.11}$$

Since L is the Lipschitz constant of ∇f_2, it follows from Taylor expansion of f_2 around the point $\boldsymbol{x}^r$ that

$$f_2(\boldsymbol{x}^{r+1}) \leq f_2(\boldsymbol{x}^r) + \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \rangle + \frac{L}{2} \|\boldsymbol{x}^{r+1} - \boldsymbol{x}^r\|^2.$$

The above inequality together with (3.11) imply that

$$\begin{aligned} F(\boldsymbol{y}) &\geq f_1(\boldsymbol{x}^{r+1}) + f_2(\boldsymbol{x}^r) + \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \rangle - \langle \boldsymbol{e}^r + \frac{1}{\alpha^r}(\boldsymbol{x}^{r+1} - \boldsymbol{x}^r), \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle \\ &\geq f_1(\boldsymbol{x}^{r+1}) + f_2(\boldsymbol{x}^{r+1}) - \frac{L}{2}\|\boldsymbol{x}^{r+1} - \boldsymbol{x}^r\|^2 - \langle \boldsymbol{e}^r + \frac{1}{\alpha^r}(\boldsymbol{x}^{r+1} - \boldsymbol{x}^r), \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle \\ &= F(\boldsymbol{x}^{r+1}) - \frac{L}{2}\|\boldsymbol{x}^{r+1} - \boldsymbol{x}^r\|^2 - \langle \boldsymbol{e}^r + \frac{1}{\alpha^r}(\boldsymbol{x}^{r+1} - \boldsymbol{x}^r), \boldsymbol{y} - \boldsymbol{x}^{r+1} \rangle. \end{aligned}$$

Specializing $\boldsymbol{y} = \boldsymbol{x}^r$, and using the Cauchy-Schwartz inequality to get

$$F(\boldsymbol{x}^r) - F(\boldsymbol{x}^{r+1}) \geq \frac{2 - 2\alpha^r\kappa - \alpha^r L}{2\alpha^r}\|\boldsymbol{x}^{r+1} - \boldsymbol{x}^r\|^2.$$

Moreover, we know that

$$\frac{2 - 2\alpha^r\kappa - \alpha^r L}{2\alpha^r} \geq \frac{2 - 2\bar{\alpha}\kappa - \bar{\alpha}L}{2\bar{\alpha}} > 0, \quad \forall r,$$

which further implies the intended result.

Finally, we need the following Lemma to prove the linear convergence of the APS method. Its proof is relegated to the Appendix.

Lemma 3 *For $\alpha > 0$, we have*

1. *The function $\alpha\|\tilde{\nabla} f(\boldsymbol{x}, \alpha)\|$ is monotonically increasing with α.*
2. *The function $\|\tilde{\nabla} f(\boldsymbol{x}, \alpha)\|$ is monotonically decreasing with α.*

Now we are ready to state and prove the linear convergence of APS class of algorithms.

Theorem 2 *Assume problem* (1.1) *satisfies the local error bound property. Let $\boldsymbol{x}^1$, $\boldsymbol{x}^2$, $\boldsymbol{x}^3, \cdots$ be any sequence, which together with a sequence of scalars $\{\alpha^r\}$ satisfying $\liminf_r \alpha^r > 0$ and some sequence $\{\boldsymbol{e}^r\}$ in $\mathbb{R}^n$, satisfies* (3.1)-(3.3). *Then $\{f(\boldsymbol{x}^r)\}$ converges at least Q-linearly and $\{\boldsymbol{x}^r\}$ converges at least R-linearly to an optimal solution in X^*.*

Proof First of all the sufficient decrease condition (3.3) implies

$$\| \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \|^2 \to 0.$$

Moreover, we have

$$\begin{aligned}
&\| \boldsymbol{x}^r - \mathrm{prox}_{\alpha^r f_1}[\boldsymbol{x}^r - \alpha^r \nabla f_2(\boldsymbol{x}^r)] \| \\
&\qquad \leq \| \boldsymbol{x}^r - \boldsymbol{x}^{r+1} \| + \| \boldsymbol{x}^{r+1} - \mathrm{prox}_{\alpha^r f_1}[\boldsymbol{x}^r - \alpha^r \nabla f_2(\boldsymbol{x}^r)] \| \\
&\qquad \leq \| \boldsymbol{x}^r - \boldsymbol{x}^{r+1} \| + \alpha^r \| \boldsymbol{e}^r \| \\
&\qquad \leq (\bar{\alpha}\kappa + 1) \| \boldsymbol{x}^r - \boldsymbol{x}^{r+1} \|, \qquad (3.12)
\end{aligned}$$

where the first inequality follows from the triangular inequality, the second one is due to the non-expansiveness property of the proximity operator and the third one is due to (3.2).

Since $\alpha^r \geq \underline{\alpha}$ for all $r > 0$, we obtain that

$$\begin{aligned}
\psi(\boldsymbol{x}^r) &= \| \boldsymbol{x}^r - \mathrm{prox}_{f_1}[\boldsymbol{x}^r - \nabla f_2(\boldsymbol{x}^r)] \| \\
&\leq \frac{1}{\min\{1, \underline{\alpha}\}} \| \boldsymbol{x}^r - \mathrm{prox}_{\alpha^r f_1}[\boldsymbol{x}^r - \alpha^r \nabla f_2(\boldsymbol{x}^r)] \| \\
&\leq \frac{\kappa + 1}{\min\{1, \underline{\alpha}\}} \| \boldsymbol{x}^r - \boldsymbol{x}^{r+1} \|
\end{aligned}$$

where the first inequality is due to Lemma 3 and the second one is due to (3.12). Therefore, since $\| \boldsymbol{x}^{r+1} - \boldsymbol{x}^r \| \to 0$, we have that

$$\psi(\boldsymbol{x}^r) \to 0.$$

Now using the local error bound condition implies that, for sufficiently large r, there exists a constant τ such that

$$\varphi(\boldsymbol{x}^r) \leq \tau \, \psi(\boldsymbol{x}^r) \to 0, \qquad (3.13)$$

which further implies $\varphi(\boldsymbol{x}^r) \to 0$. Then, it follows from the cost-to-go estimate that

$$F(\boldsymbol{x}^r) \to F^*.$$

Now we use the local error bound condition together with the cost-to-go estimate

to get

$$\begin{aligned} F(\boldsymbol{x}^{r+1}) - F^* &\leq c_2 \left(\varphi^2(\boldsymbol{x}^r) + \parallel \boldsymbol{x}^r - \boldsymbol{x}^{r+1} \parallel^2\right) \\ &\leq c_2 \left(\tau \parallel \boldsymbol{x}^r - \mathrm{prox}_{f_1}[\boldsymbol{x}^r - \nabla f_2(\boldsymbol{x}^r)] \parallel^2 + \parallel \boldsymbol{x}^r - \boldsymbol{x}^{r+1} \parallel^2\right) \\ &\leq \frac{c_2\tau}{\min\{1, \underline{\alpha}^2\}} (\parallel \boldsymbol{x}^r - \mathrm{prox}_{\alpha^r f_1}[\boldsymbol{x}^r - \alpha^r \nabla f_2(\boldsymbol{x}^r)] \parallel^2) + c_2 \parallel \boldsymbol{x}^r - \boldsymbol{x}^{r+1} \parallel^2 \end{aligned}$$

Next we use (3.2) and the non-expansiveness of the proximity operator to bound

$$\begin{aligned} &\parallel \boldsymbol{x}^r - \mathrm{prox}_{\alpha^r f_1}[\boldsymbol{x}^r - \alpha^r \nabla f_2(\boldsymbol{x}^r)] \parallel^2 \\ &\qquad \leq \ 2 \left(\parallel \boldsymbol{x}^r - \boldsymbol{x}^{r+1} \parallel^2 + \parallel \boldsymbol{x}^{r+1} - \mathrm{prox}_{\alpha^r f_1}[\boldsymbol{x}^r - \alpha^r \nabla f_2(\boldsymbol{x}^r)] \parallel^2\right) \\ &\qquad \leq \ 2 \left(\bar{\alpha}^2 \kappa^2 + 1\right) \|\boldsymbol{x}^{r+1} - \boldsymbol{x}^r\|^2. \end{aligned}$$

This further implies

$$\begin{aligned} F(\boldsymbol{x}^{r+1}) - F^* &\leq c_2 \left(\frac{2\tau(1 + \bar{\alpha}^2\kappa^2)}{\min\{1, \underline{\alpha}^2\}} + 1\right) \parallel \boldsymbol{x}^r - \boldsymbol{x}^{r+1} \parallel^2 \\ &\leq \frac{c_2}{c_1} \left(\frac{2\tau(1 + \bar{\alpha}^2\kappa^2)}{\min\{1, \underline{\alpha}^2\}} + 1\right) (F(\boldsymbol{x}^r) - F(\boldsymbol{x}^{r+1})), \end{aligned}$$

where the last step is due to sufficient decrease condition. Let $\gamma > 0$ denotes the constant before the term $(F(\boldsymbol{x}^r) - F(\boldsymbol{x}^{r+1}))$ in the last inequality. Therefore, we have shown that

$$F(\boldsymbol{x}^{r+1}) - F^* \leq \gamma \left(F(\boldsymbol{x}^r) - F(\boldsymbol{x}^{r+1})\right).$$

Rearranging the terms in this inequality yields that

$$F(\boldsymbol{x}^{r+1}) - F^* \leq \frac{\gamma}{\gamma + 1} \left(F(\boldsymbol{x}^r) - F^*\right)$$

This implies the Q-linear convergence of $F(\boldsymbol{x}^r) \to F^*$. Together with the sufficient decrease condition, it implies the R-linear convergence of $\{\boldsymbol{x}^r\}$ to an optimal solution. This completes the proof of Theorem 2.

3.3 Related Works

We studied the APS algorithm which is a general framework of first order methods for the nonsmooth convex optimization problem (1.1). This framework combines the existing framework of AGP with the proximal splitting technique, and as such, it includes the GP, AGP and proximal splitting methods as special cases. Moreover, the well known block coordinate descent (BCD) algorithm is also a special case of APS (see chapter 4).

Our result differs from the existing proximal splitting methods and analysis in several aspects. Among the existing works [22, 28, 29], the only one which considers an error term in the proximal splitting algorithm is [29], while the other two ([28] and [22]) are focussed on the pure proximal splitting algorithm. The result in [29] does not provide the linear convergence except in the strongly convex case which is a special case of our result. The reason for such difference is that we use a local error bound condition in place of the strong convexity assumption.

The result in [28] deals with the problem of linear convergence from a statistical point of view. It assumes that problem (1.1) comes from an M-estimator formulation with some probabilistic construction. It proves that the iterates will converge linearly to a neighborhood around the optimal solution, but not necessarily an optimal solution. As such, this result is probabilistic and not deterministic. This is in contrast to our result which is a general convex optimization problem in the form (1.1), regardless how it is generated. That said, by utilizing the so called restricted strong convexity and restricted smoothness (see [28]) instead of an error bound, the authors have established the linear convergence of the proximal splitting algorithm for a broad range optimization problems with non-smooth regularizers such as L_1 norm or nuclear norm.

3.4 Simulation Results

In this section we use the PSM algorithm, which is a special case of the APS with $\boldsymbol{e}^r = 0$ for every r, to solve the LASSO problem (1.3)

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda\|\boldsymbol{x}\|_1.$$

Clearly, LASSO is a special case of problem (1.1) with $f_1(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1$ and $f_2(\boldsymbol{x}) =$

$\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2$. In the simulations the ambient dimension n and the number of measurements m are set to 2^{10} and 2^8, respectively. The true vector $\boldsymbol{x}^* \in \mathbb{R}^n$ has only five percent of its entries non-zero. The non-zero entries of $\boldsymbol{x}^*$ are independently generated according to the standard Gaussian distribution $\mathcal{N}(0,1)$. The matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ has i.i.d. standard Gaussian entries as well. Finally, the measurements vector $\boldsymbol{b} \in \mathbb{R}^m$ is generated as

$$\boldsymbol{b} = \boldsymbol{A}\,\boldsymbol{x}^* + \boldsymbol{e}, \tag{3.14}$$

where e is a Gaussian i.i.d. noise vector whose components have a variance $\sigma > 0$. Assuming the true answer $\boldsymbol{x}^*$ is unknown, our goal is to estimate it by solving the LASSO problem. We set the regularization parameter to $\lambda = 0.1\|\boldsymbol{A}^T\boldsymbol{y}\|_\infty$ as in [10].

In iteration r, the PSM method suggests solving the following problem to obtain the search direction $\boldsymbol{d}^r$:

$$\boldsymbol{d}^r = \arg\min_{\boldsymbol{d}\in\mathbb{R}^n} \langle \nabla f_2(\boldsymbol{x}^r), \boldsymbol{d}\rangle + \frac{1}{2}\|\boldsymbol{d}\|^2 + \lambda\|\boldsymbol{x}^r + \boldsymbol{d}\|_1.$$

The solution to this problem is obtained by using the soft-threshold function

$$\boldsymbol{d}^r = \text{soft}\left(\boldsymbol{x}^r - \nabla f_2(\boldsymbol{x}^r); \lambda\right) - \boldsymbol{x}^r,$$

where $\text{soft}(y,\tau) = \text{sign}(y)\max\{|y|-\tau, 0\}$ is the well-known soft-threshold function.

We choose the Armijo step-size rule since it is simple and efficient. This step-size rule is widely used in the case of smooth optimization, however it can also be adapted to non-smooth problem (1.1) (see [30] for more discussions on this step-size rule). In each iteration r, we set $\alpha^r = \beta^k$ where $\beta \in (0,1)$ is a constant and k is the smallest positive integer satisfying

$$f(\boldsymbol{x}^r + \beta^k\boldsymbol{d}^r) \le f(\boldsymbol{x}^r) + \gamma\beta^k\left\{\nabla f_2(\boldsymbol{x}^r)^T\boldsymbol{d}^r + \lambda\left(\|\boldsymbol{x}^r + \boldsymbol{d}^r\|_1 - \|\boldsymbol{x}^r\|_1\right)\right\}, \tag{3.15}$$

with a constant $\gamma \in (0,1)$. In our simulations, we set $\beta = 0.3$ and $\gamma = 0.05$. The Armijo rule requires several objective evaluations to find the appropriate integer k. A simpler method is to use a constant step-size $\alpha^r = \alpha$ with $\alpha < 1/L$. The convergence is also guaranteed in this case but will not be as fast as the Armijo rule.

The algorithm is terminated whenever the following optimality condition is satisfied

$$\|\nabla f_2(\boldsymbol{x}^r)\|_\infty < \lambda + \epsilon$$

where $\epsilon > 0$ is a small constant (e.g. $\epsilon = 10^{-7}$) and $\|\cdot\|_\infty$ is called the infinity norm and is defined as $\|\boldsymbol{y}\|_\infty = \max_i |y_i|$. We measure the converging of the PSM algorithm by computing the following normalized error function.

$$error(r) = \log\left(\frac{F(\boldsymbol{x}^r) - F(\boldsymbol{x}^*)}{F(\boldsymbol{x}^1) - F(\boldsymbol{x}^*)}\right)$$

for every iteration r. Since our analysis guarantees a linear rate of convergence for the PSM algorithm when applied to the LASSO problem, we expect this function to be linearly decreasing.

Figure 3.1 shows the error function $error(r)$ versus the iteration number r for three different values of noise variance, $\sigma = 10^{-4}, 10^{-3}, 10^{-2}$. As the figure shows, the convergence of the PSM algorithm is linear for the LASSO problem.

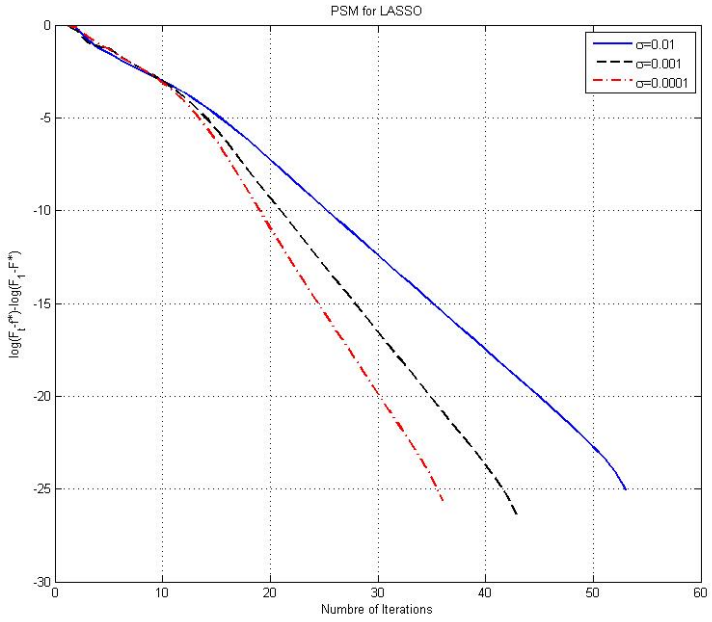

Figure 3.1: Convergence of PSM for the LASSO problem

Chapter 4

Block Coordinate Descent Method

The block coordinate descent method (BCD) has a long history in optimization and numerical analysis.

In this chapter we will show that, the BCD algorithm is also a special case of the APS framework. The convergence rate of APS class of algorithms was analyzed under a local error bound condition (see the previous chapter). Therefore, our result implies the linear convergence rate of Block Coordinate Descent Method (BCD) for (1.1) for the LASSO or group LASSO type of problems when $f_1(\boldsymbol{x}) = \sum_{J\in\mathcal{J}} w_J\|\boldsymbol{x}_J\|_2$ or $f_1(\boldsymbol{x}) = \sum_{J\in\mathcal{J}} w_J\|\boldsymbol{x}_J\|_2 + \lambda\|\boldsymbol{x}\|_1$. The BCD algorithm is one of the main algorithms used to solve large scale optimization problems due to the simplicity of its updates (especially for the LASSO or group LASSO type of problems in which each step of BCD is equivalent to a shrinkage operator [22]). This linear convergence result provides theoretical proof for effectiveness of BCD in handling such problems. In the sequel, we formally define the BCD method and introduce the main assumptions required for our convergence analysis of this algorithm.

Let $\boldsymbol{x} \in \mathbb{R}^n$ have the block form of $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K)'$, where $\boldsymbol{x}_k \in \mathbb{R}^{i_k}$ and $\sum_{k=1}^K i_k = n$. Consider the minimization problem (1.1), in which f_1 is separable over

the blocks. In other words it can be written as

$$f_1(\boldsymbol{x}) = d_1(\boldsymbol{x}_1) + \cdots + d_K(\boldsymbol{x}_K), \tag{4.1}$$

where $d_k,\ k = 1, \cdots, K$ are all convex (but not necessarily smooth) functions. Furthermore, $\mathbf{X}$ is a closed convex set in $\mathbb{R}^n$ which is also separable over the blocks, i.e. it can be written as the following Cartesian product

$$\mathbf{X} = \mathbf{X}_1 \times \mathbf{X}_2 \times ... \times \mathbf{X}_K, \tag{4.2}$$

where $\mathbf{X}_k$ is a closed convex subset of $\mathbb{R}^{i_k}$. Note that the LASSO problem (1.3), group LASSO problem (1.5) and logestic group LASSO problem (1.6) admit the decomposition specified by (4.1) and (4.2).

Consider the BCD method whereby after the r-th iteration, $r \geq 0$, we choose an index $s \in \{1, 2, ..., K\}$ and compute the new iterate $\boldsymbol{x}^{r+1} = (\boldsymbol{x}_1^{r+1}, \boldsymbol{x}_2^{r+1}, ..., \boldsymbol{x}_K^{r+1})$ as follows

$$\begin{aligned} \boldsymbol{x}_s^{r+1} &= \text{argmin}_{\boldsymbol{x}_s \in \mathbf{X}_s} F(\boldsymbol{x}_1^r, \boldsymbol{x}_2^r, ..., \boldsymbol{x}_{s-1}^r, \boldsymbol{x}_s, \boldsymbol{x}_{s+1}^r, ..., \boldsymbol{x}_K^r) \\ \boldsymbol{x}_j^{r+1} &= \boldsymbol{x}_j^r, \quad j \neq s. \end{aligned} \tag{4.3}$$

where $(\boldsymbol{x}_1^r, \boldsymbol{x}_2^r, ..., \boldsymbol{x}_K^r)$ denotes the iterate at r-th iteration. The blocks are chosen cyclically or essentially cyclically to be updated at every iteration. The essentially cyclic update ensures that there exists an integer $N \geq K$ such that after this many iterations all the blocks are updated at least once.

It is known [27] that the BCD algorithm with cyclic update or essentially cyclic update converges to the optimal solution for the set of non-smooth optimization problems that the non-smooth part is separable as defined in (4.1).

To establish linear convergence of the BCD method, we need the assumption that the smooth part f_2 is strongly convex in each block, in the sense that there exists a scalar $\gamma \geq 0$ such that, for any $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K) \in \mathbf{X}$ and any $s \in \{1, 2, ..., K\}$,

$$f_2(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{s-1}, \boldsymbol{x}_s + \Delta\boldsymbol{x}_s, \boldsymbol{x}_{s+1}, ..., \boldsymbol{x}_K) - f_2(\boldsymbol{x}) - \langle \nabla_s f_2(\boldsymbol{x}), \Delta\boldsymbol{x}_s \rangle \geq \gamma \|\Delta\boldsymbol{x}_s\|^2, \tag{4.4}$$

for all feasible $\Delta \boldsymbol{x}_s \in \mathbb{R}^{i_s}$, where $\nabla_s f_2$ denotes the vector of partial derivatives of f_2 with respect to the s-th block. It is obvious that if the function f_2 is block coordinate-wise strongly convex, then the coordinate descent method satisfies the sufficient decrease condition (3.3) [cf. Proposition 3.4 in [14]].

For the applications described in chapter 1, the coordinate-wise strong convexity of f_2 imposes a mild condition on the linear operator $\boldsymbol{A}$. For example, in LASSO problem (1.3), if each column of $\boldsymbol{A}$ is non-zero and we consider each element in $\boldsymbol{x}$ to be a block, then the problem is coordinate-wise strongly convex. Furthermore, for the group LASSO problem, f_2 is block coordinate-wise strongly convex if the columns of $\boldsymbol{A}$ corresponding to a block are linearly independent. A similar condition can be derived for the logestic group LASSO problem (1.6) to ensure block coordinate-wise strong convexity. The following proposition shows that the block coordinate descent method for the L_1 norm minimization problem and the Group LASSO minimization problem is an APS method.

Proposition 3 *Under the above assumptions in* (4.1), (4.2) *and* (4.4), *the block coordinate descent method with cyclic update can be written in the APS form with an error term* $\boldsymbol{e}$ *which satisfies* (3.2).

Proof Let us define two different iteration counters. The outer iteration index s is a counter for the number of updating cycles of the BCD algorithm, and the inner iteration index k corresponds to the variable block being updated in a given cycle. Thus, at iteration $r = sK + k$ (with $1 \leq k \leq K$), the k-th variable block is updated in the s-th cycle. Throughout the proof, the notation $\boldsymbol{x}^r$ means the r-th iterate of the BCD algorithm, and $\boldsymbol{x}_k^r$ represents the k-th block of r-th iterate.

For simplicity, let us assume that there are no constraints. This assumption is not restricting as one can always add the indicator functions of the constraining sets to the objective. Since the feasible set is assumed to have a special structure as in (4.2), the separability of the non-smooth objective component will still be preserved after this change.

The optimality condition at the r-th iteration for BCD method is,

$$\boldsymbol{g} + \nabla_k f_2(\boldsymbol{x}^r) = \boldsymbol{0}, \tag{4.5}$$

for some $\boldsymbol{g}$ in $\partial d_k(\boldsymbol{x}_k^r)$ (Note that we assumed $f_1(\boldsymbol{x}) = d_1(\boldsymbol{x}_1) + \cdots + d_K(\boldsymbol{x}_K)$). Now in each fixed cycle s, define $r' = Ks$ and the error vector $\boldsymbol{e}^s = (\boldsymbol{e}_1^s, \cdots, \boldsymbol{e}_K^s)$, as follows

$$\boldsymbol{e}_k^s = \boldsymbol{x}_k^{r'+k} - \boldsymbol{x}_k^{r'} + \nabla_k f_2(\boldsymbol{x}^{r'}) - \nabla_k f_2(\boldsymbol{x}^{r'+k}),\ \forall\ k = 1, \cdots, K. \tag{4.6}$$

Then, it is obvious that $\boldsymbol{x}^{r'+K}$ generated by BCD can also be derived from the following update rule

$$\boldsymbol{x}^{(s+1)K} = \mathrm{prox}_{f_1}(\boldsymbol{x}^{sK} - \nabla f_2(\boldsymbol{x}^{sK}) + \boldsymbol{e}^s). \tag{4.7}$$

Now we can show that $\|\boldsymbol{e}^s\| \leq \kappa \|\boldsymbol{x}^{(s+1)K} - \boldsymbol{x}^{sK}\|$ for some $\kappa > 0$. Since f_2 has Lipschitz continuous gradient, it follows that

$$\begin{aligned}
\|\boldsymbol{e}_k^s\| \ &\leq \|\boldsymbol{x}_k^{r'+k} - \boldsymbol{x}_k^{r'}\| + \|\nabla_k f_2(\boldsymbol{x}^{r'}) - \nabla_k f_2(\boldsymbol{x}^{r'+k})\| \\
&\leq \ \|\boldsymbol{x}_k^{r'+k} - \boldsymbol{x}_k^{r'}\| + L\|\boldsymbol{x}^{r'} - \boldsymbol{x}^{r'+k}\| \\
&\leq \ (L+1)\|\boldsymbol{x}^{r'+k} - \boldsymbol{x}^{r'}\| \ \leq \ (L+1)\|\boldsymbol{x}^{r'+K} - \boldsymbol{x}^{r'}\| \ = \ (L+1)\|\boldsymbol{x}^{(s+1)K} - \boldsymbol{x}^{sK}\|,
\end{aligned}$$

where the second step is due to the Lipschitz condition on ∇f_2 and the last inequality is due to the block coordinate-wise update in the algorithm. This further implies that

$$\|\boldsymbol{e}^s\| \leq K(L+1)\|\boldsymbol{x}^{(s+1)K} - \boldsymbol{x}^{sK}\|,$$

so that the condition (3.2) holds with $\kappa = K(L+1)$.

Remark 1 Note that a similar proof can be done to show that BCD algorithm with essentially cyclic update lies within the APS framework, with an error term $\boldsymbol{e}$ which satisfies (3.2).

The following Proposition is a direct consequence of Corollary 1 and Proposition 3.

Proposition 4 *The BCD algorithm (with cyclic or essentially cyclic update) generates a sequence of iterates that converges R-linearly to a solution in X^* for LASSO problem* (1.3)*, Group LASSO problem* (1.5) *and logistic Group LASSO problem* (1.6)*, if the objective function is block coordinate-wise strongly convex.*

4.1 Related Works

To our knowledge this is the first result which shows the linear convergence rate of the exact BCD algorithm for solving problem (1.1) without requiring the strong convexity of the objective function. Here we would like to survey past works relating to our results: Earliest studies on the convergence rate of the BCD algorithm required the smoothness of the objective function [15], [31], [14]. These works showed that when the objective function is smooth (but not necessarily strongly convex) the BCD algorithm with the Gauss-Seidel update rule converges linearly, provided that the local error bound condition is satisfied around the solution set. The (Block) Coordinate Gradient Descent (abbreviated as CGD) algorithm proposed in [30] is a relevant method to BCD which can solve non-smooth problem (1.1) under the assumptions (4.1)-(4.2). As shown in [30], this algorithm enjoys having linear rates of convergence when the local error bound condition holds. The BCD and CGD algorithms both exploit block coordinate-wise updates to solve the problem (1.1). However, unlike BCD which solves the exact subproblem in each iteration, CGD approximates the smooth component f_2 by a strictly convex quadratic function. Therefore, the analysis given in [30] does not prove the linear convergence rate property of the exact BCD algorithm.

The Block Successive Upper-bound Minimization (BSUM) approach, studied in [32], is a general inexact BCD method for minimizing a (possibly non-convex) objective F over a decomposable feasible set X formed as in (4.2). BSUM updates the variable blocks by successively minimizing a sequence of approximations of F which are locally tight upper-bounds of this function. The convergence analysis in [32] shows the iterates generated by the BSUM algorithm converge to the set of stationary points when the function level sets are compact. However, it does not imply any convergence rate results for the BCD algorithm. From a different perspective, BSUM can be viewed as a BCD variant of the majorization-minimization (also called successive upper-bound minimization) method. The convergence analysis of the majorization-minimization algorithm is done in [33]. As shown in that work, this algorithm exhibits linear rates of convergence when the objective function satisfies the strong convexity assumption.

Another relevant line of work is done in [34], [35], [36], [37], [38] which study randomized versions of BCD for solving problem (1.1). In each iteration of the randomized

algorithm, a block of variables is randomly picked according to some probability distribution; Then, a strictly convex objective approximation, constructed in a similar way as in [30], is minimized with respect to those variables. Since the approximation function is (block) separable across the variables, the updates can be implemented in a distributed fashion. When the assumptions (4.1)-(4.2) hold, the authors in [35] prove that the randomized algorithm exhibits a global linear convergence rate in probability when a, so called, generalized error bound condition holds for the problem. In contrast to their analysis, we consider the case where the variables are updated in a cyclic (or essentially cyclic) fashion by minimizing the exact objective function. Therefore, our convergence rate results provide deterministic gurantees when the local error bound condition holds. In [36], the randomized BCD is applied for solving problem (1.1) when the variables are coupled with linear constraints. The analysis given in [36], [37] and [38] prove the linear convergence rate of the randomized BCD algorithm under the restricting assumption of the objective function being strongly convex. Further related works on the distributed implementation of the randomized BCD algorithm can be found in [39], [40], [41].

Beside the extensive interest in the randomized version of the BCD algorithm, the iteration complexity of the deterministic version is also studied recently [42], [43]. For a unified iteration complexity analysis for a family of BCD-type algorithms with either Gauss-Seidel or randomized coordinate update rules, the readers can refer to [44].

4.2 Simulation Results

4.2.1 LASSO Problem

In this section, we use the coordinate descent algorithm to solve the LASSO problem (1.3). We assume a linear data model as

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}, \tag{4.8}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n] \in \mathbb{R}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{e} \in \mathbb{R}^m$ are generated randomly as in section 3.4.

The algorithm works by cyclically updating the coordinates. Therefore, at iteration $r = kn+i$, $k \in \mathbb{Z}$, the i-th coordinate is updated according to the following optimization

problem

$$x_i^r = \arg\min_{x_i} \frac{1}{2}\|\boldsymbol{b} - \sum_{j=1,\, j\neq i}^{n} \boldsymbol{a}_i x_j^{r-1} - \boldsymbol{a}_i x_i\|_2^2 + \lambda\,|x_i|. \tag{4.9}$$

Let $\boldsymbol{z}_i^r = \boldsymbol{b} - \sum_{j=1,\, j\neq i} \boldsymbol{a}_i x_j^{r-1}$. Then the optimal solution to (4.9) is given by

$$x_i^r = \text{soft}\left(\frac{1}{\boldsymbol{a}_i^T \boldsymbol{a}_i}\boldsymbol{z}_i^r; \frac{\lambda}{\boldsymbol{a}_i^T \boldsymbol{a}_i}\right),$$

which is a very simple update rule. The algorithm will be terminated if the stopping criterion (3.15) is met. We evaluate the convergence of the coordinate descent algorithm by using the following relative error function

$$error(r) = \log\left(\frac{F(\boldsymbol{x}^r) - F(\boldsymbol{x}^*)}{F(\boldsymbol{x}^1) - F(\boldsymbol{x}^*)}\right) \tag{4.10}$$

for iterations $r = kn$. Since our analysis guarantees a linear rate of convergence for the BCD algorithm when applied to the LASSO problem, we expect this function to linearly decrease when the number of iterations is sufficiently large. Figure 4.1 shows the error function versus the cycle number k for three different values of noise variance, $\sigma = 10^{-2}$, 10^{-3}, 10^{-4}. As can be seen in this figure, the error function is indeed linearly decreasing with the cycle number.

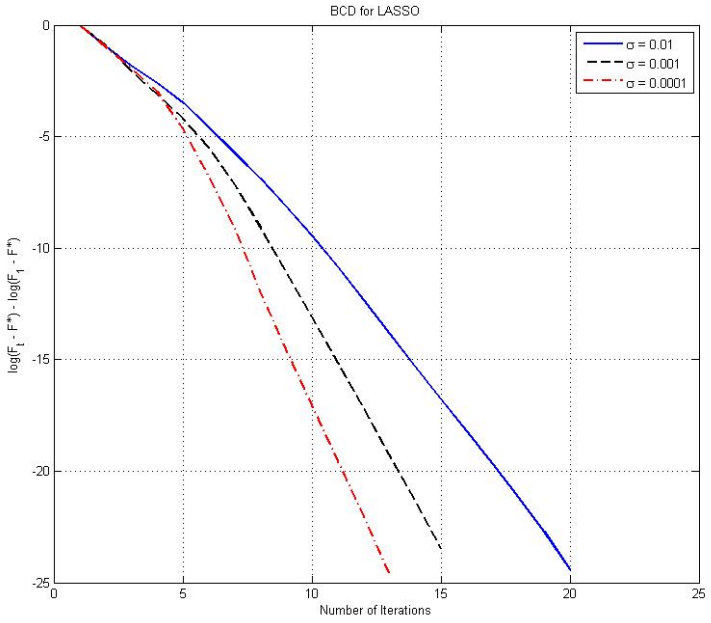

Figure 4.1: Convergence of CD for the LASSO problem

4.2.2 Group LASSO Problem

We now illustrate the use of the block coordinate descent algorithm with group sparse regularization functions.When $\boldsymbol{x}^*$ has a predefined block structure, it can be estimated by solving the group LASSO problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \lambda \sum_{J\in\mathcal{J}} \|\boldsymbol{x}_J\|_2$$

where $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}^* + \boldsymbol{e}$. Our simulations in this subsection uses synthetic data and is mainly designed to show the efficiency of BCD in solving the group LASSO subproblem.

The matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ has dimensions $m = 2^{10}$ and $n = 2^{12}$ and is filled with i.i.d. standard Gaussian entries. The true vector $\boldsymbol{x}^*$ has $n = 2^{12}$ components, divided into $n' = 64$ groups of length $l = 64$. To generate $\boldsymbol{x}^*$, we randomly choose 8 groups and fill them with zero-mean Gaussian random samples of unit variance, while all other groups are filled with zero (this construction was taken from section 4 of [10]). The error vector is white Gaussian noise with variance σ. Finally the value of λ was set to $\lambda = 0.1\|\boldsymbol{A}^T\boldsymbol{y}\|$.

Figure 4.2 shows the original signal as well as the perfectly reconstructed one.

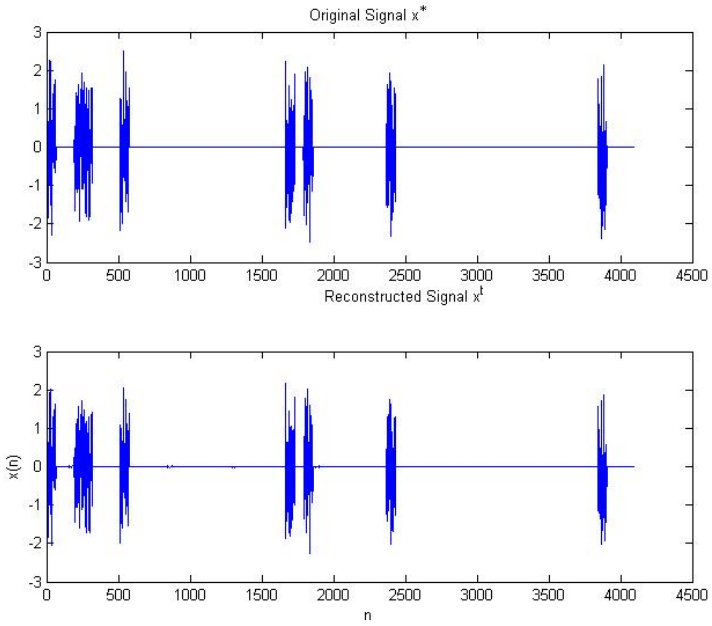

Figure 4.2: Comparison of the original and the BCD reconstructed vectors for group LASSO

Like in the case of the LASSO problem, the BCD method has simple updates when applied to group LASSO problem. In particular, in iteration $r = kn' + i$, the i-th block of $\boldsymbol{x}$ is updated according to the following formulation

$$\boldsymbol{x}_i^r = \arg\min_{\boldsymbol{x}_i \in \mathbb{R}^l} \frac{1}{2}\|\boldsymbol{b} - \sum_{J \in \mathcal{J}, J \neq i} \boldsymbol{A}_J \boldsymbol{x}_J^{r-1} - \boldsymbol{A}_i \boldsymbol{x}_i\|_2^2 + \lambda \|\boldsymbol{x}_i\|_2. \tag{4.11}$$

Defining $\boldsymbol{z}_i^r = \boldsymbol{b} - \sum_{J \in \mathcal{J}, J \neq i} \boldsymbol{A}_J \boldsymbol{x}_J^{r-1}$, the first order optimality condition of the problem (4.11) implies that

$$\boldsymbol{A}_i^T(\boldsymbol{A}_i \boldsymbol{x}_i^r - \boldsymbol{z}_i^r) + \lambda \boldsymbol{g} = 0, \quad \text{for some } \boldsymbol{g} \in \partial \|\boldsymbol{x}_i^r\|_2.$$

If $\|\boldsymbol{A}_i^T \boldsymbol{A}_i \boldsymbol{z}_i^r\|_2 \leq \lambda$, then the optimal solution is zero, i.e. $\boldsymbol{x}_i^r = 0$. Otherwise, the optimal solution is given by

$$\boldsymbol{x}_i^r = \left(\boldsymbol{A}_i^T \boldsymbol{A}_i + \lambda\, \delta\, \boldsymbol{I}_l\right)^{-1} \boldsymbol{A}_i^T \boldsymbol{z}_i^r,$$

where $\boldsymbol{I}_l$ is an identity matrix of size l (remember that l is the size of each block of variables) and δ is a positive constant which is also equal to $\delta = 1/\|\boldsymbol{x}_i^r\|_2$ and can be found using the bisection method.

Figure 4.3 shows the relative error, defined in (4.10), as a function of the cycle number for three different values of the noise variance. The error function has a linear decrease after a few number of iterations, which implies the linear convergence rate of the BCD method in the case of the group LASSO problem.

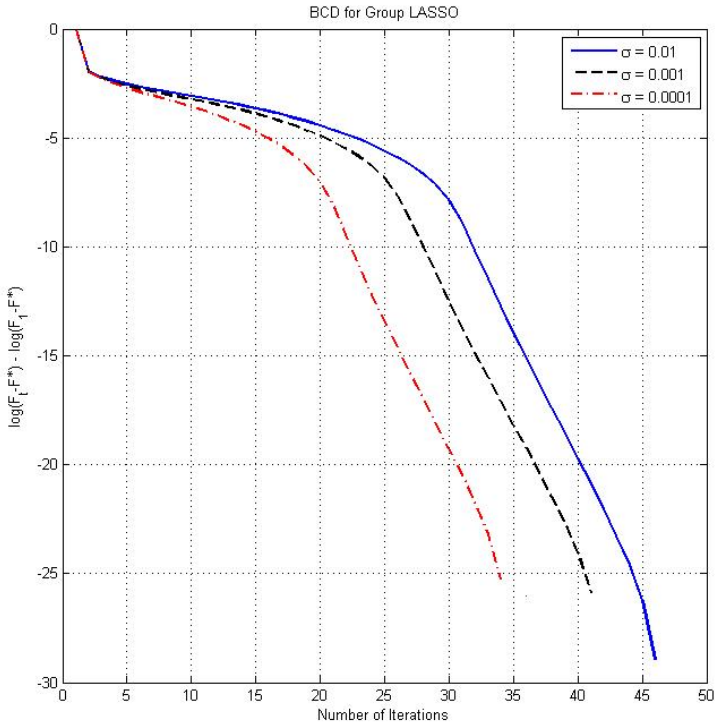

Figure 4.3: Convergence of BCD for the Group LASSO problem

4.2.3 Support Vector Machine Classification

Support vector machines (SVM) are very effective tools for the purpose of classification learning. The task of learning is typically cast as a constrained quadratic problem. Formally, given a training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, +1\}$, we

would like to find the minimizer of the following problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \, \forall i, \end{aligned} \tag{4.12}$$

where $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ is called the Kernel matrix and $\boldsymbol{e} \in \mathbb{R}^n$ is the vector of all ones. In the case of linear SVM, the entries of the kernel matrix are given by the equation $K_{i,j} = y_i \boldsymbol{x}_i^T \boldsymbol{x}_j y_j$. In many large-scale classification problems, the kernel matrix is dense and ill-conditioned making the problem (4.12) challenging to solve.

The problem (4.12) has the form of the formulation (1.1) with $f_2(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha}$ and $f_1(\boldsymbol{\alpha}) = \iota_C(\boldsymbol{\alpha})$, where $\iota_C(\boldsymbol{\alpha})$ is the indicator function of the feasible set defined by the box constraints. Moreover, this problem satisfies the local error bound condition (see Theorem 1 in Chapter 2). Since the feasible set can be expressed as a Cartesian product of closed convex sets as in (4.2) and the objective function is strictly convex in each variable α_i, the BCD algorithm can be applied for solving this problem. As illustrated in [45], the coordinate descent method can update the variables via very simple updates. Assume that the algorithm updates the coordinates in a cyclic manner, i.e. it cycles through $\{1\}, \{2\}, \cdots, \{n\}$. Then at iteration $r = kn + i$, $k \in \mathbb{Z}$, the i-th coordinate α_i must be updated by solving the following problem

$$\begin{aligned} \alpha_i^r = \arg\min_{\alpha_i\in\mathbb{R}} \quad & f_2(\alpha_1^r, \alpha_2^r, \cdots, \alpha_{i-1}^r, \alpha_i, \alpha_{i+1}^{r-1}, \cdots, \alpha_n^{r-1}) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C. \end{aligned} \tag{4.13}$$

The first-order optimality condition of this problem implies that the optimal solution is $\alpha_i^r = \alpha_i^{r-1}$ (i.e. α_i^{r-1} does not need to be updated) if and only if $\nabla_i^P f_2(\boldsymbol{\alpha}^{k-1}) = 0$, where $\nabla^P f_2(\boldsymbol{\alpha})$ is the projected gradient vector defined as

$$\nabla_i^P f_2(\boldsymbol{\alpha}) = \begin{cases} \nabla_i f_2(\boldsymbol{\alpha}) & \text{if } 0 < \alpha_i < C, \\ \min(0, \nabla_i f_2(\boldsymbol{\alpha})) & \text{if } \alpha_i = 0, \\ \max(0, \nabla_i f_2(\boldsymbol{\alpha})) & \text{if } \alpha_i = C. \end{cases}$$

If the projected gradient is zero, we move to the next iteration without updating

α_i^{r-1}. Otherwise, we must find the optimal solution of 4.13 which is given by

$$\alpha_i^r = \min\left(\max\left(\alpha_i^{r-1} - \frac{\nabla_i f_2(\boldsymbol{\alpha}^r)}{K_{i,i}}, 0\right), C\right)$$

where $\nabla_i f_2(\boldsymbol{\alpha}) = (\boldsymbol{K}\boldsymbol{\alpha})_i - 1$ and $K_{i,i} = \boldsymbol{x}_i^T \boldsymbol{x}_i$.

Here, we would like to illustrate the linear convergence of the coordinate descent algorithm for solving the SVM problem. In this experiment, we also use the coordinate descent algorithm with random coordinate selection and compare its performance with the one which uses cyclic coordinate selection. At the beginning of each cycle of the randomized algorithm, a permutation of $\{1, \cdots, n\}$ is randomly chosen and then the variables are updated in the order specified by this permutation. Past results [45] show that solving sub-problems in a random order may give faster convergence.

Our experiments in this subsection are based on a real-world dataset which is a subset of the Reuters Corpus dataset (RCV1[1] Dataset [46]). RCV1 is a benchmark dataset for text classification. The RCV1 subset that was used in this experiment contains 9625 documents with 29992 distinct words, including categories"C15","ECAT","GCAT", and "MCAT", each with 2,022, 2,064, 2,901, and 2,638 documents respectively. In order to do binary classification, the C15 and ECAT categories were labeled as positive and GCAT and MCAT were labeled as negative [47]. In our experiments, we set $C = 10$.

Figure 4.4 shows the training accuracy of the BCD method which is defined as

$$\frac{1}{n}\sum_{i=1}^{n} 1\{y_i = \hat{y}_i\}$$

where $\hat{y}_i = \text{sign}\left(\boldsymbol{w}^T \boldsymbol{x}_i\right)$ is the estimated class for the i-th training example and is obtained by using the linear classifier $\boldsymbol{w} = \sum_{j=1}^{n} \alpha_j y_j \boldsymbol{x}_j$. As the figure shows, after a few iterations, the algorithm correctly classifies almost all of the training examples. The figure also shows that the random update rule requires very few cycles to provide the correct classification of the training data.

[1] The RCV1 dataset is available publicly at `http://www.csie.ntu.edu.tw/cjlin/libsvmtools/` datasets and contains a training set and a test set of pre-designed size

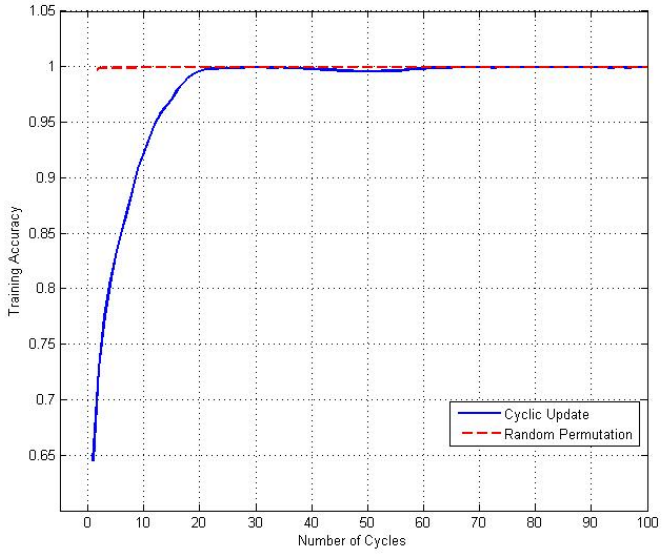

Figure 4.4: Training Accuracy of BCD

Figure 4.5 shows the relative error defined in (4.10) for the BCD algorithm, with both cyclic and random update rules. As the figure shows, the cyclic BCD has a linear rate of convergence. The randomized algorithm shows an even faster convergence behavior.

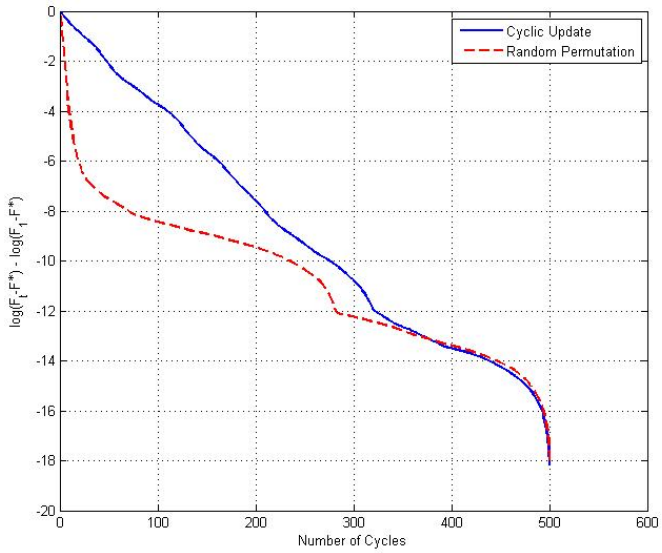

Figure 4.5: Convergence Rate of BCD for SVM

Chapter 5

Conclusion

In this thesis we have introduced the class of approximate proximal splitting methods and established its linear convergence under some conditions (sufficient decrease and local error bound). This general result implies the linear convergence of the BCD algorithm for a class of non-smooth convex problems. As a future work, it will be interesting to generalize the proofs of linear convergence for the APS algorithms to the problems with nuclear norm regularization [48], [49].

References

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

[2] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 68:49–67, 2006.

[3] L. Meier, S. Van De Geer, and P. Buhlmann. The group lasso for logistic regression. *Journal of Royal Statistical Society: Series B*, 70(1):53–71, 2008.

[4] F. Bach. Consistency of the group lasso and multiple kernel learning. *The Journal of Machine Learning Research*, 9:1179–1225, 2008.

[5] S. Ma, X. Song, and J. Huang. Supervised group lasso with applications to microarray data analysis. *BMC bioinformatics*, 8(1):60, 2007.

[6] D. Kim, S. Sra, and I. Dhillon. A scalable trust-region algorithm with applications to mixed-norm regression. *International Conference of Machine Learning (ICML)*, 1, 2010.

[7] J. Liu, S. Ji, and J. Ye. Slep: Sparse learning with efficient projections. *Arizona State University*, 2009.

[8] V. Roth and B. Fischer. The group-lasso for generalized linear models: Uniqueness of solutions and efficient algorithms. *In Proceedings of the 25th International Conference on Machine Learning, ACM*, pages 848–855, 2008.

[9] E. Van Der Berg, M. Schmidt, M. Friedlander, and K. Murphy. Group sparsity via linear-time projection. *Technical Report TR-2008-09, Department of Computer Science, University of British Columbia*, 2008.

[10] S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transaction on Signal Processing*, 57:2479–2493, 2009.

[11] J. B. Rosen. The gradient projection method for non-linear programming, part I linear constraints. *Journal of Society of Industrial and Applied Mathematics*, 8(1):181–217, March 1960.

[12] J. B. Rosen. The gradient projection method for non-linear programming, part II non-linear constraints. *Journal of Society of Industrial and Applied Mathematics*, 9(4):514–532, December 1961.

[13] P. Tseng. Approximation accuracy, gradient methods, and error bound for structured convex optimization. *Mathematical Programming*, (Technical Report), 2009.

[14] Z.-Q. Luo and P. Tseng. Error bounds and convergence analysis of feasible descent methods: A general approach. *Annals of Operations Research*, 46:157–178, 1993.

[15] Z.-Q. Luo and P. Tseng. On the linear convergence of descent methods for convex essentially smooth minimization. *SIAM Journal on Control and Optimization*, 3(2):408–425, March 1992.

[16] Z.-Q. Luo and P. Tseng. Analysis of an approximate gradient projection method with applications to the back-propagation algorithm. *Optimization Methods and Software, Special Issue Neural Networks via Mathematical Programming*, 4:85–101, 1994.

[17] O. L. Mangasarian. Convergence of iterates of an inexact matrix splitting algorithm for the symmetric monotone linear complementarity problem. *SIAM Journal on Optimization*, 1(1):114–122, February 1991.

[18] W. Li. Remarks on the convergence of the matrix splitting algorithm for the symmetric linear complementarity problem. *SIAM Journal on Optimization*, 3(1):155–163, February 1993.

[19] J.-S. Pang. On the convergence of the basic iterative method for the implicit complementarity problem. *Journal of Optimization Theory and Applications*, 37:149–162, 1982.

[20] G. M. Korpelevich. The extra-gradient method for finding saddle points and other problems. *Ekon. i Mat Metody, Trasnlated to English as Matecon*, 12:747–756, 1976.

[21] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4:1168–1200, 2005.

[22] H. Zhang, J. Jiang, and Z.-Q. Luo. On the linear convergence of proximal splitting method for a class of nonsmooth convex minimization problems. *Journal of Operations Research Society of China*, 1:163–186, June 2013.

[23] R. S. Dembo and U. Tulowizki. Local convergence analysis for successive inexact quadratic programming methods. *Working Paper, School of Organization and Management, Yale University, New Haven*, 1984.

[24] J.-S. Pang. Inexact newton methods for nonlinear complementarity problem. *Math. Prog.*, (36):54–71, 1986.

[25] J.-S. Pang. A posteriori error bounds for the linearly-constrained variational inequality problem. *Math. Oper. Res.*, (12):474–484, 1987.

[26] Z.-Q. Luo and P. Tseng. Error bound and convergence analysis of matrix splitting algorithm for the affine variational inequality problem. *SIAM Journal on Optimization*, 2(1):43–54, February 1992.

[27] P. Tseng. Convergence of block coordinate method for non-differentiable minimization. *Journal of Optimization Theory and Appplications*, 109(3):475–494, June 2001.

[28] A. Agarwal, S.N. Negahban, and M.J. Wainwright. Fast global convergence of gradient methods for high-dimensional statistical recovery. *Annals of Statistics*, 40(5):2452–2482, 2012.

[29] M. Schmidt, N.L. Roux, and F. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. *Arxiv preprint arXiv:1109.2415*, 2011.

[30] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

[31] Z.-Q. Luo and P. Tseng. On the linear convergence of descent methods for convex essentially smooth minimization. *SIAM Journal on Control & Optimization*, (30), March 1992.

[32] M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[33] J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *arXiv preprint arXiv:1402.4419*, 2014.

[34] Z. Lu and L. Xiao. On the complexity analysis of randomized block-coordinate descent methods. *arXiv preprint arXiv:1305.4723*, 2013.

[35] I. Necoara and D. Clipici. Distributed random coordinate descent method for composite minimization. *arXiv preprint arXiv:1312.5302*, 2013.

[36] I. Necoara and A. Patrascu. A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. *Computational Optimization and Applications*, pages 1–31, 2013.

[37] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[38] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, pages 1–38, 2012.

[39] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin. Parallel coordinate descent for l1-regularized loss minimization. *arXiv preprint arXiv:1105.5379*, 2011.

[40] I. Necoara. Random coordinate descent algorithms for multi-agent convex optimization over networks. *IEEE Transactions on Automatic Control*, 58:2001–2012, 2013.

[41] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *arXiv preprint arXiv:1212.0873*, 2012.

[42] A. Saha and A. Tewari. On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, 23(1):576–601, 2013.

[43] A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

[44] M. Hong, X. Wang, M. Razaviyayn, and Z.-Q. Luo. Iteration complexity analysis of block coordinate descent methods. *arXiv preprint arXiv:1310.6957*, 2013.

[45] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[46] D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[47] S. Paul, C. Boutsidis, M. Magdon-Ismail, and P. Drineas. Random projections for linear support vector machines. *arXiv preprint arXiv:1211.6085*, 2012.

[48] J.-F. Cai, E. J. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, pages 1956–1982, January 2010.

[49] K.-C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, 6:615–640, 2009.

[50] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

Appendix A

Proof of Lemma 3

We define $h(\alpha) = \|\tilde{\nabla} f(\boldsymbol{x}, \alpha)\|$ for any $\alpha > 0$. Hence, the first part of the lemma is to show that $\alpha h(\alpha)$ is increasing with α.
From the definition of the proximity operator (2.4) and the proximal gradient (2.8), we have

$$\alpha h(\alpha) = \left\| \boldsymbol{x} - \arg\min_{\boldsymbol{y} \in X} \left\{ \alpha f_1(\boldsymbol{y}) + \frac{1}{2} ||\boldsymbol{y} - (\boldsymbol{x} - \alpha \nabla f_2(\boldsymbol{x}))||^2 \right\} \right\|.$$

By the change of variable $\boldsymbol{z} \triangleq \boldsymbol{y} - \boldsymbol{x}$, we have

$$\begin{aligned} \alpha h(\alpha) &= \left\| \arg\min_{\boldsymbol{z} \in X'} \left\{ \alpha f_1(\boldsymbol{x} + \boldsymbol{z}) + \frac{1}{2} \|\boldsymbol{z} + \alpha \nabla f_2(\boldsymbol{x})\|^2 \right\} \right\| \\ &= \left\| \arg\min_{\boldsymbol{z} \in X'} \left\{ \alpha f_1(\boldsymbol{x} + \boldsymbol{z}) + \frac{1}{2} \|\boldsymbol{z}\|^2 + \alpha \boldsymbol{z}^T \nabla f_2(\boldsymbol{x}) \right\} \right\|, \end{aligned} \tag{A.1}$$

where $X^{'} = \{\boldsymbol{z} | \boldsymbol{z} = \boldsymbol{y} - \boldsymbol{x} \text{ for some } \boldsymbol{y} \in X\}$. Then, we have

$$\alpha h(\alpha) = \left\| \arg\min_{\boldsymbol{z} \in X'} \left\{ \alpha g(\boldsymbol{z}) + \frac{1}{2} \|\boldsymbol{z}\|^2 \right\} \right\|, \tag{A.2}$$

where $g(\boldsymbol{z}) = f_1(\boldsymbol{x} + \boldsymbol{z}) + \boldsymbol{z}^T \nabla f_2(\boldsymbol{x})$ is a (non-smooth) convex function. Our goal now is to show that if $0 < \alpha_1 < \alpha_2$, then

$$\alpha_1 h(\alpha_1) = \|\boldsymbol{z}^*(\alpha_1)\| \leq \|\boldsymbol{z}^*(\alpha_2)\| = \alpha_2 h(\alpha_2)$$

where $\boldsymbol{z}^*(\alpha)$ denotes the optimal solution of

$$\min_{\boldsymbol{z} \in X'} \left\{ \alpha g(\boldsymbol{z}) + \frac{1}{2} \|\boldsymbol{z}\|^2 \right\}. \tag{A.3}$$

The optimality of $\boldsymbol{z}^*(\alpha_1)$ implies that

$$g(\boldsymbol{z}^*(\alpha_1)) + \frac{1}{2\alpha_1}\|\boldsymbol{z}^*(\alpha_1)\|^2 \leq g(\boldsymbol{z}) + \frac{1}{2\alpha_1}\|\boldsymbol{z}\|^2, \ \ \forall \boldsymbol{z} \in X^{'}$$

In particular, when $\boldsymbol{z}$ is set to $\boldsymbol{z}^*(\alpha_2)$, we have

$$g(\boldsymbol{z}^*(\alpha_1)) + \frac{1}{2\alpha_1}\|\boldsymbol{z}^*(\alpha_1)\|^2 \leq g(\boldsymbol{z}^*(\alpha_2)) + \frac{1}{2\alpha_1}\|\boldsymbol{z}^*(\alpha_2)\|^2. \tag{A.4}$$

Similarly, the optimality of $\boldsymbol{z}^*(\alpha_2)$ implies that

$$g(\boldsymbol{z}^*(\alpha_2)) + \frac{1}{2\alpha_2}\|\boldsymbol{z}^*(\alpha_2)\|^2 \leq g(\boldsymbol{z}^*(\alpha_1)) + \frac{1}{2\alpha_2}\|\boldsymbol{z}^*(\alpha_1)\|^2. \tag{A.5}$$

Adding up the last two equations yields that

$$\left(\frac{\alpha_2 - \alpha_1}{2\alpha_1\alpha_2}\right)\|\boldsymbol{z}^*(\alpha_1)\| \leq \left(\frac{\alpha_2 - \alpha_1}{2\alpha_1\alpha_2}\right)\|\boldsymbol{z}^*(\alpha_2)\|.$$

Since $0 < \alpha_1 < \alpha_2$, the above inequality implies that $\|\boldsymbol{z}^*(\alpha_1)\| \leq \|\boldsymbol{z}^*(\alpha_2)\|$. Note that the convexity of f_1 (or equivalently g) was not used in this part.

Next we prove the second part of the lemma which states that $h(\alpha)$ is monotonically decreasing with α. Introducing the new variable $\boldsymbol{u} \triangleq \frac{1}{\alpha}\boldsymbol{z}$, the equation (A.2) can be rewritten as

$$h(\alpha) = \left\| \arg\min_{\boldsymbol{u} \in X^{''}} \left\{ \alpha g(\alpha \boldsymbol{u}) + \frac{1}{2}\alpha^2 \|\boldsymbol{u}\|^2 \right\} \right\|$$

or equivalently,

$$h(\alpha) = \left\| \arg\min_{\boldsymbol{u} \in X^{''}} \left\{ \frac{1}{\alpha} g(\alpha \boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u}\|^2 \right\} \right\| \tag{A.6}$$

where $X^{''} = \{\boldsymbol{u} | \boldsymbol{u} = \frac{1}{\alpha}(\boldsymbol{y} - \boldsymbol{x}),$ for some $\boldsymbol{y} \in X\}$. We define $\boldsymbol{u}^*(\alpha)$ as the optimal solution of

$$h(\alpha) = \min_{\boldsymbol{u} \in X^{''}} \left\{ \frac{1}{\alpha} g(\alpha \boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u}\|^2 \right\}. \tag{A.7}$$

It suffices to show that

$$h(\alpha_1) = \|\boldsymbol{u}^*(\alpha_1)\| \geq \|\boldsymbol{u}^*(\alpha_2)\| = h(\alpha_2),$$

for $0 < \alpha_1 < \alpha_2$. The first order optimality condition of (A.7) at $\boldsymbol{u}^*(\alpha)$ implies

$$\boldsymbol{v} + \boldsymbol{u}^*(\alpha) = 0, \ \ \text{for some } \boldsymbol{v} \in \partial g(\alpha \boldsymbol{u}^*(\alpha)), \tag{A.8}$$

where $\partial g(\alpha \boldsymbol{u}^*(\alpha))$ is the sub-differential set of the function g at the point $\alpha \boldsymbol{u}^*(\alpha)$. Rewriting (A.8) for $\boldsymbol{u}^*(\alpha_1)$ and $\boldsymbol{u}^*(\alpha_2)$, we obtain

$$\boldsymbol{v}_1 + \boldsymbol{u}^*(\alpha_1) = 0, \quad \text{for some } \boldsymbol{v}_1 \in \partial g(\alpha_1 \boldsymbol{u}^*(\alpha_1)), \tag{A.9}$$

$$\boldsymbol{v}_2 + \boldsymbol{u}^*(\alpha_2) = 0, \quad \text{for some } \boldsymbol{v}_2 \in \partial g(\alpha_2 \boldsymbol{u}^*(\alpha_2)). \tag{A.10}$$

Since g is a convex function, ∂g is a monotone mapping [50]. Therefore, $\boldsymbol{v}_1 \in \partial g(\alpha_1 \boldsymbol{u}^*(\alpha_1))$ and $\boldsymbol{v}_2 \in \partial g(\alpha_2 \boldsymbol{u}^*(\alpha_2))$ imply

$$\langle \alpha_1 \boldsymbol{u}^*(\alpha_1) - \alpha_2 \boldsymbol{u}^*(\alpha_2), \boldsymbol{v}_1 - \boldsymbol{v}_2 \rangle \geq 0. \tag{A.11}$$

Combining (A.11) with (A.9) and (A.10) implies that

$$\langle \alpha_1 \boldsymbol{u}^*(\alpha_1) - \alpha_2 \boldsymbol{u}^*(\alpha_2), \boldsymbol{u}^*(\alpha_2) - \boldsymbol{u}^*(\alpha_1) \rangle \geq 0.$$

Define $\boldsymbol{d} = \boldsymbol{u}^*(\alpha_2) - \boldsymbol{u}^*(\alpha_2)$. Then the above inequality can be written as

$$\langle\, (\alpha_1 - \alpha_2)\boldsymbol{u}^*(\alpha_1) - \alpha_2 \boldsymbol{d}, \boldsymbol{d} \,\rangle \geq 0$$

which yields

$$\alpha_2 \|\boldsymbol{d}\|^2 \leq (\alpha_1 - \alpha_2)\langle \boldsymbol{u}^*(\alpha_1), \boldsymbol{d} \rangle.$$

Since $\alpha_1 - \alpha_2 < 0$, we have

$$\langle \boldsymbol{u}^*(\alpha_1), \boldsymbol{d} \rangle \leq \frac{\alpha_2}{\alpha_1 - \alpha_2} \|\boldsymbol{d}\|^2. \tag{A.12}$$

Now we can write that

$$\begin{aligned}
\|\boldsymbol{u}^*(\alpha_2)\|^2 &= \|\boldsymbol{u}^*(\alpha_1)\|^2 + 2\langle \boldsymbol{u}^*(\alpha_1), \boldsymbol{d} \rangle + \|\boldsymbol{d}\|^2 \\
&\leq \|\boldsymbol{u}^*(\alpha_1)\|^2 + \frac{2\alpha_2}{\alpha_1 - \alpha_2} \|\boldsymbol{d}\|^2 + \|\boldsymbol{d}\|^2 \\
&= \|\boldsymbol{u}^*(\alpha_1)\|^2 + \frac{\alpha_1 + \alpha_2}{\alpha_1 - \alpha_2} \|\boldsymbol{d}\|^2 \\
&\leq \|\boldsymbol{u}^*(\alpha_1)\|^2,
\end{aligned}$$

where the first inequality is due to (A.12) and the second inequality is due to the fact that $0 < \alpha_1 < \alpha_2$. This completes the proof.