STOCHASTIC ALTERNATING OPTIMIZATION METHODS FOR SOLVING LARGE-SCALE MACHINE LEARNING PROBLEMS

By

KAICHENG WU

A dissertation submitted to the

Graduate School—Newark

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Operations Research

written under the direction of

Dr. Xiaodong Lin and Dr. Andrzej Ruszczyński

and approved by

_____

_____

_____

_____

Newark, New Jersey

May, 2018

ABSTRACT OF THE DISSERTATION

Stochastic Alternating Optimization Methods For Solving Large-Scale Machine Learning Problems

By KAICHENG WU

Dissertation Director:

Dr. Xiaodong Lin and Dr. Andrzej Ruszczyński

In this dissertation, we propose two stochastic alternating optimization methods for solving structured regularization problems, which have been widely used in machine learning and data mining. The first algorithm is called Stochastic Alternating Linearization (SALIN), which is an stochastic extension of the Alternating Linearization (ALIN) in solving convex optimization problems with complex non-smooth regularization term. SALIN linearizes the loss function and penalty function alternatively at each iteration, based on the stochastic approximation of sub-gradients. By applying a special update test at each iteration, and a carefully designed sub-gradients update scheme, the algorithm achieves fast and stable convergence. The update test just relies on a fixed pre-defined set, and we show that the choice of the test set has little influence on the overall performance of the algorithm. Therefore SALIN is a robust method.

The other algorithm is called preconditioned stochastic Alternating Direction Method of Multipliers, which is specially designed to handle structured regularized regression problems such as Fused LASSO, but with the design matrix being ill-conditioned. We prove its $O(1/\sqrt{t})$ convergence rate for general convex functions and $O(\log t/t)$ for strongly convex functions, and show that the constant depends only on the lower dimension of the data

matrix.

We present results of extensive numerical experiments for structured regularization problems such as Fused LASSO and graph-guided SVM, with both synthetic and real-world datasets. The numerical results demonstrate the efficacy and accuracy of our methods.

Acknowledgments

I would like to express my most sincere appreciation to my advisors, Professor Xiaodong Lin and Professor Andrzej Ruszczyński, for their dedication and encouragement in my PhD life. I still remember five years ago when I attended optimization courses taught by Prof. Ruszczyński, I was deeply impressed by the beauty of the rigorous yet interesting optimization theory. I was also fortunate enough to have discussions with Prof. Lin during seminars and symposiums to learn about statistical learning. I am truly grateful to them for inspiring me to pursue PhD in optimization and statistical learning.

During my PhD study, Prof. Lin gives me so many interesting projects to work on, from which I have gained precious experiences in both practical and theoretical aspects. He always encourages me to keep up with the most recent research trends in statistical machine learning, and gives me guidance to understand and answers any of my questions. He also gives greatest support to my internship and job search.

Prof. Ruszczyński plays the most fundamental role in my dissertation. His series of optimization courses inspire me and gives me a solid knowledge foundation. He gives me precious guidance and insightful discussions of my research, and always encourages me to pursue creative ideas. His commitment to excellence, optimistic altitudes towards life, and great sense of humor also touch me a lot and will benefit my life.

I also want to thank Professor Mert Gurbuzbalaban for his guidance and advices in both theoretical proof part and numerical experiments; and thank Professor Darinka Dentcheva for the advice and suggestion to my dissertation. I also want to thank all the professors and staff members in MSIS department and RUTCOR, for providing such an excellent research environment. I also feel grateful to for time spent with my colleagues and friends, for the discussion and happy moments.

Most of all, I would like to thank my Mom and Dad. They raised me up with their

uttermost care, love and hard work. They give me unconditional support for studying abroad, and sacrifice the time that could have been spent with me. I want to thank my loving and supportive wife who accompanies me with my PhD life.

Dedication

To my wife and my parents.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Preliminaries

1.1 Introduction

Structured regularization is a very useful modeling technique in machine learning and data mining. Large number of problems and applications can benefit from structured regularization, such as regression, classification, prediction, pattern recognition. The use of regularization will reduce the solution's over-fitting to training data and thus improve its generalization on unseen data; and will also guarantee the solvability when system is underdetermined. In certain applications such as signal processing in electrical engineering or image processing in biology and medicine, the regularizer is derived from domain knowledge, thus regularization is crucial to the feasibility of the solution and the solvability of the problem. Structured regularization problems have a general form of minimizing a sum of two functions:

$$\min_x : f(x) + h(x) \tag{1.1}$$

where $f(\cdot)$ is the loss function such as squared loss, and $h(\cdot)$ is the penalty function, such as $\|Gx\|_1$, where G encodes structural prior information to the specific problem.

Solving such problems is usually non-trivial, especially when the regularization part $h(\cdot)$ has a complex form. When the structural matrix G has simple form, such as $G = I$ in LASSO problem, traditional path algorithm and coordinate descent algorithm can efficiently solve the optimization problem (Friedman, Hastie, Hoefling, & Tibshirani, 2007). These methods have being widely implemented in various optimization software packages. However, for a complex G matrix, these methods are usually no longer applicable, due to the non-separable nature of the regularization function.

Many deterministic methods have been proposed to solve (1.1), including operator splitting methods (Douglas & Rachford, 1956; Lions & Mercier, 1979; Eckstein & Bertsekas, 1992; Combettes, 2009; Bauschke & Combettes, 2011), and their dual version: Alternating Direction method of Multipliers (ADMM) (Gabay & Mercier, 1976; Glowinski & Tallec, 1989; Boyd, Parikh, Chu, Peleato, & Eckstein, 2010). The Alternating Linearization method (ALIN) (Kiwiel, Rosa, & Ruszczyński, 1999) is an extension of the operator splitting method and the bundle methods (Kiwiel, 1985; Ruszczyński, 2006) in non-smooth optimization; it incorporates an additional improvement test to guarantee the monotonic decrease of the objective function. ALIN has been applied to various structured regularization problems such as fused LASSO or image reconstruction, and exhibits fast and stable convergence as well as scalability to high dimensions (Lin, Minh, & Ruszczyński, 2014). Recently, a multi-block extension called Selective Alternating Linearization method (SLIN) has been proposed by (Du, Lin, & Ruszczyński, 2017), which proved global convergence for an arbitrary number of operators without artificial duplicates of variables; the rate of convergence is derived as well.

In recent years, with the explosive growth of data volume, optimization problems tend to deal with datasets of large size. In those optimization problems, the loss function is usually expressed to represent the average of loss over all observations:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

where n is the sample size which is assumed to be very large. In this "big-data" setting, deterministic methods become computationally expensive, because they process full samples at each pass. Stochastic methods, on the other hand, at each pass deal with a random sample or a batch of samples. They consume less memory than the deterministic methods and make more updates for the same amount of data processed. They have potential to converge to neighbor of the optimal solution much faster. For the above reasons, stochastic methods are very popular in modern computing, for example in training artificial neural networks, stochastic gradient descent (SGD) and its variants including Momentum method, Adaptive sub-gradient descent (Adagrad), Adaptive Moment Estimation (Aadm) (Qian, 1999; Duchi, Hazan, & Singer, 2011; Kingma & Ba, 2014) have been widely used. And for solving

structural regularization problems where the objective function contains a non-separable penalty term, many stochastic optimization methods have been proposed: including Online Alternating Direction Method of Multipliers (ADMM) (Wang & Banerjee, 2012), stochastic ADMM (Ouyang, He, Tran, & Gray, 2013), etc. Though popular, those stochastic methods have some drawbacks. They usually suffer from high variance at tail because of the randomness of samples. They use the gradient information only to update the solution, but not to update the approximation of function, which makes them less competitive compared to operator splitting or bundle methods when dealing with functions with highly complicated form.

In this dissertation, we propose two algorithms to solve structured regularization problems. The first one is Stochastic Alternating Linearization (SALIN), a stochastic extension of the Alternating Linearization method (Kiwiel et al., 1999). SALIN alternatively linearizes component functions from the approximated subgradient built from random samples, and thus saves the memory usage per iteration. By a carefully designed approximation scheme and update test, SALIN is able to achieve fast convergence speed and stabilized convergence at the tail. The other method is Preconditioned Stochastic ADMM, which uses preconditioner derived by randomized linear algebra techniques.

We also pay specific attention to the ubiquitous ℓ_1 and ℓ_2 regression problem with regularization, but with ill-conditioned design matrix. In such cases, generic methods without preconditioning would require excessive number of iterations to converge or even suffer from non-convergence. Recently a stochastic gradient descent with preconditioning called pwSGD (Yang, Chow, Ré, & Mahoney, 2016) has been proposed to solve ill-conditioned ℓ_1 and ℓ_2 regression problems and achieves good results. With the utilization of preconditioning techniques, both of our proposed methods can handle the ill-conditioned regularization problems and achieve good performance.

1.2 Problem Formulation

The optimization problem we consider here has the following general form:

$$\min_x : f(x) + h(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + h(x) \tag{1.2}$$

where $f(\cdot)$ and $h(\cdot)$ are convex, but not necessarily continuously differentiable. Usually $f_i(\cdot)$ represents the loss function of each observation $i \in [1, ...n]$; and $h(\cdot)$ is the regularizer that imposes structure to solution x. Popular choices of loss function include squared loss or absolute deviation for regression problems, and hinge loss or cross entropy loss for classification problems. Popular choices of regularization function $h(\cdot)$ include ℓ_2 ridge, ℓ_1 LASSO, elastic-nets (convex combination of ℓ_2 and ℓ_1) or generalized LASSO.

We can also formulate the problem into a stochastic optimization problem:

$$\min_x \frac{1}{n}\mathbb{E}_{\xi\sim P}[f(x,\xi)/p_\xi] + h(x) \tag{1.3}$$

where a sequence of identical and independent (i.i.d.) observations can be drawn from the random vector ξ, which follows a fixed but unknown distribution $P = \{p_i\}_{i=1}^n$. When ξ is drawn uniformly random, i.e., $P = \{1/n\}_{i=1}^n$, the problem has simpler form:

$$\min_x \mathbb{E}_{\xi\sim P}[f(x,\xi)] + h(x) \tag{1.4}$$

We draw specific attention to the overdetermined ill-conditioned ℓ_2 and ℓ_1 regularized regression problem:

$$\min_x \frac{1}{2n}\|Wx - b\|_2^2 + h(x) \quad \text{or} \quad \min_x \frac{1}{n}\|Wx - b\|_1 + h(x)$$

where $W \in \mathbb{R}^{n\times d}(n \gg d)$ is the design matrix, possibly ill-conditioned, $h(x)$ is a convex regularization term, not necessarily continuously differentiable. A popular example is generalized LASSO (Tibshirani & Taylor, 2011):

$$\min_x \frac{1}{2n}\|Wx - b\|_2^2 + \lambda\|Gx\|_1 \tag{1.5}$$

1.3 Mathematical Notations

We denote x^T and W^T for the transpose of vector x and matrix W. For a positive definite matrix $D \in \mathbb{R}^{d\times d}$, we define the D-norm of a vector x as $\|x\|_D := \sqrt{x^T D x}$. We use $\|\cdot\|_\diamond$ to denote the general norm, and its dual norm is expressed as $\|\cdot\|_*$; when not specified, $\|\cdot\|$ means Euclidean norm $\|\cdot\|_2$. We use $\langle\cdot,\cdot\rangle$ to denote the inner product in a finite dimensional Euclidean space. Whole population is denoted by $\Omega = \{i|i = 1, ..., n\}$. The cardinality of a set S is $|S|$, which represents the number of elements of that set.

1.4 Dissertation Outline

Chapter two will give a literature review to the type of alternating optimization methods, including operator splitting methods, alternating direction method of multiplier and its stochastic variants, as well as the alternating linearization method. The optimization methods used in solving their sub-problems will also be reviewed. We will also give a brief review of the preconditioning technique. Chapter three will introduce our proposed preconditioned stochastic ADMM algorithm, its application on ill-conditioned regularized regression problem, and its rate of convergence analysis. Chapter four presents our main algorithm: the Stochastic ALIN, and its application to LASSO and generalized LASSO regularizations. Chapter five will present results of numerical experiments, including LASSO, generalized LASSO, sparse signal reconstruction, graph-guided support vector machine (SVM), etc. Chapter six contains the conclusion of the dissertation.

Chapter 2

Review of related optimization methods

2.1 Alternating Direction Method of Multipliers (ADMM)

The Alternating Direction Method of Multipliers (ADMM) (Gabay & Mercier, 1976; Glowinski & Tallec, 1989) is a method that minimizes the sum of two convex functions. It was developed in the 1970s and receives increasing attention recent years because of its capability to carry out large-scale distributed optimization (Boyd et al., 2010). ADMM is closely related to many other methods, such as dual decomposition, the method of multipliers, DouglasRachford splitting, Spingarns method of partial inverses, Dykstras alternating projections, Bregman iterative algorithms for ℓ_1 problems and proximal methods (Douglas & Rachford, 1956; Lions & Mercier, 1979; Gabay & Mercier, 1976); see (Boyd et al., 2010) Chapter 2 and 3 for details.

ADMM has been widely studied and many theoretical results on its properties have been established. The global convergence was established in (Glowinski & Tallec, 1989; Eckstein & Bertsekas, 1992). The rate of convergence has been shown to be $O(1/T)$ for general convex problems (He & Yuan, 2012), where T is the number of iteration; and $O(\alpha^T), \alpha \in (0, 1)$, when the objective functions are strongly convex and smooth (Hong & Luo, 2017; Deng & Yin, 2016). ADMM has been shown to perform well in a wide range of real-world applications such as compressed sensing (Yang & Zhang, 2011), image restoration (Goldstein & Osher, 2009), matrix estimation and matrix completion (Y. Xu, Yin, Wen, & Zhang, 2011; F. Xu, Huang, & Wen, 2015; Goldfarb, Ma, & Scheinberg, 2013), etc.

Deterministic ADMM has to compute the empirical loss function from the whole training set at each iteration, which makes its computational complexity to be proportional to the number of samples. Thus the method is unable to handle large-scale problems. To overcome this issue, a number of stochastic versions of ADMM were proposed, which process only

a random sample (or a mini-batch) of data at each iteration. Online ADMM (Wang & Banerjee, 2012) replaces the objective function with an online function at each iteration, and establishes regret bounds for objective function and constraints violation separately. However, the x-optimization step in Online ADMM is not easy to solve. To address this issue, Stochastic ADMM (Ouyang et al., 2013) further replaces the online loss function with its first order approximation at current solution point. The sub-gradient used for linearization is constructed from a uniformly randomly drawn data sample; moreover, a time-varying proximal term is added to the online loss function to guarantee convergence. The two methods can both achieve convergence rate of $O(1/\sqrt{T})$ for general convex objective functions and $O(\log T/T)$ for strongly convex case.

To introduce ADMM, we rewrite (1.2) in the following form:

$$\begin{aligned} &\min_{x,y} f(x) + h(y) \\ &\text{s.t. } Ax + By - c = 0 \end{aligned} \tag{2.1}$$

It differs from the usual problem representation in a way that variable x is split into two parts: x and y, which are related by the linear equation constraint $Ax + By = c$. In our case, $B = -I, c = 0$, while A and the function $h(.)$ depend on the regularization term. In the simplest case, we can use $A = G$ and keep the same $h(\cdot)$ as in (1.2). The augmented Lagrangian of the problem is given by:

$$\mathcal{L}_\rho(x, y, z) = f(x) + h(y) + z^T(Ax + By - c) + \frac{\rho}{2}||Ax + By - c||_2^2 \tag{2.2}$$

where z is the dual variable or called Lagrangian multiplier, and $\rho > 0$ is called the penalty parameter, which is also used as step size. As a precursor of ADMM, the method of multipliers has the following form:

$$(x_{k+1}, y_{k+1}) \leftarrow \arg\min_{x,y} \mathcal{L}_\rho(x, y, z_k)$$

$$z_{k+1} \leftarrow z_k + \rho(Ax_{k+1} + By_{k+1} - c)$$

where x and y are minimized jointly.

2.1.1 Deterministic ADMM

Compared to the method of multipliers, ADMM minimizes x and y in an alternating fashion, which accounts for the term *alternating direction*. The **Deterministic ADMM** algorithm

mainly consists of the three repeated iterates: the x-minimization step, the y-minimization step, and a dual variable z update step. The method has the following form:

Algorithm 1 Deterministic ADMM

1: Initialize y_0 and $z_0 = 0$.
2: **for** $k = 0, 1, 2, ...$ **do**
3: $\quad x_{k+1} \leftarrow \arg\min_x \left\{ f(x) + \frac{\rho}{2}||Ax + By_k - c + z_k/\rho||^2 \right\}$
4: $\quad y_{k+1} \leftarrow \arg\min_y \left\{ h(y) + \frac{\rho}{2}||Ax_{k+1} + By - c + z_k/\rho||^2 \right\}$
5: $\quad z_{k+1} \leftarrow z_k + \rho(Ax_{k+1} + By_{k+1} - c)$
6: **end for**

For general convex problems, deterministic ADMM can achieve a convergence rate of $O(1/T)$ (He & Yuan, 2012), and linear convergence if objective functions are strongly convex and smooth (Hong & Luo, 2017; Deng & Yin, 2016).

There is a variant called **Generalized ADMM**, which adds a proximal term into the x-optimization step:

$$x_{k+1} \leftarrow \arg\min_x \left\{ f(x) + \frac{\rho}{2}\|(Ax + By_k - c) + z_k/\rho\|^2 + \frac{1}{2}\|x - x_k\|_G^2 \right\} \tag{2.3}$$

where G is positive semidefinite. When $G = 0$, it is exactly the deterministic ADMM (1). When $G = \frac{1}{\eta}I - \rho A^T A$, it is equivalent to the following variant (2.4) called **Linearized ADMM**:

$$x_{k+1} \leftarrow \arg\min_x \left\{ f(x) + \rho(x - x_k)^T[A^T(Ax_k + By_k - c + z_k/\rho)] + \frac{\|x - x_k\|^2}{2\eta} \right\} \tag{2.4}$$

where the quadratic term has been linearized at x_k. This approximation may help in cases where the x-minimization problem does not produce a closed form solution, e.g., when $f(x) = \|x\|_1$ and $A \neq I$; and when $f(\cdot)$ is differentiable, (2.4) reduces to a gradient step for unconstrained x.

2.1.2 Stochastic ADMM

When the data set is large, solving the x-minimization step is computationally expensive, and to overcome this issue, stochastic variants of ADMM have been proposed recently. One **stochastic ADMM** algorithm (Ouyang et al., 2013) uses stochastic linear approximation of f, and an ℓ_2-norm prox-function with time-varying step-size η_k at each iteration. The

x-minimization step is then given by:

$$x_{k+1} \leftarrow \arg\min_x \left\{ f(x_k) + \langle f'(x_k, \xi_{k+1}), x \rangle + \frac{\rho}{2} \|(Ax + By_k - c) + u_k\|^2 + \frac{\|x - x_k\|^2}{2\eta_{k+1}} \right\} \tag{2.5}$$

where ξ_{k+1} is a randomly drawn sample, and $u = z/\rho$ is the scaled Lagrangian variable. From now on, we will use scaled Lagrangian variable u for simpler representation. version for simplicity. In (2.5), the first-order approximation of f simplifies the nonlinear programming step, and the monotonically decreasing factor η_k would guarantee convergence. For unconstrained problems, stochastic-ADMM gives the x-update step as:

$$x_{k+1} = \left(\frac{1}{\eta_{k+1}} I + \rho A^\top A \right)^{-1} \left[\frac{x_k}{\eta_{k+1}} - \nabla f(x_k, \xi_{k+1}) - \rho A^\top (By_k - c + u_k) \right]$$

This is a symmetric linear system, which can be solved by the conjugate gradient method. For large-scale problems, the time-varying step-size η_k can be replaced by a fixed one to reduce computation complexity, while still maintaining convergence.

One drawback of this stochastic ADMM algorithm is that even if we use fixed step size, we still need to calculate the matrix-vector multiplication which has $O(d^2)$ cost per iteration.

There is another variant of the stochastic ADMM called **Online Proximal Gradient Descent ADMM (OPG-ADMM)** (Suzuki, 2013). The author deals with the special case of $B = -I, c = 0$, but it can be extended to general ADMM equation constraint. In OPG-ADMM, both $f(x)$ and the quadratic term $\|Ax_k - y_k + u_k\|^2$ are linearized at current solution point x_k in favor of computational efficiency. Similar to (2.3), a general form of the x-update step is:

$$x_{k+1} \leftarrow \arg\min_x \left\{ \langle \nabla f(x_k, \xi_{k+1}), x \rangle + \frac{\rho}{2} \|Ax_k - y_k + u_k\|^2 + \frac{\|x - x_k\|^2_{G_{k+1}}}{2} \right\} \tag{2.6}$$

When $G_{k+1} = \frac{1}{\eta_{k+1}} I - \rho A^T A \succ 0$, solving the above linear system can be avoided. The x-update will be effectively reduced to:

$$x_{k+1} \leftarrow \arg\min_x \left\{ \langle \nabla f(x_k, \xi_{k+1}), x \rangle + \rho (x - x_k)^T [A^T (Ax_k - y_k + u_k)] + \frac{\|x - x_k\|^2}{2\eta_{k+1}} \right\} \tag{2.7}$$

This is a type of inexact ADMM (Boyd et al., 2010). This step can be expressed in an SGD type projection step:

$$x_{k+1} = \Pi_{\mathcal{X}} \left[x_k - \eta_{k+1} \left(\nabla f(x_k, \xi_{k+1}) + \rho A^\top (Ax_k - y_k + u_k) \right) \right] \tag{2.8}$$

Note that for this implementation, a fixed step-size η_k satisfying $\eta_k < \frac{1}{\rho\|A^TA\|}$ could be used and still achieves good convergence.

2.2 Operator Splitting Method

Both ADMM and ALIN stem from the operator splitting method which was invented for more than a half century ago. The operator splitting method and its variants were widely used to solve ordinary differential equations (ODE) and partial differential equations (PDE). Consider problem (1.2), the optimal solution $\hat{x}$ would satisfy

$$\mathbf{0} \in \partial f(\hat{x}) + \partial h(\hat{x})$$

thus we can view the original minimization problem (1.2) as finding zero of sum of two maximum monotone operators:

$$\mathbf{0} \in (F + H)(x) \tag{2.9}$$

where $F(\cdot) = \partial f(\cdot), H(\cdot) = \partial h(\cdot)$. The definition of maximum monotone operator is given as follows:

Definition 2.1. Maximum Monotone Operator. Suppose $u \in F(x)$ and $v \in F(y)$; F is monotone if $(u-v)^T(x-y) \geq 0, \forall (x,y), (u,v) \in F$. F is *maximum monotone* if there is no monotone operator that properly contains it.

The Peaceman-Rachford (Peaceman & Rachford, 1955) and Douglas-Rachford method (Douglas & Rachford, 1956) for solving (2.9) are listed separately as follows:

$$x_{k+1} \leftarrow (I + \rho H)^{-1}(I - \rho F)(I + \rho F)^{-1}(I - \rho H)x_k \tag{2.10}$$

$$x_{k+1} \leftarrow (I + \rho H)^{-1}[(I + \rho F)^{-1}(I - \rho H) + \rho H]x_k \tag{2.11}$$

where ρ is a positive parameter and I is identity matrix.

For solving problem (1.2), the Peaceman-Rachford method works as follows:

Algorithm 2 Peaceman-Rachford

1: Initialize $\hat{x}, \tilde{x}_f, s_f$.
2: **repeat**
3: Linearize $f(\cdot)$: $\tilde{f}(x) = f(\tilde{x}_f) + s_f^T(x - \tilde{x}_f)$
4: $\tilde{x}_h \leftarrow \arg\min_x \left\{ \tilde{f}(x) + h(x) + \frac{1}{2\rho}\|x - \hat{x}\|^2 \right\}$
5: $s_h \leftarrow -s_f - \frac{1}{\rho}(\tilde{x}_h - \hat{x})$
6: $\hat{x} \leftarrow \tilde{x}_h$
7: Linearize $h(\cdot)$: $\tilde{h}(x) = f(\tilde{x}_h) + s_h^T(x - \tilde{x}_h)$
8: $\tilde{x}_f \leftarrow \arg\min_x \left\{ f(x) + \tilde{h}(x) + \frac{1}{2\rho}\|x - \hat{x}\|^2 \right\}$
9: $s_f \leftarrow -s_h - \frac{1}{\rho}(\tilde{x}_f - \hat{x})$
10: $\hat{x} \leftarrow \tilde{x}_f$
11: **until** Stopping test passed

In the Peaceman-Rachford method, the proximal center $\hat{x}$ always gets updated after every proximal step (step 3 and 6). However, global convergence is not guaranteed for general two maximum monotone operators.

If we remove the update step say after step 4, we get the Douglass-Rachford method:

Algorithm 3 Douglass-Rachford

1: Initialize $\hat{x}, \tilde{x}_f, s_f$.
2: **repeat**
3: Linearize $f(\cdot)$: $\tilde{f}(x) = f(\tilde{x}_f) + s_f^T(x - \tilde{x}_f)$
4: $\tilde{x}_h \leftarrow \arg\min_x \left\{ \tilde{f}(x) + h(x) + \frac{1}{2\rho}\|x - \hat{x}\|^2 \right\}$
5: $s_h \leftarrow -s_f - \frac{1}{\rho}(\tilde{x}_h - \hat{x})$
6: Linearize $h(\cdot)$: $\tilde{h}(x) = f(\tilde{x}_h) + s_h^T(x - \tilde{x}_h)$
7: $\tilde{x}_f \leftarrow \arg\min_x \left\{ f(x) + \tilde{h}(x) + \frac{1}{2\rho}\|x - \hat{x}\|^2 \right\}$
8: $s_f \leftarrow -s_h - \frac{1}{\rho}(\tilde{x}_f - \hat{x})$
9: $\hat{x} \leftarrow \tilde{x}_f$
10: **until** Stopping test passed

Note that the order of $f(\cdot)$ and $h(\cdot)$ can be switched, so the method where $\hat{x}$ updates happen only after step 6 but never after step 9, is also a Douglass-Rachford method.

The Peaceman-Rachford and Douglass-Rachford methods have been developed and analyzed (Lions & Mercier, 1979; Eckstein & Bertsekas, 1992; Combettes, 2009; Bauschke & Combettes, 2011), with variants such as the version with over- and under- relaxation when updating proximal center. Global convergence is guaranteed for Douglass-Rachford method, but not for Peaceman-Rachford method; and the convergence is expressed based on monotonicity of the distance to the optimal solution of the problem (Lions & Mercier, 1979; Eckstein & Bertsekas, 1992). Note that operator splitting methods are not monotonic with respect to the objective function value.

2.3 Alternating Linearization Method

The Alternating Linearization Method (ALIN) (Kiwiel et al., 1999) is based on the operator splitting method, and uses some ideas from the bundle method (Kiwiel, 1985) of non-smooth optimization. The most important improvement of ALIN over operator splitting method is the introduction of a special update test, and thus the monotonicity of objective function value is guaranteed. After solving the h or f proximal step, ALIN uses an update test to decide whether to update the proximal center $\hat{x}$ or not. So during each iteration, the proximal center $\hat{x}$ can get updated once, twice, or not at all, depending on the update criterion.

We give the outline of ALIN as follows:

The update test after h-problem has the following form:

$$f(\tilde{x}_h) + h(\tilde{x}_h) \leq (1-\gamma)[f(\hat{x}) + h(\hat{x})] + \gamma[\tilde{f}(\tilde{x}_h) + h(\tilde{x}_h)]$$

for $0 < \gamma < 1$. Before the update test, the following stop test is applied first to determine whether to terminate the algorithm.

$$\tilde{f}(\tilde{x}_h) + h(\tilde{x}_h) \geq f(\hat{x}) + h(\hat{x}) - \epsilon$$

The stop and update test after f-problem has a similar form, where h-function is linearized instead and $\tilde{x}_h$ is replaced by $\tilde{x}_f$.

The update test basically tests whether the current candidate $\tilde{x}_h$ (or $\tilde{x}_f$) has enough improvement on objective value; and the proximal center $\hat{x}$ will change to current candidate

Algorithm 4 Alternating Linearization Method (ALIN)

Initialize $\hat{x}, \tilde{x}_f, s_f$.
repeat
  Linearize $f(\cdot)$: $\tilde{f}(x) = f(\tilde{x}_f) + s_f^T(x - \tilde{x}_f)$
  $\tilde{x}_h \leftarrow \arg\min_x \left\{ \tilde{f}(x) + h(x) + \frac{1}{2}\|x - \hat{x}\|_D^2 \right\}$
  $s_h \leftarrow s_f - D(\tilde{x}_h - \hat{x})$
  if Update test passed **then**
    $\hat{x} \leftarrow \tilde{x}_h$
  end if
  Linearize $h(\cdot)$: $\tilde{h}(x) = f(\tilde{x}_h) + s_h^T(x - \tilde{x}_h)$
  $\tilde{x}_f \leftarrow \arg\min_x \left\{ f(x) + \tilde{h}(x) + \|x - \hat{x}\|_D^2 \right\}$
  $s_f \leftarrow s_h - D(\tilde{x}_f - \hat{x})$
  if Update test passed **then**
    $\hat{x} \leftarrow \tilde{x}_f$
  end if
until Stopping test passed

only if the test passed. The introduction of this update-test guarantees the monotone decrease of objective value and also guarantees the global convergence.

The proof of global convergence of ALIN was provided in (Kiwiel et al., 1999). In (Lin et al., 2014), ALIN was compared with other algorithms on different structured regularization problems, and achieved very good results. Compared to others, it shows shorter computational time monotone and stable tail convergence. Recently, (Du et al., 2017) extended two-block ALIN to multi-block case and proved global convergence for an arbitrary number of operators without artificial duplicates of variables; the rate of convergence was derived as well.

2.4 Selected Optimization Methods

In this section we review two optimization methods that are used in solving sub-problems in Stochastic Alternating Linearization method framework.

2.4.1 Coordinate Descent Method

Coordinate descent method (also known as Gauss-Seidel method) is an optimization algorithm that successively minimizes along coordinate direction to find the minimum of a function. At each iteration, coordinate descent (CD) method determines one coordinate (or

one coordinates block) to solve its univariate minimization problem, while fixing all other coordinates fixed.

Suppose CD is used to solve the problem: $\min_x f(x)$, where $f(\cdot)$ is convex, and $x = (x_1, x_2, ..., x_m)$. Then starting from initial point $x^0 = (x_1^0, x_2^0, ..., x_m^0)$, coordinate descent iteratively solves univariate minimization problem:

$$x_i^{k+1} = \arg\min_{\omega} f(x_1^k, ..., x_{i-1}^k, \omega, x_{i+1}^k, ..., x_m^k) \tag{2.12}$$

The coordinate direction can be chosen in cyclic order or by certain selection rule. The following theorem shows that the sequence generated by CD converges to stationary point when objective function is continuously differentiable.

Theorem 2.1. (Ruszczyński, 2006) Assume function $f(\cdot)$ is continuously differentiable and the set $X_0 = \{x \in \mathbb{R}^d : f(x) \leq f(x^0)\}$ is bounded. Moreover, assume that for every direction (or block) i and $x \in X_0$, the following minimization problem

$$\min_{\omega} f(x_1^k, ..., x_{i-1}^k, \omega, x_{i+1}^k, ..., x_m^k)$$

has a unique solution. Then every accumulation point x^* of the sequence $\{x^k\}$ by the coordinate descent method satisfies the equation $\nabla f(x^*)$.

The block coordinate descent (BCD) is a block extension to the standard CD method, where the indices of coordinates are replaced by the indices of the blocks that they belong to. BCD proceeds in the same way as CD, and is highly efficient for problems where objective function and constraint have partially decomposable structure in terms of decision variable.

2.4.2 Conjugate Gradient Method

Conjugate gradient method can be used to find minimum of a quadratic function:

$$\min_x \frac{1}{2} x^T Q x + c^T x \tag{2.13}$$

where $x \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n \times n}$ is a positive definite matrix. It will successively minimize $f(\cdot)$ along conjugate directions, which are defined as follows:

Definition 2.2. *Q-conjugate.* Let Q be a symmetric positive definite matrix of dimension n. Vectors $d^1, d^2, ..., d^n$ are called *Q-conjugate (Q-orthogonal)* if they are all nonzero and

$$\langle d^i, Qd^j \rangle = 0 \quad \text{for all } i \neq j$$

The conjugate gradient descent algorithm is described in the following:

Algorithm 5 Conjugate Gradient Descent

1: **for** $k = 1, ...$ **do**
2: Calculate $\nabla f(x^k)$.
3: **if** $\nabla f(x^k) = 0$ **then**
4: Stop
5: **end if**
6: Calculate

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

where

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

7: Calculate the next point

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

such that

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

8: **end for**

The following theorem suggests that conjugate gradient descent is able to find the minimum of (2.13) in no more than n steps, where n is dimension of x.

Theorem 2.2. (Ruszczyński, 2006) Assume that $d^1, d^2, ..., d^n$ are conjugate directions and that the sequence $\{x^1, x^2, ..., x^{n+1}\}$ is obtained by successive minimization of function (2.13) in direction $d^k, k = 1, ..., n$:

$$x^{k+1} = x^k + \tau_k d6k,$$

$$f(x^{k+1}) = \min_{\tau \in \mathbb{R}} f(x^k + \tau d^k).$$

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

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

Coordinate descent and conjugate gradient method will serve as building blocks in the framkework of Stochastic Alternating Linearization Method.

Chapter 3

Stochastic ADMM with randomized preconditioning and weighted sampling

3.1 Overview

In this chapter we describe our first proposed method called Stochastic ADMM with randomized preconditioning and weighted sampling (pwSADMM). The method is specifically designed to solved the overdetermined ill-conditioned ℓ_1 and ℓ_2 regularized regression problem:

$$\begin{aligned} &\min_{x,y} \frac{1}{2n}\|Wx-b\|_2^2 + h(y) \\ &\text{s.t. } Ax+By-c=0 \end{aligned} \qquad \text{or} \qquad \begin{aligned} &\min_{x,y} \frac{1}{n}\|Wx-b\|_1 + h(y) \\ &\text{s.t. } Ax+By-c=0 \end{aligned} \tag{3.1}$$

where $W \in \mathbb{R}^{n\times d}(n \gg d)$ is the design matrix, presumed to be ill-conditioned; $h(x)$ is a convex regularization term, not necessarily continuously differentiable. We specifically consider the regression problem with generalized LASSO regularization:

$$\begin{aligned} &\min_{x,y} \frac{1}{2}\|Wx-b\|_2^2 + \lambda\|y\|_1 \\ &\text{s.t. } Gx-y=0 \end{aligned} \qquad \text{or} \qquad \begin{aligned} &\min_{x,y} \|Wx-b\|_1 + \lambda\|y\|_1 \\ &\text{s.t. } Gx-y=0 \end{aligned} \tag{3.2}$$

where G encodes structural prior information to the specific problem. We omit the denominator n for simpler notation in the proof later on; we can always make them equivalent by scaling λ by n.

The method uses two important concepts: randomized preconditioning and non-uniform importance sampling distribution, to deal with ill-conditioned design matrix and guarantee strong performance. The method constructs a preconditioner and an importance sampling distribution, using randomized linear algebra (RLA) (Drineas, Mahoney, Muthukrishnan, & Sarlos, 2011); and then applies ADMM-type iterations on the preconditioned system, with

samples drawn from the importance sampling distribution. The method inherits the strong performance guarantees of RLA and the capability of ADMM of dealing with two-block objective functions.

We show that the asymptotic convergence rate of pwSADMM is as good as any other stochastic ADMM algorithms, i.e., $O(1/\sqrt{t})$ for general convex objective function and $O(\log t/t)$ for strongly convex objective function. Moreover, given a relative error level, the constants in the rate only depend on the lower dimension of the design matrix.

3.2 Preliminaries

Before we describe our proposed algorithm we first specify some notations. We denote by $\kappa(W)$ the conventional condition number of a matrix W. We denote by $|\cdot|_p$ the element-wise ℓ_p norm of a matrix: $|W|_p = (\Sigma_{i,j}|W_{ij}|_p)^{1/p}$. In particular, when $p = 2, |\cdot|_2$ is the Frobenius norm $\|\cdot\|_F$.

Below we discuss two critical ideas used in pwSADMM: well-conditioned basis and leverage scores. The notion of well-conditioned basis was originally introduced by (Clarkson, 2005) as conditioning of ℓ_1 regression problem, and later stated more precisely and extended to ℓ_p problem by (Dasgupta, Drineas, Harb, Kumar, & Mahoney, 2009).

Definition 3.1. ((α, β, p)-conditioning and well-conditioned basis). $W \in \mathbb{R}^{n\times d}$ is (α, β, p)-conditioned if $|W|_p \leq \alpha$ and for all $x \in \mathbb{R}^d, \beta\|Wx\|_p \geq \|x\|_q$, where $1/p + 1/q = 1$. The condition number $\bar{\kappa}_p(W)$ is defined as the minimum value of $\alpha\beta$ such that W is (α, β, p)-conditioned. A basis U of range(W) is well-conditioned if $\bar{\kappa}_p(U)$ is a low-degree polynomial in d, independent of n.

Note that when W is a square matrix, $\bar{\kappa}_p(W)$ is equivalent to the conventional condition number $|W|_p|W^{-1}|_p$. Below is the definition of leverage score for each row of observation in a design matrix.

Definition 3.2. (ℓ_p leverage score). Given $W \in \mathbb{R}^{n\times d}$, suppose U is an (α, β, p) well-conditioned basis of range(W). The i-th leverage score s_i of W is defined as $s_i = \|U_i\|_p^p, \forall i = 1, ..., n$, and $p = 1$ or 2.

Preconditioning

One can found a detailed summary of various ℓ_p preconditioning methods in (Yang, Meng, & Mahoney, 2014, 2016). In our pwSADMM algorithm, we use the following preconditioning procedure similar to (Yang, Chow, et al., 2016). It consists of two steps:

1. Given a matrix $W \in \mathbb{R}^{n\times d}$ with full rank, first build a sketch $SW \in \mathbb{R}^{s\times d}$ for W that satisfies:
$$\sigma_S \cdot \|Wx\|_p \leq \|SWx\|_1 \leq \kappa_S \sigma_S \cdot \|Wx\|_p \tag{3.3}$$
where κ_S is the distortion factor, independent of n.
2. Compute the QR decomposition of SW whose size only depends on d. Return R^{-1}.

The following Lemma guarantees that WR^{-1} is well-conditioned since κ_S and s depend on lower dimension d, independent of higher dimension n.

Lemma 3.1. (Yang, Chow, et al., 2016) Let R be the matrix returned by the above preconditioning procedure, then
$$\bar{\kappa}_p(WR^{-1}) \leq \kappa_S d^{\max\{\frac{1}{2},\frac{1}{p}\}} s^{|\frac{1}{p}-\frac{1}{2}|}.$$

A sketching matrix S satisfying (3.3) can be calculated by various methods in nearly input-sparsity time $O(\text{nnz}(W))$, with the use of random projections (Clarkson & Woodruff, 2013; Meng & Mahoney, 2013; Woodruff & Zhang, 2013). One can find a summary of these sketching methods and resulting time complexity and condition number in (Yang, Chow, et al., 2016).

3.3 Outline of pwSADMM

We now summarize the main steps and implementation details of pwSADMM as follows.

The first step is to compute U such that $U = WR^{-1}$ is a well-conditioned basis for range(W). This can be done using the "sketch then QR decomposition" approach that we just discussed, or through other methods. Note that if R were obtained from the QR-decomposition of the original design matrix W, then U would be a perfectly conditioned basis.

The second step is to obtain leverage score and build importance sampling distribution. The leverage score $\{s_i\}_{i=1}^n$ here is the row norm of U, which can be either calculated exactly (calculate $U = WR^{-1}$ with complexity $O(nd^2)$) or approximated as a by-product of the sketching-based preconditioning methods, see (Yang et al., 2014; Yang, Meng, & Mahoney, 2016) for details. The leverage score of any row i satisfies:

$$(1-\iota)\|U_i\|_p^p \leq s_i \leq (1+\iota)\|U_i\|_p^p \tag{3.4}$$

where $p = 1, 2$ and ι is an approximation factor. When the leverage score is calculated exactly, $\iota = 0$. Based on leverage scores, the non-uniform importance sampling distribution P is defined based on leverage scores:

$$p_i = \frac{s_i}{\Sigma_{i=1}^n s_i}, i = 1, ..., n \tag{3.5}$$

In next step we calculate the preconditioner F, which transforms WF into a well-conditioned base. The preconditioner F can be chosen as R^{-1}, or diagonal matrix that scales R to have unit column norms; or just an identity matrix, same as no preconditioning. $H = (FF^T)^{-1}$ is calculated and cached.

Then the ADMM iterative phase starts. At each iteration a new random sample is drawn according to the defined sampling distribution P. We can also draw a mini-batch according to P, or using volume sampling, which picks a subset of rows with probabilities proportional to the squared ovlumes of the simplicies defined by them. An example of volume sampling algorithm can be found in (Deshpande & Rademacher, 2010).

The loss function is linearized at current solution point using the gradient built from this sample. The x-optimization step is to minimize the sum of linearized f function, quadratic term and proximal term.

We summarize the pwSADMM algorithm below:

Algorithm 6 Preconditioned Weighted Stochastic ADMM

Input: Design matrix $W \in \mathbb{R}^{n\times d}$; penalty matrix G, parameter λ, ρ; Initialized y_0, u_0.

Compute $R \in \mathbb{R}^{d\times d}$ such that $U = WR^{-1}$ is a well-conditioned basis for range(W).

Compute or approximate $\|U_i\|_1$ or $\|U_i\|_2^2$ with leverage scores s_i for $i \in [n]$

Define the distribution $P = \{p_i = s_i/\Sigma s_i\}_{i=1}^n$.

Construct the preconditioner $F \in \mathbb{R}^{d\times d}$ based on R; $H = (FF^T)^{-1}$.

for $k = 0, 1, 2, ..., t$ **do**

  Pick $\xi_{k+1} \in [1...n]$ according to $\{p_i\}_{i=1}^n$.

$$g_{k+1} = \begin{cases} W_{\xi_{k+1}}^T \text{sgn}(W_{\xi_{k+1}} x_k - b_{\xi_{k+1}})/p_{\xi_{k+1}}, & \text{for } \ell_1 \text{ case} \\ W_{\xi_{k+1}}^T (W_{\xi_{k+1}} x_k - b_{\xi_{k+1}})/p_{\xi_{k+1}}, & \text{for } \ell_2 \text{ case} \end{cases}$$

  Update steps:

$$x_{k+1} \leftarrow \arg\min_x \left\{ g_{k+1}^T x + \frac{\rho}{2} ||Ax + By_k - c + u_k||_2^2 + \frac{||x - x_k||_H^2}{2\eta_{k+1}} \right\} \tag{3.6}$$

  $y_{k+1} \leftarrow \arg\min_y \left\{ h(y) + \frac{\rho}{2} \|Ax_{k+1} + By - c + u_k\|_2^2 \right\}$

  $u_{k+1} \leftarrow u_k + (Ax_{k+1} + By_{k+1} - c)$

end for

return $(\bar{x}, \bar{y}, \bar{u})$ for ℓ_1 regression or (x_t, y_t, u_t) for ℓ_2 regression.

3.4 Remarks on the algorithm

Equivalence with applying ordinary ADMM iterations to well-conditioned base

Here we prove that the application of the preconditioned Stochastic ADMM to an ill-conditioned system is equivalent to applying standard Stochastic ADMM to a transformed well-conditioned system.

Suppose we are working on a well-conditioned base $U = WF$ and the variable corresponds to this base is x', where $x = Fx'$. We have the following equivalence:

$$Ux' = WFx' = Wx$$

Rewrite the problem (1.2) in terms of x':

$$\begin{array}{ll} \min\limits_{x,y} f(x)+h(y) \\ \text{s.t. } Ax+By=c \end{array} \iff \begin{array}{ll} \min\limits_{x',y} f(Fx')+h(y) \\ \text{s.t. } AFx'+By=c \end{array} \iff \begin{array}{ll} \min\limits_{x',y} q(x')+h(y) \\ \text{s.t. } (AF)x'+By=c \end{array}$$

where $q(z)=f(Fz)$. If we apply the standard ADMM to this problem, the x'-optimization step will have the following form:

$$\min_{x'}\left\{q(x')+\frac{\rho}{2}||AFx'+By_k-c+u_k||_2^2+\frac{||x'-x'_k||^2}{2\eta_{k+1}}\right\} \tag{3.7}$$

By using the relation $x=Fx'$, this is equivalent to:

$$\min_{x}\left\{f(x)+\frac{\rho}{2}||Ax+By_k-c+u_k||_2^2+\frac{||F^{-1}(x-x_k)||^2}{2\eta_{k+1}}\right\} \tag{3.8}$$

This is exactly the x-optimization step in pwSADMM.

So far, we have shown that applying preconditioned stochastic ADMM to original basis is equivalent to applying a conventional stochastic ADMM to projected well-conditioned basis.

Different choice of preconditioner F

- When $F=R^{-1}$, $H=(FF^\top)^{-1}=R^\top R$ is a good approximation to the Hessian $W^\top W$ since $H=R^\top R=R^\top Q^T QR=(SW)^T(SW)\approx W^TW$. This makes the x-optimization step close to a Newton-type step, which is highly efficient for convex functions but at the cost of doing dense matrix multiplication.

- When $F=D$ is the diagonal matrix that scales R to have unit column norms, $\kappa(RD)$ is always upper-bounded by the original condition number $\kappa(R)$. The computational cost per iteration is only $O(d)$ compared to $O(d^2)$ with dense F matrix. When dimension d is high the diagonal matrix is preferred.

- When $F=I$, then the method reduces to stochastic ADMM.

Comparison with Stochastic ADMM method

We give a brief comparison of our pwSADMM with Stochastic ADMM (SADMM) method of (Ouyang et al., 2013). Recalled the x-optimization step of SADMM:

$$x_{k+1} \leftarrow \arg\min_x \left\{ \langle f'(x_k, \xi_{k+1}), x \rangle + \frac{\rho}{2}\|(Ax + By_k - c) + u_k\|^2 + \frac{\|x - x_k\|_2^2}{2\eta_{k+1}} \right\}$$

where ξ_{k+1} is drawn uniformly random, and the proximal term is a simple half squared norm. In comparison, the x-optimization step in pwSADMM is:

$$x_{k+1} \leftarrow \arg\min_x \left\{ \langle f'(x_k, \xi_{k+1}), x \rangle + \frac{\rho}{2}\|(Ax + By_k - c) + u_k\|^2 + \frac{\|x - x_k\|_H^2}{2\eta_{k+1}} \right\}$$

where ξ_{k+1} is drawn from a sampling distribution defined by leverage scores of design matrix W; and the use of the H-norm in the proximal term acts as preconditioning. Both methods use a random loss function $f(x, \xi)$ to replace the empirical loss$f(x)$, but the difference lies in the way that ξ is sampled. The introduction of $\|\cdot\|_H^2$ norm transforms original system into a well-conditioned system and helps pwSADMM handle ill-conditioned problems. The two improvements over SADMM greatly improve the convergence speed, especially in problems with an ill-conditioned design matrix.

3.5 Theoretical Results

This section is devoted to the analysis of expected convergence rate of the pwSADMM method, and the propositions that serve as basics of main results. First we define some useful notations that appear in our results and proofs:

$$\delta_{k+1} \equiv f'(x_k, \xi_{k+1}) - f'(x_k)$$

$$D_H(\mathcal{X}) \equiv \sup_{x_a, x_b \in \mathcal{X}} \|x_a - x_b\|_H, \quad D_{y^*,B} \equiv \|B(y_0 - y^*)\|$$

$$\mathbf{w}_k = \begin{pmatrix} x_k \\ y_k \\ u_k \end{pmatrix}, \quad \Theta(\mathbf{w}) = \begin{pmatrix} A^T u \\ B^T u \\ c - Ax - By \end{pmatrix}$$

Before we provide the main theorems on convergence rates, we present a proposition about an upper bound of objective value distance and variation of the Lagrangian function based on each iteration points.

Proposition 3.1. (Progress Bound). Define a norm $\|\cdot\|_\diamond$ that satisfies: $\exists\lambda, \text{s.t. } \frac{\lambda}{2}\|\cdot\|_\diamond^2 \leq \frac{1}{2}\|\cdot\|_2^2$. Denote its dual norm as $\|\cdot\|_*$. Then for all $k \geq 1$, we have:

$$f(x_k) + h(y_{k+1}) - f(x) - h(y) + \rho(\mathbf{w}_{k+1} - \mathbf{w})^\top \Theta(\mathbf{w}_{k+1}) \leq \frac{\eta_{k+1}\|Ff'(x_k, \xi_{k+1})\|_*^2}{2\lambda} + \frac{1}{2\eta_{k+1}}(||x_k - x||_H^2 - ||x_{k+1} - x||_H^2) + \frac{\rho}{2}(||Ax + By_k - c||^2 - ||Ax + By_{k+1} - c||^2) + \frac{\rho}{2}(||u - u_k||_2^2 - ||u - u_{k+1}||_2^2) + \langle \delta_{k+1}, x - x_k\rangle \quad (3.9)$$

Based on this proposition we derive the estimate of convergence rate for general convex objective function.

General Convex Function

Theorem 3.1. ($1/\sqrt{t}$ rate for general convex function).

Assume $\mathbb{E}[\|Ff'(x,\xi)\|_*^2] \leq M^2$, where $\|\cdot\|_*$ is the dual norm to $\|\cdot\|_\diamond$ as stated in Proposition (3.1). Let step size $\eta_k = \frac{\sqrt{\lambda}D_H(\mathcal{X})}{M\sqrt{2k}}$, for $\forall t$, we have:

$$\mathbb{E}[f(\bar{x}_t) + h(\bar{y}_t) - f(x^*) - h(y^*) + \varpi||A\bar{x}_t + B\bar{y}_t - c||] \leq \frac{\sqrt{2/\lambda}MD_H(\mathcal{X})}{\sqrt{t}} + \frac{\rho D_{y^*,B}^2 + \rho\varpi^2}{2t} \quad (3.10)$$

We can apply the above result to the ℓ_1 regularized regression problem, where we can further express the constant M exactly. For ℓ_1 problem, define $\|\cdot\|_\diamond \equiv \|\cdot\|_\infty$; note that it satisfies $\frac{1}{2}\|\cdot\|_\infty^2 \leq \frac{1}{2}\|\cdot\|_2^2$. The dual norm of $\|\cdot\|_\infty$ is $\|\cdot\|_* = \|\cdot\|_1$. Thus

$$\begin{aligned}\|Ff'(x,\xi)\|_* = \|Ff'(x,\xi)\|_1 = \|f'(x',\xi)\|_1 &= \|U_\xi RF \cdot \text{sgn}(U_i RFx' - b_\xi)\|_1/p_\xi \\ &= |RF|_1\|U_\xi\|_1\frac{1+\iota}{1-\iota}\cdot\frac{|U|_1}{\|U_\xi\|_1} \leq \alpha|RF|_1\frac{1+\iota}{1-\iota}\end{aligned}$$

where the second equation is due to $\frac{\partial f(x)}{\partial x'} = \frac{\partial f(x)}{\partial x}\cdot\frac{\partial x}{\partial x'}$ and $x = Fx'$; and the last inequality is due to the definition of (α,β,p)-conditioning of basis U: $|U| \leq \alpha$. Thus M is set to be $\alpha|RF|_1\frac{1+\iota}{1-\iota}$.

Therefore, we can see for ℓ_1 regularized regression problem, the convergence rate is $O(1/\sqrt{t})$ and the constants are independent of the higher dimension n.

Strongly Convex Function

When f is strongly convex, the estimate of convergence rate can be further improved to $\log(t)/t$.

Theorem 3.2. ($\log(t)/t$ rate for strongly convex problem)

Assume $f(\cdot)$ is strongly convex with modulus μ, and $\forall x \in \mathcal{X}, \mathbb{E}[\|Ff'(x,\xi)\|_2^2] \leq M^2$. Taking step size $\eta_t = 1/t\mu$, we have:

$$\mathbb{E}[f(\bar{x}_t) + h(\bar{y}_t) - f(x^*) - h(y^*) + \varpi\|A\bar{x}_t + B\bar{y}_t - c\|] \leq \frac{M^2 \log t}{\lambda\mu t} + \frac{\mu D_H^2(\mathcal{X}) + \rho D_{y^*,B}^2 + \rho\varpi^2}{2t} \tag{3.11}$$

For ℓ_2 regression problem where $f(\cdot)$ is least squares loss, we can bound the convexity modulus μ exactly:

$$\mu = 2\sigma_{\min}^2(WF) = \frac{2}{\|((URF)^T URF)^{-1}\|_2} \geq \frac{2}{\|(U^TU)^{-1}\|_2 \cdot \|(RF)^{-1}\|_2^2} = \frac{2}{\beta^2 \cdot \|(RF)^{-1}\|_2^2}.$$

where U is an $(\alpha, \beta, 2)$-conditioned basis of range(W).

3.6 Proofs

3.6.1 Proof for Proposition (3.1)

Lemma 3.2. Let $l(x)$ be a convex differentiable function. Let scalar $s > 0$. Denote the Bregman divergence as $D(x,u) = w(x) - w(u) - \langle \nabla w(u), x - u\rangle$, e.g. $\frac{1}{2}\|x-u\|_H^2$. Consider

$$x^* \equiv \arg\min_x l(x) + sD(x,u) \tag{3.12}$$

then

$$\langle \nabla l(x^*), x^* - x\rangle \leq s[D(x,u) - D(x,x^*) - D(x^*,u)]$$

Proof. From the optimality condition for (3.12), we have

$$\langle \nabla l(x^*) + s\nabla D(x^*,u), x - x^*\rangle \geq 0, \forall x \in \mathcal{X}$$

and thus

$$\begin{aligned}\langle \nabla l(x^*), x^* - x\rangle &\leq \langle \nabla sD(x^*, x_k), x - x^*\rangle \\ &= s\langle \nabla w(x^*) - \nabla w(u), x - x^*\rangle \\ &= s[D(x,u) - D(x,x^*) - D(x^*,u)]\end{aligned}$$

□

Apply the above Lemma to the x-update step of our ADMM algorithm, where the Bregman divergence is $D(x,u) = \frac{1}{2}\|x-u\|_H^2$. We got:

$$\begin{aligned}&\langle f'(x_k,\xi_{k+1}) + \rho A^T[A_{k+1} + By_k - c + u_k], x_{k+1} - x\rangle \\ &\leq \frac{1}{2\eta_{k+1}}(\|x_k - x\|_H^2 - \|x_{k+1} - x\|_H^2 - \|x_k - x_{k+1}\|_H^2)\end{aligned} \tag{3.13}$$

Also due to convexity of f:

$$f(x_k)-f(x) \leq \langle f'(x_k), x_k - x\rangle = \langle f'(x_k,\xi_{k+1}), x_{k+1}-x\rangle + \langle \delta_{k+1}, x - x_k\rangle + \langle f'(x_k,\xi_{k+1}), x_k - x_{k+1}\rangle \tag{3.14}$$

And by combining (3.13) and (3.14) we will have:

$$\begin{aligned}&f(x_k) - f(x) + \langle x_{k+1} - x, \rho A^\top u_{k+1}\rangle \\ =&f(x_k) - f(x) + \langle x_{k+1} - x, \rho A^\top[Ax_{k+1} + By_{k+1} - c + u_k]\rangle \\ \leq&\frac{1}{2\eta_{k+1}}(\|x_k - x\|_H^2 - \|x_{k+1} - x\|_H^2 - \|x_k - x_{k+1}\|_H^2) + \langle \delta_k, x - x_k\rangle + \\ &\langle x - x_{k+1}, \rho A^\top B(y_k - y_{k+1})\rangle + \langle f'(x_k,\xi_k), x_k - x_{k+1}\rangle\end{aligned} \tag{3.15}$$

We deal with the last two terms. For the first term:

$$\begin{aligned}&\langle x - x_{k+1}, \rho A^\top B(y_k - y_{k+1})\rangle = \rho\langle Ax - Ax_{k+1}, By_k - By_{k+1}\rangle \\ &= \frac{\rho}{2}[(\|Ax + By_k - c\|^2 - \|Ax + By_{k+1} - c\|^2) + (\|Ax_{k+1} + By_{k+1} - c\|^2 - \|Ax_{k+1} + By_k - c\|^2)] \\ &\leq \frac{\rho}{2}(\|Ax + By_k - c\|^2 - \|Ax + By_{k+1} - c\|^2) + \frac{\rho}{2}\|u_{k+1} - u_k\|^2\end{aligned} \tag{3.16}$$

and for the last term, using the Fenchel-Young Inequality applied to the conjugate pair $\frac{1}{2}\|\cdot\|_\diamondsuit^2, \frac{1}{2}\|\cdot\|_*^2$ (see (Boyd & Vandenberghe, 2004) Example 3.27):

$$\langle f'(x_k,\xi_k), x_k - x_{k+1}\rangle = \langle Ff'(x_k,\xi_k), F^{-1}(x_k - x_{k+1})\rangle \leq \frac{\eta_k\|Ff'(x_k,\xi_k)\|_*^2}{2\lambda} + \frac{\lambda\|F^{-1}(x_k - x_{k+1})\|_\diamondsuit^2}{2\eta_k} \tag{3.17}$$

and by the assumption that $\frac{\lambda}{2}\|\cdot\|_\diamondsuit^2 \leq \frac{1}{2}\|\cdot\|_2^2$, we have:

$$\frac{\lambda\|F^{-1}(x_k - x_{k+1})\|_\diamondsuit^2}{2} \leq \frac{\|F^{-1}(x_k - x_{k+1})\|_2^2}{2} = \frac{\|x_k - x_{k+1}\|_H^2}{2}$$

Thus we have

$$\begin{aligned}
&f(x_k) - f(x) + \langle x_{k+1} - x, \rho A^\top u_{k+1}\rangle \\
\leq &\frac{1}{2\eta_{k+1}}(\|x_k - x\|_H^2 - \|x_{k+1} - x\|_H^2) + \frac{\eta_{k+1}\|Ff'(x_k, \xi_k)\|_*^2}{2\lambda} + \langle \xi_{k+1}, x - x_k\rangle \\
+&\frac{\rho}{2}(\|Ax + By_k - c\|^2 - \|Ax + By_{k+1} - c\|^2) + \frac{\rho}{2}\|u_{k+1} - u_k\|^2
\end{aligned} \tag{3.18}$$

Now we move to the y-step. Due to convexity of $h(\cdot)$ and optimality condition for the y-update step, we have

$$h(y_{k+1}) - h(y) + \langle y_{k+1} - y, \rho B^\top u_{k+1}\rangle \leq 0 \tag{3.19}$$

and finally according to update rule for dual variable u

$$\begin{aligned}
&\langle (u_{k+1} - u), \rho(Ax_{k+1} + By_{k+1} - c)\rangle \\
=&\rho\langle u_{k+1} - u, u_k - u_{k+1}\rangle \\
=&\frac{\rho}{2}(\|u - u_k\|^2 - \|u - u_{k+1}\|^2 - \|u_{k+1} - u_k\|^2)
\end{aligned} \tag{3.20}$$

By adding them together we have proved Proposition (3.1):

$$\begin{aligned}
f(x_k) + h(y_{k+1}) - f(x) - h(y) + \rho(\mathbf{w}_{k+1} - \mathbf{w})^\top \Theta(\mathbf{w}_{k+1}) \leq \frac{\eta_{k+1}\|Ff'(x, \xi)\|_*^2}{2\lambda} \\
+\frac{1}{2\eta_{k+1}}(||x_k - x||_H^2 - ||x_{k+1} - x||_H^2) + \frac{\rho}{2}(||Ax + By_k - c||^2 - ||Ax + By_{k+1} - c||^2) \\
+\frac{\rho}{2}(||u - u_k||_2^2 - ||u - u_{k+1}||_2^2) + \langle \delta_k, x - x_k\rangle
\end{aligned} \tag{3.21}$$

3.6.2 Proof of Theorem (3.1) ($1/\sqrt{t}$ rate)

Due to convexity of $f(x)$ and $h(y)$ we have $\forall \mathbf{w} \in \mathcal{W}$:

$$\begin{aligned}
&f(\bar{x}_t) + h(\bar{y}_t) - f(x) - h(y) + \rho(\bar{\mathbf{w}}_t - \mathbf{w})^\top \Theta(\bar{\mathbf{w}}_t) \\
&\leq \frac{1}{t}\sum_{k=0}^{t-1}[f(x_k) + h(y_{k+1}) - f(x) - h(y) + \rho(\mathbf{w}_{k+1} - \mathbf{w})^\top \Theta(\mathbf{w}_{k+1})]
\end{aligned} \tag{3.22}$$

then applying Proposition (3.1) at the optimal (x^*, y^*), we have $\forall u$:

$$\begin{aligned}
&f(\bar{x}_t) - f(x_*) + h(\bar{y}_t) - h(y_*) + (\bar{x}_t - x_*)^\top(\rho A^\top \bar{u}_t) \\
&\quad + (\bar{y}_t - y_*)^\top(\rho B^\top \bar{u}_t) + \rho(\bar{u}_t - u)^\top(A\bar{x}_t + B\bar{y}_t - c) \\
&\leq \frac{1}{t}\sum_{k=0}^{t-1}\left[\frac{\eta_k\|Ff'(x_k,\xi_k)\|_*^2}{2\lambda} + \frac{1}{2\eta_k}(\|x_k - x\|_H^2 - \|x_{k+1} - x\|_H^2) + \langle \delta_k, x_* - x_k\rangle\right] \\
&\quad + \frac{\rho}{2t}\left(\|Ax_* + By_0 - c\|^2 + \|u - u_0\|^2\right) \\
&\leq \frac{1}{t}\sum_{k=0}^{t-1}\left[\frac{\eta_k\|Ff'(x_k,\xi_k)\|_*^2}{2\lambda} + \langle \delta_k, x_* - x_k\rangle\right] + \frac{1}{2t}\left(\frac{(D_H\mathcal{X})^2}{\eta_t} + \rho\|B(y_0 - y_*)\|^2 + \rho\|u - u_0\|^2\right)
\end{aligned} \tag{3.23}$$

For the left hand side, $\forall u \in \mathbb{R}^d$, we restrict it in the ball $\mathcal{B}_0 = \{u : \|u\|_2 \leq \varpi/\rho\}$.

$$\begin{aligned}
&\max_{u\in\mathcal{B}_0} f(\bar{x}_t) - f(x_*) + h(\bar{y}_t) - h(y_*) \\
&\quad + (\bar{x}_t - x_*)^\top(\rho A^\top \bar{u}_t) + (\bar{y}_t - y_*)^\top(\rho B^\top \bar{u}_t) + \rho(\bar{u}_t - u)^\top(A\bar{x}_t + B\bar{y}_t - c) \\
&= \max_{u\in\mathcal{B}_0} f(\bar{x}_t) - f(x_*) + h(\bar{y}_t) - h(y_*) - \rho\bar{u}^\top(Ax_* + By_* - c) + \rho u^\top(A\bar{x}_t + B\bar{y}_t - c) \\
&= \max_{u\in\mathcal{B}_0} f(\bar{x}_t) - f(x_*) + h(\bar{y}_t) - h(y_*) + \rho u^\top(A\bar{x}_t + B\bar{y}_t - c) \\
&= f(\bar{x}_t) - f(x_*) + h(\bar{y}_t) - h(y_*) + \varpi\|A\bar{x}_t + B\bar{y}_t - c\|_2
\end{aligned} \tag{3.24}$$

Taking expectation over (3.24), and note that we have a bound for gradient: $\mathbb{E}[\|Ff'(x,\xi)^2\|]_* \leq M^2$ or more strictly, $\|Ff'(x,\xi)\|_* \leq M$; we have:

$$\begin{aligned}
&\mathbb{E}[f(\bar{x}_t) - f(x_*) + h(\bar{y}_t) - h(y_*) + \varpi\|A\bar{x}_t + B\bar{y}_t - b\|_2] \\
&\leq \frac{1}{t}\left(\frac{M^2}{2\lambda}\sum_{k=1}^{t}\eta_k + \frac{(D_H\mathcal{X})^2}{2\eta_t}\right) + \frac{\rho\|B(y_0 - y_*)\|^2}{2t} + \frac{\rho\varpi^2}{2t} + \frac{1}{t}\sum_{k=1}^{t}\mathbb{E}[\langle \delta_k, x_* - x_k\rangle] \\
&= \left(\frac{M^2}{2\lambda t}\sum_{k=1}^{t}\eta_k + \frac{(D_H\mathcal{X})^2}{2t\eta_t}\right) + \frac{\rho\|B(y_0 - y_*)\|^2}{2t} + \frac{\rho\varpi^2}{2t}
\end{aligned} \tag{3.25}$$

Since η_k is monotonically decreasing, there exist:

$$\frac{M^2}{2\lambda t}\sum_{k=1}^{t}\eta_k + \frac{D_{\mathcal{X}}^2}{2t\eta_t} \geq \frac{M^2\eta_k}{2\lambda} + \frac{D_{\mathcal{X}}^2}{2\eta_t}$$

To optimize the rate of convergence, we take step size η_k to be equal to $\frac{D_H\mathcal{X}}{M\sqrt{2t/\lambda}}$. Finally we find the tightest bound of the inequality:

$$\mathbb{E}[f(\bar{x}_t)-f(x_*)+h(\bar{y}_t)-h(y_*)+\varpi\|A\bar{x}_t+B\bar{y}_t-b\|_2] \leq \frac{\sqrt{2}D_H\mathcal{X}M}{\sqrt{\lambda t}}+\frac{\rho\|B(y_0-y_*)\|^2}{2t}+\frac{\rho\varpi^2}{2t} \tag{3.26}$$

3.6.3 Proof of Theorem (3.2) ($\log t/t$ rate)

By the strong convexity of f we have $\forall x$:

$$\begin{aligned} f(x_k)-f(x) &\leq \langle f'(x_k), x_k-x\rangle - \frac{\mu}{2}\|x-x_k\|^2 \\ &= \langle f^{'}(x_k,\xi_k), x_{k+1}-x\rangle + \langle \xi_k, x-x_k\rangle + \langle Ff'(x_k,\xi_k), F^{-1}(x_k-x_{k+1})\rangle - \frac{\mu}{2}\|x-x_k\|_H^2 \end{aligned}$$

similar with the proof for ℓ_1 case, but taking step size $\eta_k = 1/\mu k$, we have:

$$\begin{aligned} &\mathbb{E}[f(\bar{x}_t)+h(\bar{y}_t)-f(x^*)-h(y^*)+\varpi||A\bar{x}_t+B\bar{y}_t-c||_2] \\ &\leq \mathbb{E}\left\{\frac{1}{t}\sum_{k=0}^{t-1}\left[\frac{\eta_k\|Ff'(x_k,\xi_k)\|^2}{2}+\left(\frac{1}{2\eta_k}-\frac{\mu}{2}\right)\|x_k-x^*\|_H^2-\frac{\|x_{k+1}-x^*\|_H^2}{2\eta_k}\right]\right\} \\ &+\frac{\rho\|B(y_0-y^*)\|^2}{2t}+\mathbb{E}\left[\max_{u\in\mathcal{B}_0}\left\{\frac{\rho}{2t}\|u-u_0\|^2\right\}\right] \\ &\leq \frac{M^2}{2\lambda t}\sum_{k=1}^{t}\frac{1}{\mu k}+\frac{1}{t}\sum_{k=0}^{t-1}\mathbb{E}\left[\frac{\mu k}{2}\|x_k-x^*\|_H^2-\frac{\mu(k+1)}{2}\|x_{k+1}-x^*\|_H^2\right]+\frac{\rho\|B(y_0-y^*)\|^2}{2t}+\frac{\rho\varpi^2}{2t} \\ &\leq \frac{M^2\log t}{\lambda\mu t}+\frac{\mu(D_H\mathcal{X})^2}{2t}+\frac{\rho\|B(y_0-y^*)\|^2}{2t}+\frac{\rho\varpi^2}{2t} \end{aligned}$$

This completes our proof.

Chapter 4

Stochastic Alternating Linearization (SALIN)

4.1 Overview

In this section we describe the algorithm of Stochastic Alternating Linearization (SALIN) for solving the following structured regularization problem:

$$\min_x : f(x) + h(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + h(x)$$

where $f(x)$ is the loss function with n being large, and $h(x)$ is the structured regularization term. Both $f(x)$ and $h(x)$ are convex but not necessarily continuously differentiable. SALIN is developed to solve the above optimization problems in the large-size dataset setting.

We first give a brief review of the key features of the deterministic ALIN method (4) and describe the main changes that we have developed with SALIN. ALIN is an iterative method which alternatively solves two sub-problems: h- and f-subproblem corresponding to $\tilde{f}+h$ and $\tilde{h}+f$, where $\tilde{f}, \tilde{h}$ is linear minorant of f, h. It generates two auxiliary sequences $\{\tilde{x}_h\}$ and $\{\tilde{x}_f\}$, and an approximation sequence $\{\hat{x}\}$ that converges to the solution of the problem. At each iteration after solving the h-subproblem, ALIN updates s_h from the optimality condition and uses it for linearizing h in solving the next f-subproblem: $\tilde{h} + f$; however, the point $\hat{x}$ only gets updated if the designed h update test is passed. Analogous procedures will be carried out after the f-subproblem. The update test guarantees the monotone decrease of objective value, which is the main attracting feature of ALIN.

SALIN works in a similar way as ALIN, but differs in several aspects which make SALIN applicable to the stochastic setting. These differences will not only save computational cost per iteration, but also guarantee nice convergence behavior. The differences are mainly affecting two aspects of the method:

- The update mechanism for the sub-gradient of the loss function, i.e., s_f. It is updated

not only when the f-subproblem is solved, but also when a new random sample is drawn.

- The update test being applied after each sub-problem. It evaluates function values on a predefined fixed sample, not on the whole dataset.

The vector s_f is the estimate of sub-gradient of the loss function f. In deterministic ALIN, s_f is initialized as $f'(\tilde{x}_f)$ using the whole dataset; e.g., $s_f^0 = W^T(W\tilde{x}_f - b)$ when $f(x) = \frac{1}{2}\|Wx - b\|_2^2$. However in the stochastic setting, it is impossible to initialize s_f using the whole dataset; s_f can only be approximated gradually as more data are revealed over iterations. Therefore in SALIN, we design a special update schedule for s_f: s_f is updated when f-subproblem is solved, using optimality condition; and is also updated when a new data sample is drawn, using a weighted averaging scheme, as will be discussed in (4.1).

By doing so, SALIN avoids approximating s_f using the whole dataset at once, but manages to gradually obtain an accurate approximation over iterations, at a much lower computation cost per iteration. Moreover, the tail convergence is also stabilized by this update schedule.

And for the update test, deterministic ALIN evaluates function value and its linear approximation value using the whole dataset. In the stochastic setting, this is not possible. Instead, SALIN uses a pre-defined fixed sample S for update tests. We will show that the choice of S has little impact on the final solution; and the larger the size of S, the higher probability that SALIN returns a better solution in terms of objective value and solution distance. In short, SALIN is a robust method under different S.

4.2 Outline of the SALIN Algorithm

We give the description of SALIN here. Similar to ALIN, SALIN is an iterative method which generates an approximation sequence $\{\hat{x}^k\}$ converging to the solution of the problem, and two auxiliary sequences: $\{\tilde{x}_h^k\}$ and $\{\tilde{x}_f^k\}$, where k is the iteration number. There are also three sequences for sub-gradients: candidate s_f from the sample ξ: $\{s_f^k(\xi)\}$; approximated s_f: $\{\hat{s}_f^k\}$; and approximated s_h: $\{\hat{s}_h^k\}$. At the beginning, we initialize $\tilde{x}_f^0 = \hat{x}^0$, the starting point of the method; and also initialize $\hat{s}_f^0 = 0$. We suppress the superscript k from now on

for simplicity.

The h-subproblem

We draw a random variable ξ from $[1, ..., n]$, define $f_\xi(x) \equiv \frac{1}{|\xi|}\sum_{i\in\xi} f_i(x)$. Calculate the candidate sub-gradient $s_f(\xi)$: $s_f(\xi) = \nabla f_\xi(\tilde{x}_f)$ if $f(\cdot)$ is differentiable; $s_f(\xi) \in \partial f_\xi(\tilde{x}_f)$ if $f(\cdot)$ is general convex.

The vector $s_f(\xi)$ is not used directly for linearizing f, instead it is averaged with the previous approximation $\hat{s}_f$ to make a new approximation of s_f:

$$\hat{s}_f \leftarrow (1-\omega)\hat{s}_f + \omega s_f(\xi) \tag{4.1}$$

where $\omega \in [0, 1]$. We can set ω fixed, e.g., $\omega = 0.25$; or make it time-varying. We found that a monotonically decreasing ω accelerates convergence speed.

After updating $\hat{s}_f$, we use it to linearize $f(\cdot)$ at $\tilde{x}_f$:

$$\tilde{f}(x) = f(\tilde{x}_f) + \hat{s}_f^T(x - \tilde{x}_f)$$

The h-subproblem is to solve:

$$\tilde{x}_h^k = \arg\min_x \left\{ \tilde{f}(x) + h(x) + \frac{1}{2\eta}\|x - \hat{x}\|_D^2 \right\} \tag{4.2}$$

where D is a diagonal positive definite matrix, and η is a decaying step size.

Next we calculate the sub-gradient of $h(\cdot)$ at $\tilde{x}_h$, based on the optimality condition for the minimum at (4.2):

$$0 \in \hat{s}_f + \partial h(\tilde{x}_h) + \eta^{-1}D(\tilde{x}_h - \hat{x})$$

which yields the right sub-gradient $\hat{s}_h \in \partial h(\tilde{x}_h)$:

$$\hat{s}_h = -\hat{s}_f - \eta^{-1}D(\tilde{x}_h - \hat{x}) \tag{4.3}$$

The vector $\hat{s}_h$ will be used to linearize $h(\cdot)$ in the next sub-problem.

The decision of whether to make the update $\hat{x} \leftarrow \tilde{x}_h$ depends on the outcome of the update test, which will be explained later.

The f-subproblem

We first construct a linear minorant of the regularization function $h(\cdot)$ (usually non-smooth), using $\hat{s}_h$ obtained from last h-subproblem:

$$\tilde{h}(x) = h(\tilde{x}_h) + \hat{s}_h^T(x - \tilde{x}_h)$$

Then we solve the f-subproblem:

$$\tilde{x}_f = \arg\min_x \left\{ f_\xi(x) + \tilde{h}(x) + \frac{1}{2\eta}\|x - \hat{x}\|_D^2 \right\} \tag{4.4}$$

The minimizer $\tilde{x}_f$ will be used in the linearization of $f(\cdot)$ at the next iteration.

To conclude the current iteration we update $\hat{s}_f$, similar to (4.3):

$$\hat{s}_f = -\hat{s}_h - \eta^{-1}D(\tilde{x}_f - \hat{x}) \tag{4.5}$$

At the beginning of the next iteration when a new sample ξ is drawn, this current $\hat{s}_f$ will be averaged with $s_f(\xi)$ to form an updated approximation $\hat{s}_f$, as in (4.1).

The decision of whether to make the update $\hat{x} \leftarrow \tilde{x}_f$ depends on the outcome of the update test, which is explained in the following section.

The update step

The update step is used to decide whether to change the current solution approximation $\hat{x}$ or not. It can be applied after any of the sub-problems, or after both of them. Here we describe the update step after the h-subproblem; analogous procedures would apply to the update step after the f-subproblem.

The predefined fixed sample S is used for estimating the loss function value $f(\cdot)$, instead of the whole dataset. The test sample S is chosen at random before iteration phase and is not used for solving any subproblems. We denote by $f_S(\cdot)$ the loss function value evaluated on sample S, and $f_\Omega(\cdot)$ the loss function value evaluated on the whole population Ω.

Now we use sample S to estimate the f function value in the update test, i.e., we replace the theoretical inequality of ALIN:

$$f_\Omega(\tilde{x}_h) + h(\tilde{x}_h) \leq (1-\gamma)[f_\Omega(\hat{x}) + h(\hat{x})] + \gamma[\tilde{f}_\Omega(\tilde{x}_h) + h(\tilde{x}_h)] \tag{4.6}$$

with

$$f_S(\tilde{x}_h) + h(\tilde{x}_h) \leq (1-\gamma)[f_S(\hat{x}) + h(\hat{x})] + \gamma[\tilde{f}_S(\tilde{x}_h) + h(\tilde{x}_h)]. \tag{4.7}$$

We need to test how confident we are to update $\hat{x}$ using the update test outcome based on sample S, other than Ω. To do this, we first separate out all terms in 4.6 (or 4.7) related to the $f(\cdot)$-function, and include them in one function $\Phi(\cdot)$:

$$\Phi(\hat{x}, \tilde{x}_h, \gamma) = f(\hat{x}) - f(\tilde{x}_h) - \gamma(f(\hat{x}) - \tilde{f}(\tilde{x}_h)).$$

where $\gamma \in (0,1)$. In our experiments we set $\gamma = 0.2$; the choice of γ does not affect overall performance of SALIN.

We want to test that given $\hat{x}, \tilde{x}_h$, whether $\Phi_\Omega(\hat{x}, \tilde{x}_h, \gamma)$ is statistically different than ϕ_S, where

$$\phi_S \equiv \Phi_S(\hat{x}, \tilde{x}_h, \gamma) = f_S(\hat{x}) - f_S(\tilde{x}_h) - \gamma(f_S(\hat{x}) - \tilde{f}_S(\tilde{x}_h))$$

can be calculated and thus viewed as a deterministic value for each given iteration. To do this, we carry out the following t-test:

Test 4.1. T-Test of $H_0 : \Phi_\Omega(\hat{x}, \tilde{x}_h, \gamma) = \phi_S$.

Pick a random sample $\zeta \in \{i \in \Omega | i \notin S\}$ with size n_ζ. Calculate $\Phi_i(\hat{x}, \tilde{x}_h, \gamma) = f_\zeta(\hat{x}) - f_\zeta(\tilde{x}_h) - \gamma(f_\zeta(\hat{x}) - \tilde{f}_\zeta(\tilde{x}_h))$, $\forall i \in \zeta$. Build the following t-statistics:

$$t = \frac{\frac{1}{n_\zeta}\sum_{i\in\zeta}\Phi_i(\hat{x}, \tilde{x}_h, \gamma) - \phi_S}{\sqrt{\frac{\hat{\sigma}_\zeta^2}{n_\zeta}}} \tag{4.8}$$

where $\hat{\sigma}_\zeta^2$ is the sample variance of $\{\Phi_i(\hat{x}, \tilde{x}_h, \gamma),\ i \in \zeta\}$. The t statistics follows the Student's t distribution when n_ζ is large. We reject the null hypothesis H_0 if $|t| > T_{1-\alpha/2, |\zeta|-1}$, at significance level α.

Based on the two outcomes of the above t-test, we take different actions correspondingly:

- If the null hypothesis of the above t-test is rejected, i.e., $\Phi_\Omega(\hat{x}, \tilde{x}_h, \gamma) \neq \Phi_S(\hat{x}, \tilde{x}_h, \gamma)$ we skip the remainder of the current update step and jump to the next iteration.
- If the null hypothesis is not rejected, we continue with the following stop test and update test.

By taking the above step we can ensure that (4.7) provides close enough approximation to (4.6) and thus can be trusted. The decision of whether to make the update $\hat{x} \leftarrow \tilde{x}_h$ afterwards, is based on the consideration of the whole objective function $f_\Omega(\cdot) + h(\cdot)$, not on the randomness of a particular training sample ξ.

Test 4.2. Stopping test. The stopping test is carried out before the update test. Here ϵ is the stopping parameter independent of S. If:

$$\tilde{f}_S(\tilde{x}_h) + h(\tilde{x}_h) \geq f_S(\hat{x}) + h(\hat{x}) - \epsilon, \tag{4.9}$$

then we terminate the SALIN algorithm and output $\hat{x}$. Otherwise continue.

In practice, we can also use $\|\hat{x} - \tilde{x}_h\| \leq \epsilon$ as the stopping criterion.

Test 4.3. Update test. If (4.7) is satisfied, i.e.,

$$f_S(\tilde{x}_h) + h(\tilde{x}_h) \leq (1-\gamma)[f_S(\hat{x}) + h(\hat{x})] + \gamma[\tilde{f}_S(\tilde{x}_h) + h(\tilde{x}_h)],$$

then $\hat{x} \leftarrow \tilde{x}_h$. Otherwise keep $\hat{x}$ unchanged.

If the update step is applied after f-subproblem, then h is linearized instead, and functions are evaluated at $\tilde{x}_f$ and $\hat{x}$. The final outcome determines whether to update $\hat{x} = \tilde{x}_f$.

Below is the summary of the entire update step after the h-subproblem.

Algorithm 7 Update step after the h-subproblem

Input γ, ϵ and n_ζ.

Define $\Phi(\hat{x}, \tilde{x}_h, \gamma) = f(\hat{x}) - f(\tilde{x}_h) - \gamma(f(\hat{x}) - \tilde{f}(\tilde{x}_h))$.

Calculate $\phi_S = \Phi_S(\hat{x}, \tilde{x}_h, \gamma)$.

Draw a random sample $\zeta \in \{i \in \Omega | i \notin S\}$ with size n_ζ. Calculate $\{\Phi_i(\hat{x}, \tilde{x}_h, \gamma),\ i \in \zeta\}$.

Build the t-statistics:

$$t = \frac{\frac{1}{n_\zeta}\sum_{i\in\zeta}\Phi_i(\hat{x}, \tilde{x}_h, \gamma) - \phi_S}{\sqrt{\hat{\sigma}_\zeta^2/n_\zeta}} \tag{4.10}$$

where $\hat{\sigma}_\zeta^2$ is the sample variance of $\{\Phi_i(\hat{x}, \tilde{x}_h, \gamma),\ i \in \zeta\}$.

if $|t| > T_{1-\alpha/2,\, n_\zeta - 1}$ **then**

    Skip the rest and jump to next iteration.

end if

if $\tilde{f}_S(\tilde{x}_h) + h(\tilde{x}_h) \geq f_S(\hat{x}) + h(\hat{x}) - \epsilon$ **then**

    Terminate SALIN algorithm.

else

    if $f_S(\tilde{x}_h) + h(\tilde{x}_h) \leq (1-\gamma)[f_S(\hat{x}) + h(\hat{x})] + \gamma[\tilde{f}_S(\tilde{x}_h) + h(\tilde{x}_h)]$ **then**

        return $\hat{x} = \tilde{x}_h$

    end if

end if

Summary of SALIN

We have described all the details about the SALIN algorithm, below is a summary:

Algorithm 8 Stochastic Alternating Linearization (SALIN)

Initialize $\hat{x}, \hat{s}_f$ and D.
Initialize S for update tests.
repeat:
    Calculate candidate sub-gradient $s_f(\xi)$ based on a newly drawn sample ξ.
    $\hat{s}_f \leftarrow (1-\omega)\hat{s}_f + \omega s_f(\xi)$, where $\omega \in [0,1]$
    $\hat{x}_h \leftarrow \arg\min_x \{\hat{s}_f^T(x - \tilde{x}_f) + h(x) + \frac{1}{2\eta}\|x - \hat{x}\|_D^2\}$
    $\hat{s}_h \leftarrow \hat{s}_f - \eta^{-1}D(\tilde{x}_h - \hat{x})$
    if h-update test passed **then**
        $\hat{x} \leftarrow \tilde{x}_h$
    end if
    $\tilde{x}_f \leftarrow \arg\min\{f_\xi(x) + \hat{s}_h^T(x - \tilde{x}_h) + \frac{1}{2\eta}\|x - \hat{x}\|_D^2\}$
    $\hat{s}_f \leftarrow \hat{s}_h - \eta^{-1}D(\tilde{x}_f - \hat{x})$
    if f-update test passed **then**
        $\hat{x} \leftarrow \tilde{x}_f$
    end if
until Stop test passed

4.3 Remarks on the algorithm

Discussion of the update step

As we mentioned before, the update test and the $\hat{s}_f$ averaging update schedule will help SALIN stabilize tail convergence. If we implement SALIN without the update test, then it will always update $\hat{x}$ to the noisy approximation of $\tilde{x}_h$ (or $\tilde{x}_f$). However, this leads to very bumpy tail convergence caused by the randomness of samples.

With the update test, SALIN updates $\hat{x}$ less often: it only updates $\hat{x}$ when $\tilde{x}_h$ (or $\tilde{x}_f$) exhibit enough improvement on $f_S(\cdot) + h(\cdot)$. Because the t-test (4.1) guarantees that the summary statistics $\Phi_S(\hat{x}, \tilde{x}_h, \gamma)$ used in update test inequality is close to $\Phi_\Omega(\hat{x}, \tilde{x}_h, \gamma)$,

therefore we are confident to use the test result based on S in lieu of the true but unknown test result based on the whole population Ω.

Discussion of the predefined test sample S

The sample S is chosen uniformly random from the whole population Ω at the beginning. We now argue that different choice of S (assume same size) has little influence on the final solution of the SALIN algorithm. The sub-gradient approximations $\hat{s}_f$ and $\hat{s}_h$ are updated regardless of the outcome of the update test. This means that f and h functions are approximated more and more accurately over iterations towards their true forms, not affected by S. And as long as test sample size $|S|$ is same, $f_S(x)$ of different S gives same interval estimate to $f(x)$. Therefore, different choices of S do not affect final solution directly, and they act similar in the update test.

However, larger size S is better than smaller size S, because $f_S(x)$ gives more accurate estimate to $f(x)$. This also agrees with our numerical experiments findings: for the same problem, SALIN with larger size S will return solutions more concentrated around the true optimal point.

Moreover, because S is not used for training, it will also act as validation set for the SALIN algorithm, and thus reduce over-fitting and improve generalization.

In experiment 5.1.1, we will compare results of SALIN running on different choice of S and different size of S. We also compare SALIN with deterministic methods solving the reduced-size problem defined over those training samples that SALIN uses, and show that SALIN stochastically beats those deterministic methods.

Discussion of the $\hat{s}_f$ update schedule

The use of the $\hat{s}_f$ averaging schedule $\hat{s}_f \leftarrow (1-\omega)\hat{s}_f + \omega s_f(\xi)$ is also critical. It combines two types of updates on s_f:

- Through optimality condition: $\hat{s}_h = -\hat{s}_f - D(\tilde{x}_h - \hat{x})$ and $\hat{s}_f = -\hat{s}_h - D(\tilde{x}_f - \hat{x})$.
- Through the reveal of a new data sample: $s_f(\xi) = f'(\tilde{x}_f, \xi)$.

The first type of update is essential to the bundle method and ALIN, where the f and h are approximated alternatively over iterations. This type of update helps SALIN to handle complex and non-smooth objective functions. The second type of update is essential to most stochastic methods where they can make updates rapidly and as larger number of samples are learned, the aggregate gradient direction will point to the true gradient direction.

The averaging schedule (4.1) combines the advantages of both worlds, to make SALIN handle the stochastic setting and complex objective functions.

Moreover, we find a monotonically decreasing ω makes SALIN perform even better. At the beginning, more weight is put on the $s_f(\xi)$ part, making SALIN benefit from randomness by making greedy descent steps. As the iterations continue, more weight is shifted to existing approximation $\hat{s}_f$ part, thus $\hat{s}_f$ gets better approximated to true s_f and SALIN acts more like a deterministic method. This also has an effect of reducing noise at tail.

4.4 Application to LASSO

We first consider the application of Stochastic Alternating Linearization (SALIN) to the LASSO regression problem, one of the most widely used regularization problems. Because the penalty function is separable, the coordinate descent method is very efficient at solving it (Tseng, 2001), but here we want to demonstrate SALIN's capability to handle the most popular case.

The LASSO regression problem has the following form:

$$\min_x f(x) + h(x) = \frac{1}{2n}\sum_{i=1}^{n}(b_i - W_i x)^2 + \lambda\|x\|_1$$

where $W \in \mathbb{R}^{n\times d}$ is the design matrix, y is the response variable vector, $\lambda > 0$ is the LASSO parameter, and x is the estimator we want to get.

For the proximal term $\frac{1}{2}\|x - \hat{x}\|_D^2$ in the sub-problems, we found the diagonal matrix $D = \text{diag}(W^T W)/n$, i.e., $d_j = W_j^T W_j/n, j = 1, ..., d$, is essential for solving this problem. This has the same flavor with the x-update step in the preconditioned Stochastic ADMM method when the a diagonal D matrix is used. This is also related to the *diagonal quadratic approximation* of the squared error loss function $f(x) = \frac{1}{2n}\|y - Wx\|^2$, which was employed in (Ruszczyński, 1995) in similar objective function of augmented Lagrangian minimization.

Indeed, D is a very good diagonal approximation to the Hessian of $f(\cdot)$. Note that the calculation of the diagonal matrix D does not require explicit form of W^TW, only d times of vector-vector multiplications are required.

Another way to build D is to use a sketched matrix, like we did with the preconditioned Stochastic ADMM method. Here $D = \text{diag}(W_{\text{sk}}^T W_{\text{sk}})/\text{nrow}(W_{\text{sk}})$, where W_{sk} is a pre-calculated sketched matrix of W and $\text{nrow}(\cdot)$ represents number of rows of a matrix. By doing so we would save even more computational cost while maintain similar accuracy. During our numerical studies, we found that for the LASSO regression problem, sampling 10% rows to form sketch matrix W_{sk} is enough for the algorithm to perform well, any higher proportion has negligible marginal improvement.

The h-subproblem

At the beginning of the h-subproblem we first draw a random sample $\xi \in [1, ..., n]$ to form candidate sub-gradient $s_f = \frac{1}{|\xi|} W_\xi^T (W_\xi \tilde{x}_f - y_\xi)$. The updated approximation of $\hat{s}_f$ is given by:

$$\hat{s}_f = (1 - \omega)\hat{s}_f + \omega s_f$$

We can use fixed ω, or use monotonically decreasing ω. For LASSO example, making ω proportional to the decaying step size η is a good choice.

After linearizing the squared loss function f using $\hat{s}_f$, the h-subproblem of LASSO regression reduces to:

$$\tilde{x}_h = \arg\min_x \left\{ \hat{s}_f^T x + \lambda \|x\|_1 + \frac{1}{2\eta} \|x - \hat{x}\|_D^2 \right\} \tag{4.11}$$

It has closed form solution: write $\tau_j = \hat{x}_j - \hat{s}_{fj}\eta/d_j$, we can compute $\tilde{x}_h$ element-wise:

$$\tilde{x}_{hj} = \text{sgn}(\tau_j) \max\left(0, |\tau_j| - \frac{\lambda}{d_j}\right), \quad j = 1, ..., d \tag{4.12}$$

The updated approximation $\hat{s}_h$ is given by $\hat{s}_h = \hat{s}_f - \eta^{-1} D(\tilde{x}_h - \hat{x})$; and we apply the update step to decide whether to update $\hat{x} \leftarrow x_h$.

The f-subproblem

The f-subproblem simplifies to an unconstrained quadratic programming problem:

$$\tilde{x}_f = \arg\min_x \left\{ s_h^T x + \frac{1}{2}\|y_\xi - W_\xi x\|_2^2 + \frac{1}{2\eta}\|x - \hat{x}\|_D^2 \right\} \tag{4.13}$$

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

$$(W_\xi^T W_\xi + \eta^{-1} D)\delta = W_\xi^T (y_\xi - W_\xi \hat{x}) - s_h \tag{4.14}$$

The system can be solved efficiently by preconditioned conjugate gradient method (Golub & Loan, 1996), with the diagonal preconditioned D as discussed before. This is equivalent to applying a conventional conjugate gradient method to a system with a symmetric positive definite matrix $\bar{H} = D^{-\frac{1}{2}} H D^{-\frac{1}{2}}$, whose condition index is bounded by $\sqrt{d}+1$. See Lemma 3 of (Lin et al., 2014) for detailed proofs.

There is another way to solve (4.13) approximately: to further linearize $\frac{1}{2}\|y_\xi - W_\xi x\|_2^2$ at $\hat{x}$ and then apply a gradient step:

$$\tilde{x}_f = \hat{x} - \eta D^{-1} \left(s_h + W_\xi^T (W_\xi \hat{x} - y_\xi) \right) \tag{4.15}$$

where D^{-1} is a diagonal matrix and can be cached before iterations.

4.5 Application to SVM with generalized structural regularization

Stochastic Alternating Linearization is more powerful in solving problems with complex regularization terms, for which methods such as coordinate descent or proximal splitting methods no longer work, since they don't produce a closed form solution like in the simple LASSO case. One example is the generalized LASSO problem (Tibshirani & Taylor, 2011):

$$\min_x \frac{1}{2n}\|b - Wx\|^2 + \|Gx\|_1$$

where the matrix G encodes the structural prior. When $G = I$, it recovers LASSO problem of the previous section; when G is the following $d \times (d-1)$ matrix:

$$G = \begin{bmatrix} 1 & -1 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -1 \end{bmatrix}$$

it recovers the fused LASSO problem.

A concrete example of generalized LASSO is the graph-guided Fused LASSO (GFLasso) framework (Kim, Sohn, & Xing, 2009), a graphical extension of Fused LASSO. In graph-guided Fused LASSO, the structural matrix G encodes the mutual relation of variables which can be expressed by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. In the graph $\mathcal{G}$, $\mathcal{V} = \{v_1, ...v_d\}$ is the set of variables and $\mathcal{E} = \{\mathbf{e}_1, ...\mathbf{e}_{|\mathcal{E}|}\}$ is the set of edges, each edge $\mathbf{e}_k = \{i, j\}$ with weight w_{ij}, where $i, j \in \mathcal{V}$. Thus $G_{ki} = w_{ij}$ and $G_{kj} = -w_{ij}$ for any edge $\mathbf{e}_k = \{i, j\} \in \mathcal{G}$. The graph-guided Fused LASSO has the following form:

$$\min_x \frac{1}{2n} \sum_{i=1}^{n} (b_i - W_i^T x)^2 + \nu \|x\|_1 + \lambda \sum_{\{i,j\} \in \mathcal{E}} w_{ij} |x_i - x_j|$$

In order to perform classification, we replace the squared loss with a non-smooth hinge loss: $f_i(x) = \max\{0, 1 - b_i W_i^T x\} = (1 - b_i W_i^T x)_+$, and replace $\nu\|x\|_1$ with $\frac{\nu}{2}\|x\|_2^2$ to enforce large margin of SVM classifier. After the fusion penalty is added into the objective, the Graph-Guided SVM (known as GGSVM) (Ouyang et al., 2013) has the following objective function:

$$\min_x \frac{1}{n} \sum_{i=1}^{n} [1 - b_i W_i^T x]_+ + \frac{\nu}{2} \|x\|_2^2 + \lambda \|Gx\|_1$$

SALIN is able to handle any general norm $\|\cdot\|_\diamondsuit$, not only the ℓ_1 norm here. We express SALIN for solving SVM with a more general regularization term:

$$\min_x \frac{1}{n} \sum_{i=1}^{n} [1 - b_i W_i^T x]_+ + \frac{\nu}{2} \|x\|_2^2 + \lambda \|Gx\|_\diamondsuit \tag{4.16}$$

We denote f-function by $\frac{1}{n}\sum_{i=1}^{n}(1 - b_i W_i^T x)_+ + \frac{\nu}{2}\|x\|_2^2$, and h-function by $\lambda\|Gx\|_\diamondsuit$; and consider their corresponding sub-problems.

The h-subproblem

We draw a random sample ξ from $[1, ..., n]$ and calculate the candidate sub-gradient $s_f(\xi)$:

$$s_f(\xi) \in \partial_x [1 - b_\xi W_\xi^T \tilde{x}_f]_+ + \nu \tilde{x}_f$$

and the approximation $\hat{s}_f$ is updated by:

$$\hat{s}_f \leftarrow (1 - \omega)\hat{s}_f + \omega s_f(\xi)$$

The h-subproblem has the following form:

$$\min_{x,z} \hat{s}_f^T x + \lambda \|z\|_\diamond + \frac{1}{2\eta}\|x - \hat{x}\|_D^2, \quad \text{s.t.} \quad Gx = z \tag{4.17}$$

Note that for hinge loss function, D is identity matrix; but it can be generalized to any forms for different problems.

The Lagrangian can be formulated as:

$$L(x, z, \mu) = \hat{s}_f^T x + \lambda \|z\|_\diamond + \mu^T (Gx - z) + \frac{1}{2\eta}\|x - \hat{x}\|_D^2$$

where μ is the Lagrangian dual variable. The dual norm of $\|\cdot\|_\diamond$ is defined as:

$$\|\mu\|_* = \max_{\|z\|_\diamond \leq 1} \mu^T z, \qquad \|z\|_\diamond = \max_{\|\mu\|_* \leq 1} z^T \mu \tag{4.18}$$

Notice that $\inf_z L(x, z, \mu)$ is finite with respect to z if an only if $\lambda\|z\|_\diamond - \mu^T z \geq 0$, that is $\|\mu\|_* \leq \lambda$. And under such condition z-terms actually vanish, so we get the reduced form of the Lagrangian:

$$\hat{L}(x, \mu) = s_f^T x + \mu^T G x + \frac{1}{2\eta}\|x - \hat{x}\|_D^2 \tag{4.19}$$

The dual function is given by $\min_x \hat{L}(x, \mu)$, and its solution is:

$$\tilde{x}_h = \hat{x} - \eta D^{-1}(\hat{s}_f + G^T \mu) \tag{4.20}$$

Substituting it back to (4.19) and we can get the dual problem:

$$\max_\mu -\frac{1}{2}\mu^T G D^{-1} G^T \mu + \mu^T G(\eta^{-1}\hat{x} - D^{-1} s_f) \quad \text{s.t.} \quad \|\mu\|_* \leq \lambda \tag{4.21}$$

This is a quadratic programming problem with norm constraints. We discuss the ℓ_1-norm constraint here, which is the case of Graph-Guided SVM.

The dual norm to $\|\cdot\|_1$ is the infinity norm:

$$\|\mu\|_* = \|\mu\|_1 = \max_{1 \leq j \leq d} |\mu_j|$$

which makes (4.21) a box-constrained quadratic programming problem. This can be solved very efficiently by coordinate descent algorithm. The returned solution $\tilde{\mu}$ would be substituted into (4.20) to obtained primal solution $\tilde{x}_h$.

The f-subproblem

$$\tilde{x}_f = \arg\min_x \left\{ \hat{s}_h^T x + [1 - b_\xi W_\xi^T \hat{x}]_+ + \frac{1}{2\eta} \|x - \hat{x}\|_D^2 \right\} \tag{4.22}$$

There is no closed-form solution due to the non-separable nature of the hinge loss, thus (4.22) is solved approximately by taking a sub-gradient step:

$$\tilde{x}_f = \hat{x} - \eta D^{-1} \left(\hat{s}_h + \partial_x [1 - b_\xi W_\xi^T \hat{x}]_+ + \nu \hat{x} \right) \tag{4.23}$$

Chapter 5

Numerical Experiments

In this section, we present results of experiments on different problems, with both simulated and real-world data. We mainly focus on problems with non-differentiable penalty functions. All the studies are performed in an Intel dual core 2.7GHZ, 16GB RAM laptop using MATLAB.

5.1 Fused LASSO

The first example is the Fused LASSO least square regression problem:

$$\min_x \frac{1}{2n}\|y - Wx\|^2 + \lambda\|Gx\|_1, \qquad \lambda > 0 \tag{5.1}$$

where $W \in \mathbb{R}^{n\times d}$ is an ill-conditioned design matrix, and G is the fusion matrix:

$$G = \begin{bmatrix} 1 & -1 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -1 \end{bmatrix}$$

The fusion penalty term penalizes the difference of neighboring coefficients of the estimator x.

The data are generated from a linear regression model: $b = Wx + \epsilon$ with pre-specified coefficient x. To generate the ill-conditioned matrix W, we first generate a diagonal matrix Σ where we can easily control its condition number by setting the diagonal entries. We then calculate W using $W = U\Sigma V^T$, where U and V are orthogonal matrices. For the predefined coefficient x, 10% of the entries are 1, 20% are 2, and the remaining are zero. For example,

with $d = 100$, we have

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

The noise ϵ is drawn from the normal distribution with zero mean and variance equal to 0.01.

5.1.1 SALIN: comparison study of different choices of S

In this section we focus on the Stochastic Alternating Linearization (SALIN) method. We first evaluate the impact of different choices of the predefined test sample S on the final solution of SALIN, and then evaluate the impact of different size of S.

For this study the data setting is as follows: W is $n = 2^{18}$ by $d = 2^8$ matrix, with condition number $\kappa(W) = 26.5$, moderately ill-conditioned; and $\lambda = 10^{-3}$. SALIN is implemented in mini-batch mode with batch-size equal to 32; and the stopping criterion is: $\|\tilde{x}_h - \hat{x}\|_2 \leq 10^{-3}$; γ for both update tests are 0.2. Note that the true optimal value is 8.98×10^{-3}.

Different random samples of S

We fix the test sample size $|S|$ to be 64, and independently sample S for each repetition. We run SALIN for 500 repetitions, and at each repetition we output the objective value $f_\Omega(\hat{x}) + h(\hat{x})$ and the objective value evaluated at sample S : $f_S(\hat{x}) + h(\hat{x})$. Below is the plot of the 500 value pairs, sorted in ascending order of $f_S(\hat{x}) + h(\hat{x})$.

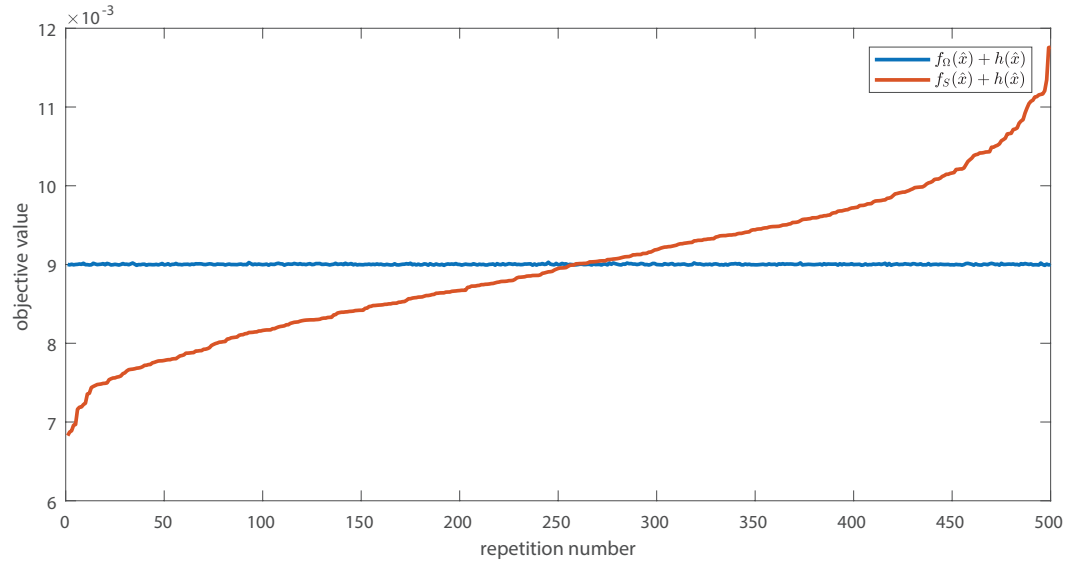

Figure 5.1: Plot of value pairs: $f_\Omega(\hat{x}) + h(\hat{x})$ & $f_S(\hat{x}) + h(\hat{x})$ over 500 repetitions, sorted by $f_S(\hat{x}) + h(\hat{x})$. S is independently drawn for each repetition, with fixed size 64.

We can see that although the values of $f_S(\hat{x}) + h(\hat{x})$ are from a relatively broad range ($0.006 \sim 0.013$) because of randomness, the global objective values $f_\Omega(\hat{x}) + h(\hat{x})$ are concentrated on a much smaller range, and are unrelated to $f_S(\hat{x}) + h(\hat{x})$. The above experimental finding supports the theoretical claim we made in the last chapter: the choice of S has little influence on the quality of the final solution of SALIN.

Different size of S

Next, we study the influence of the test sample size $|S|$ on the final solution and on the running time. We still use the previous experiment setting, and run 500 repetitions for each of test sample size from 2^2 to 2^8. Below is the statistical summary.

$\|S\|$	Mean $\pm$ Standard Deviation	95% quantile	CPU time (s)
4	$0.0093 \pm 5.4 \times 10^{-4}$	0.0107	0.4989
8	$0.0090 \pm 1.5 \times 10^{-4}$	0.0093	0.2255
16	$0.0090 \pm 6.1 \times 10^{-6}$	0.0091	0.1998
32	$0.0090 \pm 5.7 \times 10^{-6}$	0.0090	0.1579
64	$0.0090 \pm 5.8 \times 10^{-6}$	0.0090	0.1546
128	$0.0090 \pm 5.7 \times 10^{-6}$	0.0090	0.1841
256	$0.0090 \pm 6.1 \times 10^{-6}$	0.0090	0.1879

Table 5.1: Summary statistics of objective value, by using different test sample size.

From Table 5.1, we can see that usually large sample size $|S|$ leads to better solution; but once $|S|$ exceeds a certain level, the marginal improvement on the objective is negligible, but the increase in running time is significant. For this study, a moderate sample size $|S|$ from 32 to 64 seems to be the best choice.

Last, we show that no matter what size S has, SALIN stochastically beats the solution of reduced problem, whose loss function is defined on the samples that SALIN observes. Note that for this example, SALIN learns $8122(\approx 3.1\%)$ samples on average before it stops.

To show this, at each repetition, we implement SALIN and keep a record of all the data samples that it observes. We then solve the problem defined on these data samples, using deterministic pre-conditioned ADMM. Based on the analysis from the above table, we just have to show that SALIN with $|S| = 4$ beats the solution on the reduced-problem.

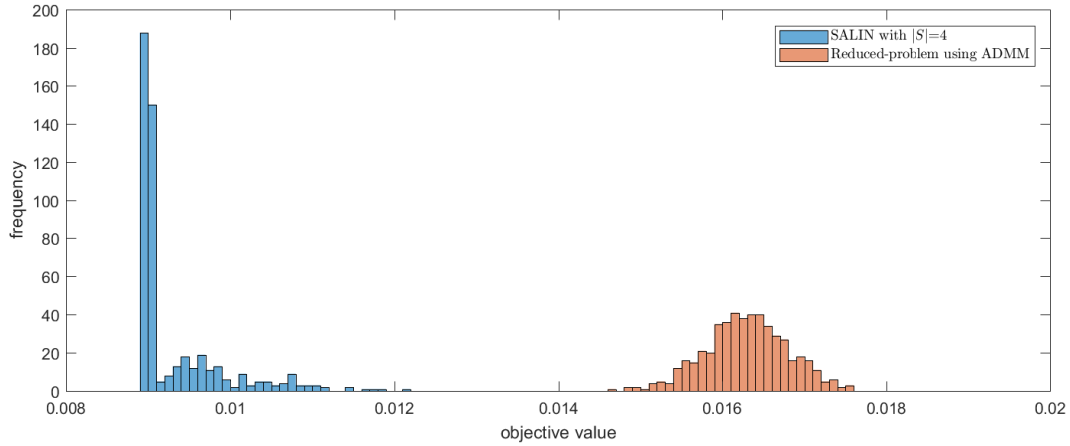

Figure 5.2: Histogram of $f(\hat{x})+h(\hat{x})$ by SALIN and deterministic ADMM on solving reduced problem. 500 repetitions.

We can see that the objective value of SALIN is stochastically smaller than the objective value of the reduced problem. Combining the results with Table 5.1, we can conclude that no matter what the size $|S|$ is, SALIN is better than solving the reduced-problem.

5.1.2 Comparison of stochastic methods

In this section, we evaluate the performance of three stochastic methods for solving the ill-conditioned Fused LASSO regression problem. The three methods are: stochastic alternating linearization (SALIN), pre-conditioned weighted Stochastic Alternating Direction Method of Multipliers (pwSADMM), and Stochastic ADMM.

To make fair comparison, we make the following specifications. All three methods are implemented in mini-batch training mode with batch-size 32. Stopping criterion for SALIN is $\|\hat{x} - \tilde{x}_h\| \leq 10^{-3}$, and $\|x^{k+1} - x^k\| \leq 10^{-3}$ for pwSADMM and SADMM. We use exponential decaying step-size for each method, and tune the step-size separately such that best performance is achieved for each method. For SALIN, we set test sample size $|S| = 32, \gamma = 0.2, n_\zeta = 32, \alpha = 0.05$ for two update test; D matrix is constructed from sketched W_{sk}, where $\text{nrow}(W_{sk}) = 0.01\times\text{nrow}(W)$. For pwSADMM, a diagonal pre-conditioner is used (hence diagonal H in the proximal term $\frac{\|x-x^k\|_H^2}{2\eta_{k+1}}$ at x-step) for low computational cost; and the pre-conditioner is constructed from sketched W with 1% rows,

same as SALIN.

The following table shows the performance of the three methods in different parameter settings. Each method is repeated 500 times on each parameter setting.

Parameters	Methods	Iteration	Time (s)
$n = 2^{18}, d = 2^8$	SALIN	302 ± 12	0.132 ± 0.016
$\kappa(W) = 3.55$	pwSADMM	587 ± 13	0.136 ± 0.010
$\lambda = 10^{-3}$	SADMM	652 ± 20	0.142 ± 0.013
$n = 2^{18}, d = 2^8$	SALIN	344 ± 13	0.155 ± 0.028
$\kappa(W) = 26.5$	pwSADMM	838 ± 14	0.181 ± 0.013
$\lambda = 10^{-3}$	SADMM	1420 ± 27	0.290 ± 0.016
$n = 2^{18}, d = 2^8$	SALIN	347 ± 13	0.166 ± 0.042
$\kappa(W) = 256$	pwSADMM	957 ± 17	0.205 ± 0.052
$\lambda = 10^{-3}$	SADMM	1696 ± 31	0.512 ± 0.283
$n = 2^{18}, d = 2^9$	SALIN	365 ± 13	0.460 ± 0.065
$\kappa(W) = 26.5$	pwSADMM	908 ± 12	0.415 ± 0.100
$\lambda = 10^{-3}$	SADMM	1606 ± 22	0.685 ± 0.360
$n = 2^{18}, d = 2^8$	SALIN	332 ± 13	0.148 ± 0.025
$\kappa(W) = 26.5$	pwSADMM	813 ± 12	0.175 ± 0.013
$\lambda = 10^{-4}$	SADMM	1258 ± 23	0.262 ± 0.015

Table 5.2: Performance comparison of different stochastic methods.

Table 5.2 suggests that both SALIN and pwSADMM are very efficient at solving ill-conditioned Fused LASSO regression problems; while for standard Stochastic ADMM, the running time increases rapidly with the increase of condition number. Moreover, SALIN requires much fewer iterations to converge than the other two ADMM methods in almost all the parameter settings. This is because of the sub-gradient update schedule, and the main philosophy of SALIN: SALIN uses gradient information not only to update solution, but also to approximate functions. If we are able to build a subroutine that calculates function value faster, SALIN will also outperform pwSADMM in terms of running time.

Next we show how these three methods approach the optimal point, on the second parameter setting above: $n = 2^{18}, d = 2^8, \kappa(W) = 26.5, \lambda = 10^{-3}$. We plot the relative error of the objective value, i.e., $(\mathcal{F}(\hat{x}) - \mathcal{F}(x^*))/\mathcal{F}(x^*)$ against iteration and running time, where $\mathcal{F}(\cdot) \equiv f(\cdot) + h(\cdot)$. We also plot the relative distance to the optimal point, i.e., $\|x - x^*\|_2/\|x^*\|_2$ against iteration number and running time.

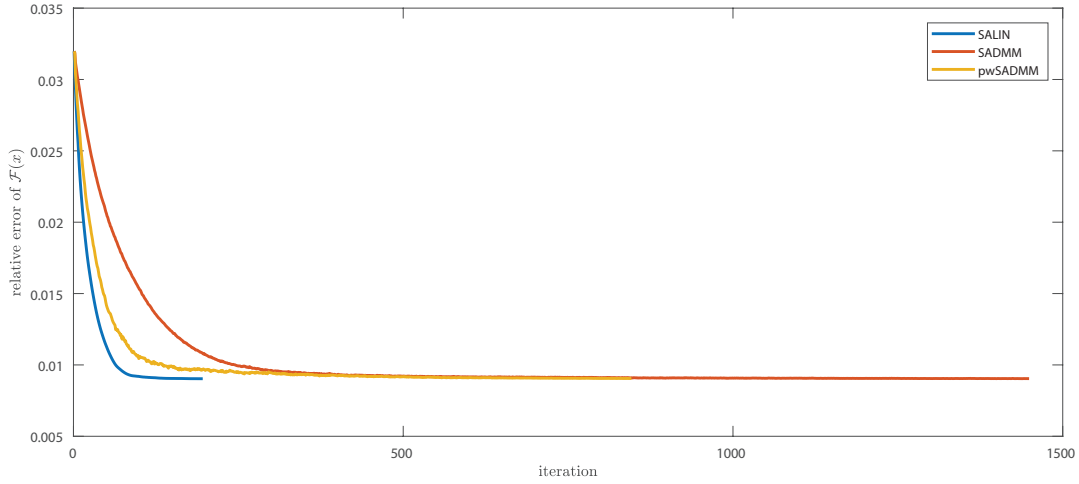

Figure 5.3: Relative error of objective value against iteration number.

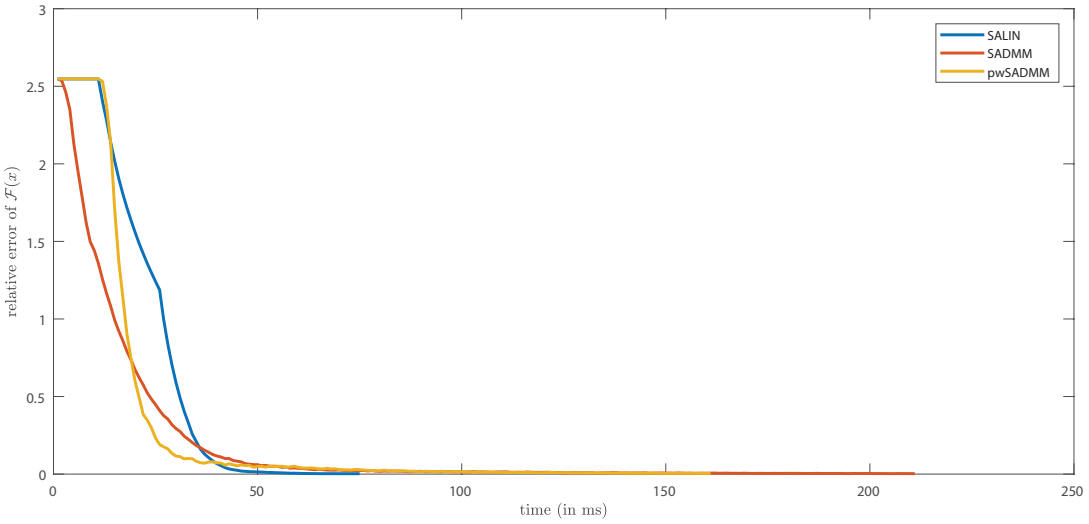

Figure 5.4: Relative error of objective value against CPU time (in ms).

Figure 5.3 shows that SALIN requires least number of iterations to converge, followed by pwSADMM. Standard Stochastic ADMM requires significantly more than the other two. In figure 5.4, we can see that at the beginning, SALIN and pwSADMM require similar

amounts of time to compute the matrices D and H from sketched data matrix W. Once they complete this setting-up phase, they are able to catch up with SADMM quickly and finally beat it. Moreover,SALIN converges very smoothly, whereas the convergence of pwSADMM and ADMM is a little bumpy.

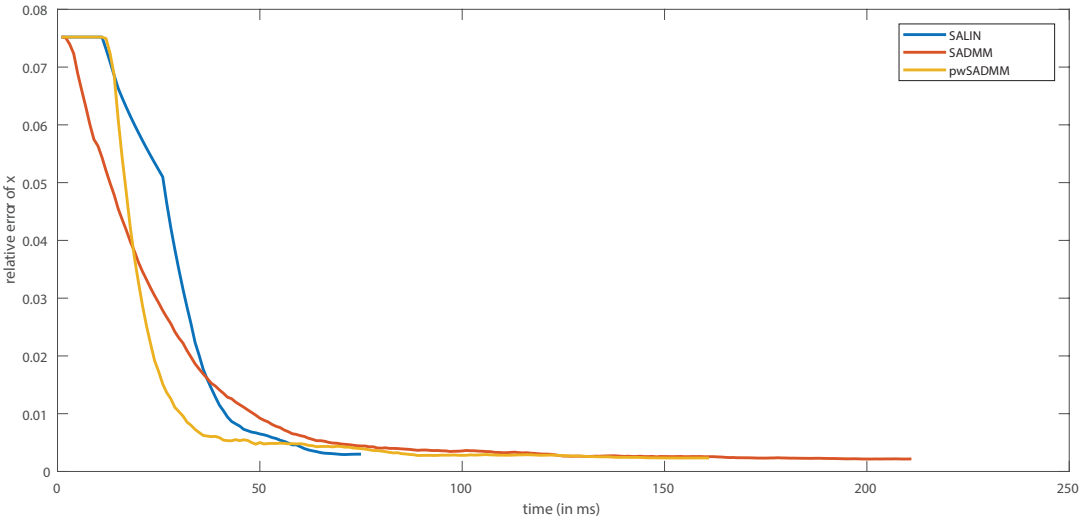

Figure 5.5: Relative distance of x against iteration number.

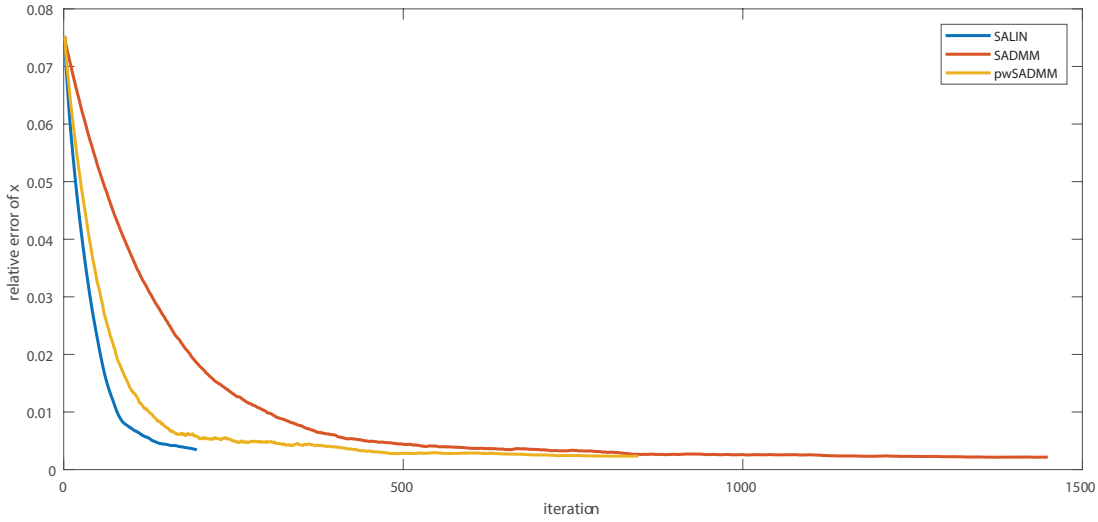

Figure 5.6: Relative distance of x against CPU time (in ms).

Figure 5.6 and 5.5 of $\|x - x^*\|_2/\|x^*\|_2$ show similar patterns with 5.3 and 5.4.

5.2 Signal Smoothing and Detection

In this example, we evaluate SALIN's performance on a signal smoothing and detection task: comparative genomic hybridization (CGH) analysis. CGH analysis is a technique in bio-informatics for measuring number of DNA copies of gene cells along genome. The CGH signals measure the log difference between DNA copy number on tumor cells and that of reference cells. A CGH signal with zero value corresponds to normal copy of the gene cell, while a non-zero CGH signal, also known as hot spot, is likely to be an irregular gene copy of tumor cell.

A fused LASSO model was proposed by (Tibshirani & Wang, 2008) to detect this copy number variation. This is a one-dimensional signal approximation problem with the design matrix X being the identity matrix, which has the following form:

$$\min_{x\in\mathbb{R}^n} \frac{1}{2n}\|x-b\|^2 + \lambda_1\|x\|_1 + \lambda_2 \sum_{j=2}^{n} \|x_j - x_{j-1}\|_1 \tag{5.2}$$

The first squared loss term controls the distance between the observed signals and the approximated ones; the second LASSO penalty term imposes sparsity to the approximator; and the third fused LASSO penalty imposes a piece-wise linear structure. We can combine the last two penalty terms into one generalized LASSO penalty term $\|Gx\|_1$, with G being:

$$G = \begin{bmatrix} 1 & -1 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

The resulting generalized LASSO problem can be easily handled by SALIN and stochastic ADMM. Because the design matrix is identity matrix which is perfectly conditioned, thus pwSADMM is exactly the same as Stochastic ADMM.

The dataset is available in the Matlab build-in Coriell cell dataset. In our experiment, we set $\lambda_1 = 0.01, \lambda_2 = 0.05$ for the model. Both SALIN and SADMM run in mini-batch

mode, with batch-size 32. Below is a plot that shows the solution quality returned by SALIN (because SADMM returns exactly same solution with SALIN, so we just output the solution of SALIN).

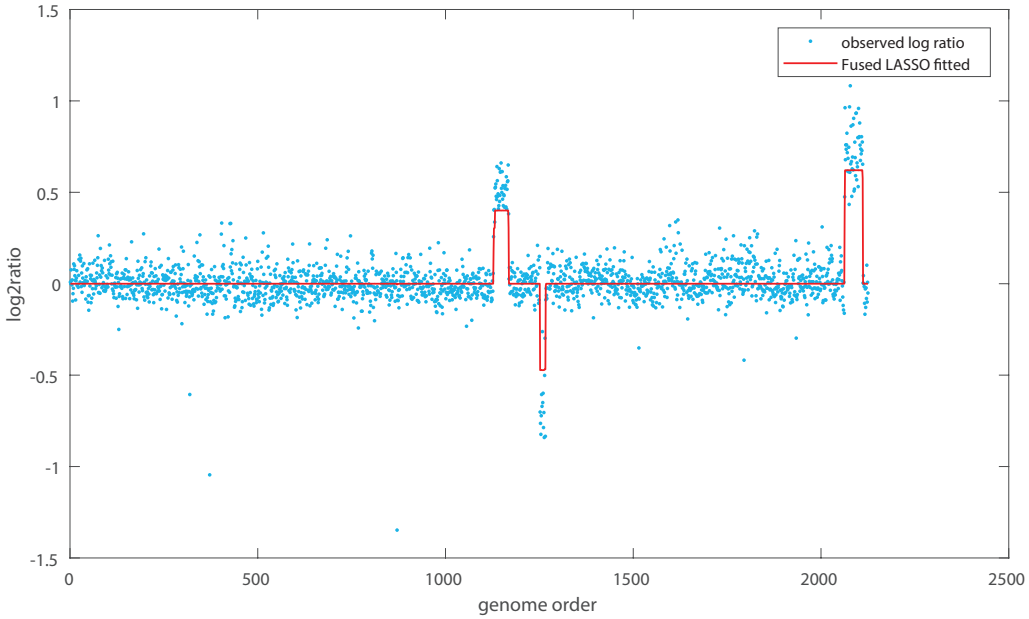

Figure 5.7: Blue dots are observed signals; red line is Fused LASSO approximation by SALIN

Figure 5.7 shows that both SALIN and SADMM return a smoothed piece-wise linear approximation to the original data, which is the desired approximator of this Fused LASSO model. The two methods shrink non-significant variables to zero while successfully detect the those regions with significant non-zero variables, which corresponds suspected abnormal cell in the CGH study.

5.3 Graph-Guided SVM

The Graph-Guided SVM (Ouyang et al., 2013) is a graphical extension to support vector machine (SVM) used in classification. As described in (4.16), the objective is to minimize:

$$\min_x \frac{1}{n}\sum_{i=1}^{n}[1 - b_i W_i^T x]_+ + \frac{\nu}{2}\|x\|_2^2 + \lambda\|y\|_1 \qquad (5.3)$$
$$s.t.\, Gx - y = 0$$

where b_i is the label for instance vector $W_i \in \mathbb{R}^{1\times d}$, and $[z]_+ = \max(0, z)$. The regularization matrix G is constructed based on graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, which represents the connection of variables. For this example, we construct graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ using sparse inverse covariance estimation (Banerjee, Ghaoui, & d'Aspremont, 2008) (also known as graphical lasso) (Friedman & Tibshirani, 2007; Boyd et al., 2010) and determine the sparsity pattern of the inverse covariance matrix Σ^{-1}. We threshold components of Σ_{ij}^{-1} to 0 and 1, and construct the G matrix accordingly, i.e., we append a row to G with i^{th} entry being 1 and j^{th} entry being -1, when $\Sigma_{ij}^{-1} = 1$. In the 20-news example, the relation of the 100 different words can be visualized as follows (Ouyang et al., 2013):

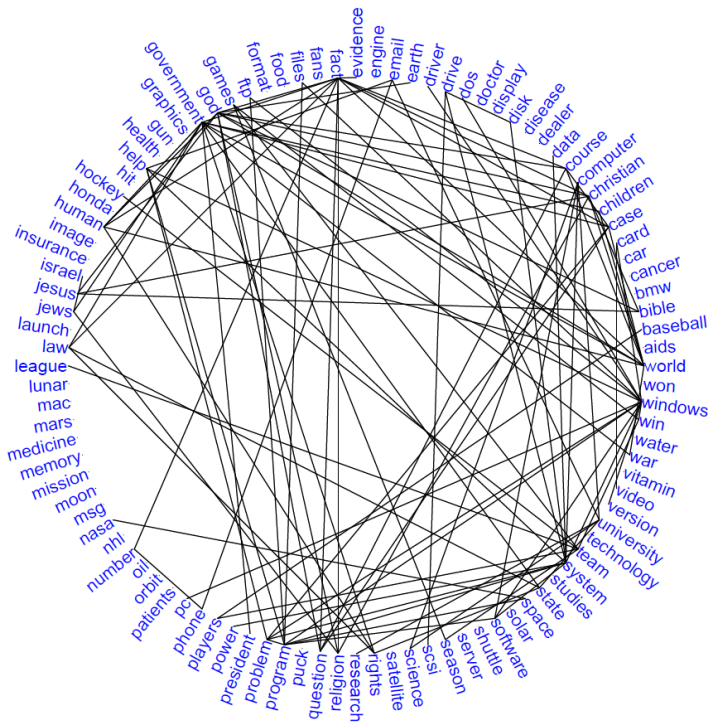

Figure 5.8: Mutual relations of the 100 words, from 20news dataset.

To evaluate the performance, we test SALIN and SADMM on two real-world datasets which are publicly available. The first dataset is '20newsgroup', downloaded from `www.cs.nyu.edu/~roweis/data.html`; the second is 'a9a', downloaded from LIBSVM website. Each original dataset is split into 80% for training and 20% for testing. The summary of datasets is given in Table 5.3:

Dataset	class	Training examples	Testing examples	Features
news20	4	12994	3248	100
a9a	2	39074	9768	123

Table 5.3: Summary of datasets.

For "new20groups" dataset, we use the "one-vs-rest" scheme for multi-class classification. We set the parameters in the GGSVM model to be: $\nu = \lambda = 1/n$. Each method is implemented in mini-batch mode with batch-size 32; SALIN has test sample size $|S| = 128$. The step size in each method is exponential decaying, and be tuned to achieve maximum performance correspondingly.

All the experiments were repeated 100 times, and at each time, each method is run for 2 epochs (when algorithm processes $2n$ data samples). The results are reported by averaging over 100 repetitions. We evaluate the two methods by measuring objective values, out-sample test error rate, and running time. The comparison results are given in Table 5.4.

Task	Methods	Objective value	Test error rate	Time (s)
news20	SALIN	0.280 ± 10^{-7}	0.1510 ± 0.010	0.928
	SADMM	0.278 ± 0.0021	0.1525 ± 0.020	0.281
a9a	SALIN	0.359 ± 10^{-7}	0.1524 ± 0.0012	0.272
	SADMM	0.358 ± 0.0034	0.1533 ± 0.0027	0.169

Table 5.4: Performance comparison of SALIN and SADMM.

The summary shows that both methods are able to acquire a good solution in very short amount of time. SALIN gives slightly lower test error and smaller variance than SADMM. Moreover, SALIN returns almost same objective value over different repetitions,

which means that SALIN is not affected by randomness and very robust. SALIN requires more computing time due to the fact that it has to compute multiple objective function values at each iteration.

Next we show how each method approaches convergence. The demonstrated example is from "a9a" dataset. We record the objective value every 16 iterations (at each iteration the algorithm observes 32 samples). The x-axis is iteration number divided by 16, and the y-axis records the test error on the testing set, averaged over 100 repetitions.

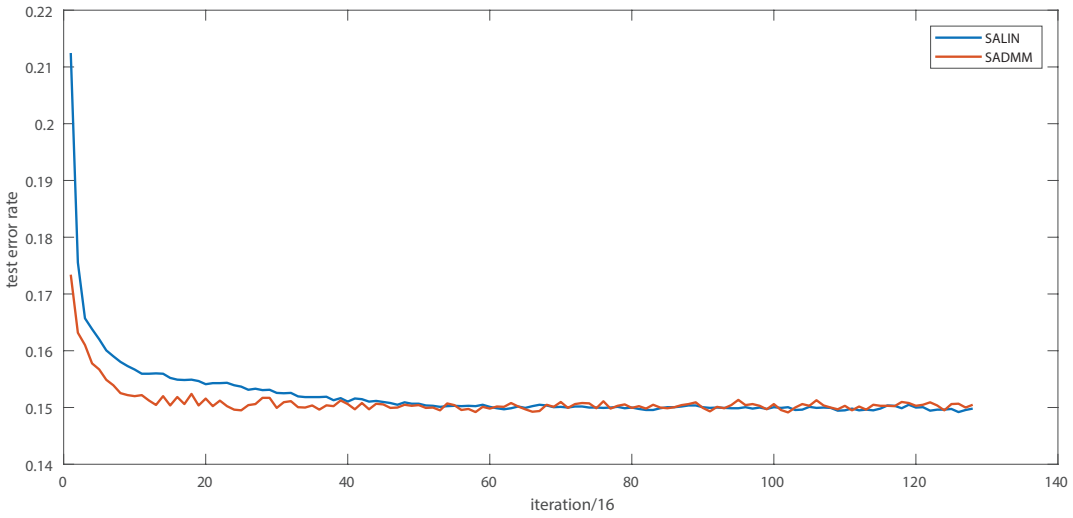

Figure 5.9: Convergence behavior of the two methods. Dataset: a9a.

Figure 5.9 shows that the tail convergence of SALIN is very smooth and almost monotonic, compared to that of SADMM. This favorable property is due to the special update test and the sub-gradient update schedule.

Chapter 6

Conclusion and Future Plans

6.1 Conclusion

In this dissertation, we propose two stochastic alternating optimization methods for solving structured regularization problems, which have been widely used in machine learning and data mining. The first algorithm is called Stochastic Alternating Linearization (SALIN), which is an stochastic extension of the Alternating Linearization (ALIN) in solving convex optimization problems with complex non-smooth regularization term. SALIN linearizes the loss function and penalty function alternatively at each iteration, based on the stochastic approximation of sub-gradients. By applying a special update test at each iteration, and a carefully designed sub-gradients update scheme, the algorithm achieves fast and stable convergence. The update test just relies on a fixed pre-defined sample set, and we show that the choice of the test set has little influence on the overall performance of the algorithm. Therefore SALIN is a robust method.

The other algorithm is called preconditioned stochastic Alternating Direction Method of Multipliers, which is specially designed to handle structured regularized regression problems such as Fused LASSO, but with the design matrix being ill-conditioned. We prove its $O(1/\sqrt{t})$ convergence rate for general convex functions and $O(\log t/t)$ for strongly convex functions, and show that the constant depends only on the lower dimension of the data matrix.

We evaluate our proposed methods in extensive numerical experiments of structured regularization problems. The experiments include Fused LASSO regression, signal smoothing, and graph-guided SVM, with both synthetic and real-world datasets. The numerical results demonstrate the efficacy and accuracy of our methods. We especially demonstrate that SALIN is a robust method that gives consistent results with different choice of test

sample; and it has stable tail convergence.

6.2 Future Plans

Our next step is to develop theoretical results and proofs for SALIN. The numerical experiments have demonstrated that SALIN has competitive and sometimes better convergence behavior than other stochastic methods, e.g., stochastic ADMM. Thus we expect SALIN to have good theoretical guarantee.

It is also interesting to develop an adaptive version of SALIN: in the proximal term $\frac{\|x-\hat{x}\|_D^2}{2\eta}$ of the h- and f-subproblem, the matrix D could be time-varying instead of fixed over iterations. Similar ideas have been applied to stochastic gradient descent and stochastic ADMM already, and demonstrated satisfactory results. We expect this variation will also benefit SALIN in certain problem types.

We will also consider extending SALIN to online learning setting, where new data samples come in a sequential fashion. In the online setting, the design of update test is of critical importance. It should consider both existing and incoming samples, and maintain low computational cost as well.

Moreover, we will apply SALIN to more problems that are currently of interest, such as high-dimensional problems, or problems with highly complicated regularization terms.

References

Banerjee, O., Ghaoui, L., & d'Aspremont. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, *9*, 485-516.

Bauschke, H., & Combettes, P. (2011). *Convex analysis and monotone operator theory in hilbert spaces*. New York: Springer.

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

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*.

Clarkson, K. L. (2005). Subgradient and sampling algorithms for ℓ_1 regression. In *Proceedings of the sixteenth annual acm-siam symposium on discrete algorithms* (p. 257-266).

Clarkson, K. L., & Woodruff, D. P. (2013). Low rank approximation and regression in input sparsity time. In *Symposium on theory of computing (stoc)* (p. 81-90).

Combettes, P. L. (2009). Iterative construction of the resolvent of a sum of maximal monotone operators. *J. Convex Anal.*, *16*(3-4), 727-748.

Dasgupta, A., Drineas, P., Harb, B., Kumar, R., & Mahoney, M. W. (2009). Sampling algorithms and coresets for ℓ_p regression. *SIAM J. on Computing*, *38*, 2060-2078.

Deng, W., & Yin, W. (2016). On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, *66*, 889-916.

Deshpande, A., & Rademacher, L. (2010, Oct). Efficient volume sampling for row/column subset selection. In *2010 ieee 51st annual symposium on foundations of computer science* (p. 329-338).

Douglas, J., & Rachford, H. H. (1956). On the numerical solution of heat conduction problems in two and three space variables. *Trans. Amer. Math. Soc.*, *82*(2), 421-439.

Drineas, P., Mahoney, M. W., Muthukrishnan, S., & Sarlos, T. (2011). Faster least squares approximation. *Numer. Math.*, *117*, 219-249.

Du, Y., Lin, X., & Ruszczyński, A. (2017). Selective linearization method for multiblock convex optimization. *SIAM Journal on Optimization*, *27*(2), 1102-1117.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121-2159.

Eckstein, J., & Bertsekas, D. P. (1992). On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, *55*(3), 293-318.

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

Friedman, J., & Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 432-441.

Gabay, D., & Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, *2*(1), 17-40.

Glowinski, R., & Tallec, P. L. (1989). *Augmented lagrangian and operator-splitting methods in nonlinear mechanics.* SIAM.

Goldfarb, D., Ma, S., & Scheinberg, K. (2013). Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming*, *141*, 349-382.

Goldstein, T., & Osher, S. (2009). The split bregman method for ℓ_1-regularized problems. *SIAM Journal on Imaging Sciences*, *2*(2), 323-343.

Golub, G. H., & Loan, C. F. V. (1996). *Matrix computations, johns hopkins studies in the mathematical sciences.* Baltimore, MD: Johns Hopkins University Press.

He, B., & Yuan, X. (2012). On the $o(1/n)$ convergence rate of the douglas-rachford alternating direction method. *SIAM Journal on Numerical Analysis*, *50*, 700-709.

Hong, M., & Luo, Z. (2017). On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, *162*(1-2), 165-199.

Kim, S., Sohn, K., & Xing, E. (2009). A multivariate regression approach to association

analysis of a quantitative trait network. *Bioinformatics*, *25*, 204-212.

Kingma, D., & Ba, J. L. (2014). Adam: a method for stochastic optimization. *International Conference on Learning Representations*, 1-13.

Kiwiel, K. (1985). *Methods of descent for non-differentiable optimization*. Springer.

Kiwiel, K., Rosa, C., & Ruszczyński, A. (1999). Proximal decomposition via alternating linearization. *SIAM Journal on Optimization*, *9*(3), 668-689.

Lin, X., Minh, P., & Ruszczyński, A. (2014). Alternating linearization for structured regularization problems. *Journal of Machine Learning Research*, *15*, 3447-3481.

Lions, P. L., & Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer. Anal.*, *16*(6), 964-979.

Meng, X., & Mahoney, M. (2013). Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. *Symposium on the Theory of Computing (STOC)*, 91-100.

Ouyang, H., He, N., Tran, L., & Gray, A. (2013). Stochastic alternating direction method of multipliers. *International Conference on Machine Learning*, 80-88.

Peaceman, D. W., & Rachford, H. H. (1955). The numerical solution of parabolic and elliptic differential equations. *J. Soc. Indust. Appl. Math.*, *3*, 28-41.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *The Official Journal of the International Neural Network Society*, *12*, 145-151.

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

Ruszczyński, A. (2006). *Nonlinear optimization*. Princeton University Press.

Suzuki, T. (2013). Dual averaging and proximal gradient descent for online alternating direction multiplier method. *International Conference on Machine Learning*, 392-400.

Tibshirani, R., & Taylor, J. (2011). The solution path of the generalized lasso. *Annual of Statistics*, *39*, 1335-1371.

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

Tseng, P. (2001). Convergence of block coordinate descent method for nondifferentiable

minimization. *Journal of Optimization Theory and Applications*, *109*, 474-494.

Wang, H., & Banerjee, A. (2012). Online alternating direction method. *International Conference on Machine Learning*, 1119-1126.

Woodruff, D., & Zhang, Q. (2013). Subspace embeddings and ℓ_p-regression using exponential random variables. *JMLR: Workshop and Conference Proceedings*, *30*, 1-22.

Xu, F., Huang, H., & Wen, Z. (2015). High dimensional covariance matrix estimation using multi-factor models from incomplete information. *Science China Mathematics*, *4*, 829-844.

Xu, Y., Yin, W., Wen, Z., & Zhang, Y. (2011). An alternating direction algorithm for matrix completion with nonnegative factors. *Frontiers of Mathematics in China*, *7*, 365-384.

Yang, J., Chow, Y., Ré, C., & Mahoney, M. (2016). Weighted sgd for ℓ_p regression with randomized preconditioning. In *Proceedings of the twenty-seventh annual acm-siam symposium on discrete algorithms* (p. 558-569).

Yang, J., Meng, X., & Mahoney, M. W. (2014). Quantile regression for large-scale applications. *SIAM J. Scientific Computing*, *36*(5), S78-S110.

Yang, J., Meng, X., & Mahoney, M. W. (2016). Implementing randomized matrix algorithms in parallel and distributed environments. In *Proceedings of the ieee* (Vol. 104, p. 58-92).

Yang, J., & Zhang, Y. (2011). Alternating direction algorithms for ℓ_1-problems in compressive sensing. *SIAM J. on Scientific Computing*, *33*, 250-278.