OPTIMAL STOCHASTIC AND DISTRIBUTED ALGORITHMS FOR MACHINE LEARNING

A Thesis
Presented to
The Academic Faculty

by

Hua Ouyang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
August 2013

OPTIMAL STOCHASTIC AND DISTRIBUTED ALGORITHMS FOR MACHINE LEARNING

Approved by:

Dr. Hongyuan Zha, Committee Chair
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Alexander G. Gray, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Maria Balcan
School of Computer Science
Georgia Institute of Technology

Dr. Arkadi Nemirovski
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Haesun Park
School of Computational Science and Engineering
Georgia Institute of Technology

Date Approved: June 27, 2013

To my wife Wan Wang.

I love you dearly.

ACKNOWLEDGEMENTS

A lot of people have helped me through this interesting and fruitful journey of Ph.D. study. Without their help I would never have been able to finish it. I would like to express my deepest gratitude to them.

First, I would like to express my deepest appreciation to my advisor Dr. Alexander Gray. He has provided me innumerous support, help and encouragement during the past few years, and has had a profound influence on my research from various perspectives. He introduced me to the promising area of large-scale machine learning. I have learnt a lot of research skills from Alex including scalable algorithm design, basic analysis methods, scientific writing, and a good taste to find important problems. Dr. Hongyuan Zha served as my co-advisor and thesis committee member. As an excellent teacher I have learnt from him numerical linear algebra, iterative methods, data mining and skills like critical thinking and problem solving. He has also given me invaluable support on my career development.

I would also like to thank the other committee members. Dr. Arkadi Nemirovski helped me build a solid background in optimization, Dr. Maria Balcan gave me a lot of knowledge and advices in theoretical machine learning and game theory, and Dr. Haesun Park exposed me to numerical methods and matrix factorization problems. Special thanks would go to Dr. Alexander Shapiro. He is the best teacher in my life and helped me build a solid foundation in statistics. Dr. Le Song also gave me lots of advices and ideas for my research.

During the summer of 2010 I interned at IBM T. J. Watson Research. I would like to thank Lexing Xie and Paul Natsev in the multimedia research group. They gave me a lot of advices in industrial research, and provided me strong support on my career development.

Then I would like to thank my fellow labmates in FASTLab, friends and colleagues at Georgia Tech that I feel so fortunate to meet, to learn from, and to work with in the past few years. To name a few: Ryan Curtin, Nan Du, Ravi Ganti, Wei Guan, Shuang Hao, Niao He, Krishna Kumar, Dongryeol Lee, Fuxin Li, Liangda Li, Yi Mao, William March, Nishant

Mehta, Parikshit Ram, Ryan Riegel, Mingxuan Sun, Long Q. Tran, Nikolaos Vasiloglou, Shuang-Hong Yang and Ke Zhou. I wish them all the best in the future.

Last but not least, I would like to thank my family for their innumerous support. I am deeply indebted to my wife Wan Wang for her endless love, patience, understanding, and encouragement. She has brought me such a wonderful life in the past five years that make everything possible. My parents and parents-in-law have always been supportive on our decisions, and helped us live a better life in the county that is tens of thousands of miles away from them.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

Stochastic and data-distributed optimization algorithms have received lots of attention from the machine learning community due to the tremendous demand from the large-scale learning and the big-data related optimization. A lot of stochastic and deterministic learning algorithms are proposed recently under various application scenarios. Nevertheless, many of these algorithms are based on heuristics and their optimality in terms of the generalization error is not sufficiently justified. In this talk, I will explain the concept of an optimal learning algorithm, and show that given a time budget and proper hypothesis space, only those achieving the lower bounds of the estimation error and the optimization error are optimal.

Guided by this concept, we investigated the stochastic minimization of nonsmooth convex loss functions, a central problem in machine learning. We proposed a novel algorithm named Accelerated Nonsmooth Stochastic Gradient Descent, which exploits the structure of common nonsmooth loss functions to achieve optimal convergence rates for a class of problems including SVMs. It is the first stochastic algorithm that can achieve the optimal $O(1/t)$ rate for minimizing nonsmooth loss functions. The fast rates are confirmed by empirical comparisons with state-of-the-art algorithms including the averaged SGD.

The Alternating Direction Method of Multipliers (ADMM) is another flexible method to explore function structures. In the second part we proposed stochastic ADMM that can be applied to a general class of convex and nonsmooth functions, beyond the smooth and separable least squares loss used in lasso. We also demonstrate the rates of convergence for our algorithm under various structural assumptions of the stochastic function: $O(1/\sqrt{t})$ for convex functions and $O(\log t/t)$ for strongly convex functions. A novel application named Graph-Guided SVM is proposed to demonstrate the usefulness of our algorithm.

We also extend the scalability of stochastic algorithms to nonlinear kernel machines, where the problem is formulated as a constrained dual quadratic optimization. The simplex

constraint can be handled by the classic Frank-Wolfe method. The proposed stochastic Frank-Wolfe methods achieve comparable or even better accuracies than state-of-the-art batch and online kernel SVM solvers, and are significantly faster.

The last part investigates the problem of data-distributed learning. We formulate it as a consensus-constrained optimization problem and solve it with ADMM. It turns out that the underlying communication topology is a key factor in achieving a balance between a fast learning rate and computation resource consumption. We analyze the linear convergence behavior of consensus ADMM so as to characterize the interplay between the communication topology and the penalty parameters used in ADMM. We observe that given optimal parameters, the complete bipartite and the master-slave graphs exhibit the fastest convergence, followed by bi-regular graphs.

CHAPTER I

INTRODUCTION

The interplay between the high-speed Internet, cheaper data storage and the popularity of mobile and ubiquitous computation are converging to the recent trend of Big Data. Enormous datasets are becoming available from many areas, such as web search, social media, security, biology, health and physics. One of the most important venue of data analysis is the area of **Machine Learning**, including the important subareas of regression, classification, density estimation, data modeling, statistical inference, clustering and reduction. Despite the rapid advances in various aspects of machine learning, the scalability issue has not received as much consideration as it deserves. This makes many statistically sound methods less applicable to the real-world large-scale datasets, or not optimally adapted to the modern distributed infrastructures of computation. In the era of Big Data, *massive data should be viewed as assets other than liabilities.* Therefore machine learning researchers must be prepared for the revolution where efficiently handling large-scale data sources becomes crucial. *This dissertation aims to reveal the fundamental theories, to design, analyze and apply new algorithms towards bridging the gap between the traditional setting of machine learning and its capability in the new Big Data environment.*

In this dissertation, we introduce a detailed decomposition of the generalization error, one of the most important quantity to measure the learning capacity of a supervised learning algorithm. We present a deeper and detailed understanding on the error sources and identify the possibilities to achieve the lower bounds of each source. The computational cost is also taken into account to make sure that an algorithm achieves the lowest possible generalization error within a time budget, which is essential in large-scale machine learning applications. Several stochastic and distributed algorithms are then proposed to explore the specific structures in the learning models and data samples.

This introduction provides an overview of the motivations, the proposed stochastic and

distributed optimization algorithms for various learning models. The contributions and outline of this dissertation will also be summarized.

1.1 Scalable Machine Learning and Optimization

Modern machine learning is very closely related to optimization techniques (Sra et al. [2011]). The learning process of many important statistical formulations can be cast as solving optimization problems. Actually any learning method that involves finding the optimal set of "parameters" of a hypothetical parametric model can be solved by optimization techniques. Some generic model-based examples are maximum likelihood estimation (MLE), maximum a posteriori estimation (MAP) and variational inference for Bayesian methods. The non-model-based examples include support vector machines, neural networks, dictionary learning and matrix factorization methods. Many semi-supervised and unsupervised learning methods are also solved by optimization methods. Some machine learning models that were originally proposed with little relevance to optimization methods were eventually shown to be optimized in certain ways. One example is the AdaBoost algorithm, where the coordinate descent method is applied on the exponential loss function (Mukherjee et al. [2011]).

The scalability of machine learning models heavily relies on the selection of novel and efficient optimization algorithms. Different models might have different functional structures or data properties. Solving a machine learning problem with generic optimization techniques without exploring problem structures could lead to suboptimal or inferior performances. Rich structures in machine learning problems provide enormous opportunities for tailor-made optimization algorithms to succeed beyond the worst case guarantees provided by generic algorithms. A simple example is illustrated in Fig.1, where the following Lasso problem (Tibshirani [1996])

$$\min_{\mathbf{x}\in\mathbb{R}^D} \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^T\mathbf{s}_i - \ell_i)^2 + \lambda\|\mathbf{x}\|_1 \tag{1}$$

is solved by two different methods, and the relative errors of this optimization process are plotted agains the number of iterations. The first method is the generic subgradient

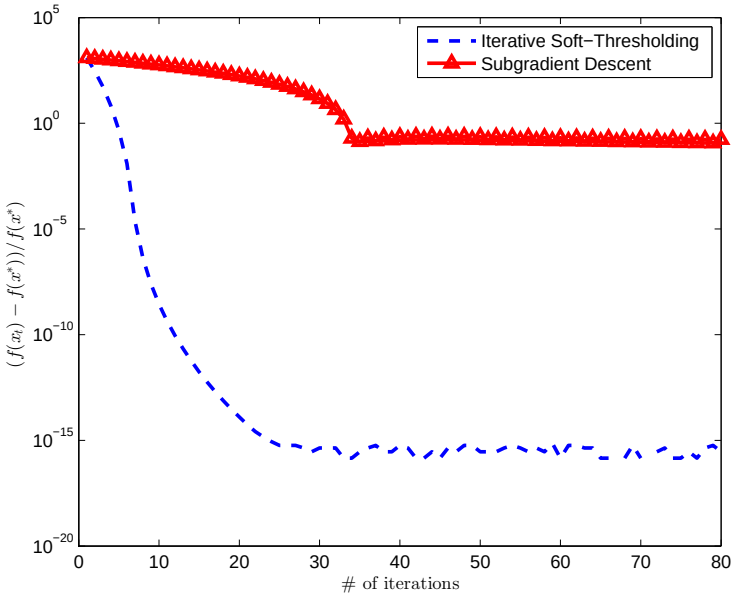

Figure 1: Relative errors v.s. number of iterations. Solve Lasso (1) by gradient descent and iterative soft-thresholding.

descent, where model (1) is treated as a general unconstrained nonsmooth convex optimization problem, and the algorithm only exhibits a sublinear convergence rate. The second method is the iterative soft-thresholding algorithm (also known as the forward-backward splitting) proposed by Lions and Mercier [1979] and analyzed by Chen and Rockafellar [1997], Daubechies et al. [2004]. This algorithm takes advantage of the separability between the least square function and the simple ℓ_1 regularization, and hence a fast linear convergence rate is attained.

On the other hand, suitable optimization techniques and frameworks can also lead to new machine learning models, or make existing models more scalable to large-scale applications. A prominent example is the alternating direction method of multipliers (ADMM) originally proposed in 1970s' by Glowinski and Marroco [1975], Gabay and Mercier [1976]. This algorithm has not attracted many interests from the machine learning community until the monograph written by Boyd et al. [2010]. Boyd et. al. showed that large scale linear models can be efficiently solved in a distributed computing environment via ADMM.

1.1.1 Some Notations

Before delving into the detailed discussions, we first introduce some notations and assumptions used in the following sections. Let $f(\mathbf{x}, \xi)$ be the cost function of the learning model, where $\mathbf{x} \in \mathbb{R}^D$ is the optimization variable, and we use the random variable ξ to denote the examples. We use subscripts $\mathbf{x}_t$ to denote the solution of an iterative optimization algorithm provided at iteration t. The hypothesis space of our model is denoted by $\mathcal{H}$, while the hypothesis space that the "true" solution belongs to is denoted by $\mathcal{H}^*$. A learning algorithm can access the finite N training examples $\xi_1, \ldots, \xi_N$. The averaged cost function over these N samples is denoted as $\hat{f}(\mathbf{x}) \equiv \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}, \xi_i)$. The solution $\mathbf{x} \in \mathcal{H}$ that minimize $\hat{f}(\mathbf{x})$ is denoted as $\mathbf{x}^*_{(N)} \equiv \arg\min_{\mathbf{x} \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}, \xi_i)$

The following i.i.d. assumption for the examples ξ is made for all the stochastic algorithms:

Assumption 1. *Examples ξ are drawn identically and independently (i.i.d.) from a fixed but unknown distribution $\mathcal{P}$.*

1.2 Motivations

In this section we present the main motivations of this thesis work. As we discussed in Section 1.1, it is very crucial that an optimization process takes advantages of the structural properties of a machine learning problem. Hence the question of our main concern is: what are the best algorithms for *large-scale* machine learning tasks?

This question is definitely hard, and the answers are not simple either. To investigate this question, we start with the decomposition of the generalization error: the most natural measurement that should be minimized to get the best performance for a supervised machine learning algorithm. To investigate the possibilities in reducing this quantity, a deeper understanding in the sources of the generalization error is desired.

We will show in this section that a detailed decomposition reveals the various possibilities that can be taken advantages of, and as a consequence it leads to the concept of *optimal learning algorithms*. This idea of decomposing the generalization error is inspired by the work of Bottou and Bousquet [2008].

1.2.1 Detailed Decomposition of Generalization Error

The quality of a supervised learning method (or learner) is typically evaluated by the generalization ability, which can be measured by the *generalization error* $\mathcal{E}_{\text{Gen}}$: the mistakes that a learner makes on the *unseen* testing samples with its current solution $\mathbf{x}_t$:

$$\mathcal{E}_{\text{Gen}} \equiv \mathbb{E}_{\xi\sim\mathcal{P}} f(\mathbf{x}_t, \xi) = \int_{\xi\sim\mathcal{H}} f(\mathbf{x}_t, \xi) d\mathcal{P}. \tag{2}$$

A very closely related quantity known as the *excess error* $\mathcal{E}_{\text{Exc}}$ (Boucheron et al. [2005]) is defined as the deviation between the generalization error and the best possible accuracy a learner could have achieved given the ideal $\mathcal{H}^*$ and infinite number of training samples. Given a distribution $\mathcal{P}$, this error can be expressed as the following expectation:

$$\mathcal{E}_{\text{Exc}} \equiv \mathbb{E}_{\xi\sim\mathcal{P}}\big[f(\mathbf{x}_t, \xi) - \inf_{\mathbf{x}\in\mathcal{H}^*} f(\mathbf{x}, \xi)\big]. \tag{3}$$

Given the "true" hypothesis space $\mathcal{H}^*$, the second term $\inf_{\mathbf{x}\in\mathcal{H}^*} f(\mathbf{x}, \xi)$ is a constant, hence reducing $\mathcal{E}_{\text{Exc}}$ is equivalent to reducing $\mathcal{E}_{\text{Gen}}$.

Traditionally the machine learning community has been focused on the trade-off between the estimation error and approximation error (Bartlett [2008]). This is also known as the bias-variance trade-off in the statistics community. A large part of the statistical learning theory literature is devoted to the problems of how to make proper model selections in terms of the complexity of the hypothesis space: $|\mathcal{H}|$, such that the sum of these two errors is small, or given some fixed error tolerance, the sample complexity requirement is low (Bishop [1996, 2007], Boucheron et al. [2005]). This decomposition leads to the concept of structural risk minimization (Vapnik [1998, 2000]).

The traditional error decomposition implicitly assumes that an algorithm can always find the *exact* optima of the empirical risk, and the the finite algorithmic accuracy of an iterative algorithm is not considered in any form. However, this assumption is not realistic, especially when the number of data samples N is large. In large scale learning problems, it is hard *and not necessary* to attain a very high algorithmic accuracy. To fill this gap, Bottou and Bousquet [2008] proposed an alternative decomposition of the excess error. It can be considered as an extension of the traditional setting. As illustrated in Fig.2, the

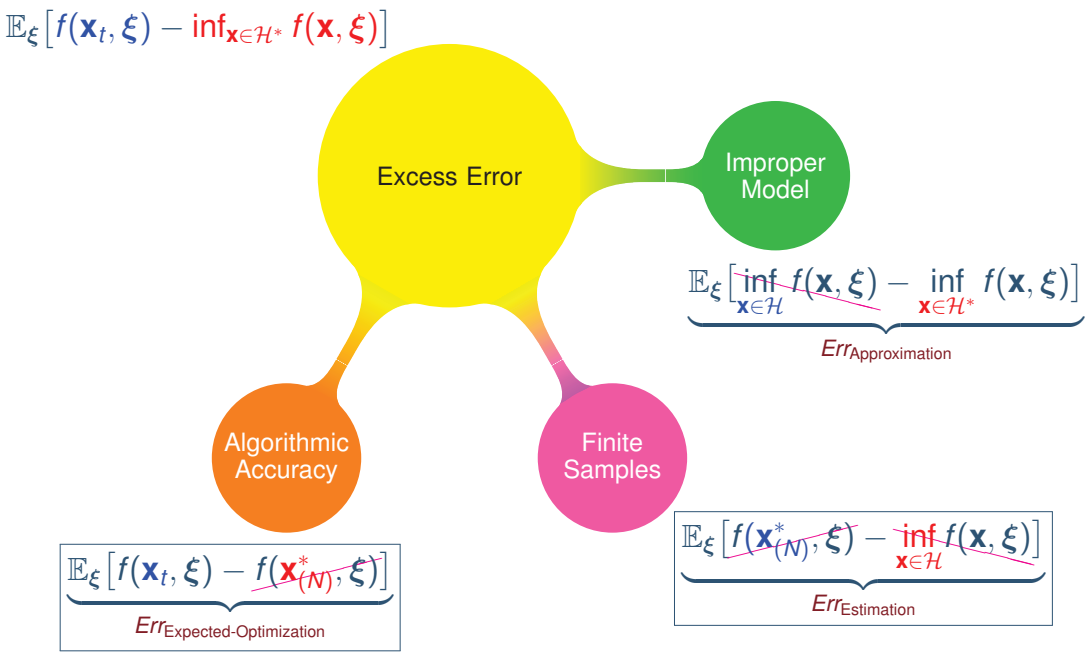

Figure 2: Three sources of generalization error. The boxed components (estimation error and expected optimization error) are of our main focus.

excess error $\mathcal{E}_{\text{Exc}}$ can be decomposed into three sources: the *approximation error* $\mathcal{E}_{\text{App}}$, the *estimation error* $\mathcal{E}_{\text{Est}}$, and the *expected optimization error* $\mathcal{E}_{\text{Exp-Opt}}$:

$$\begin{aligned}\mathcal{E}_{\text{Exc}} &= \mathbb{E}_{\xi\sim\mathcal{P}}\left[f(\mathbf{x}_t,\xi) - \inf_{\mathbf{x}\in\mathcal{H}^*} f(\mathbf{x},\xi)\right] \\ &= \underbrace{\mathbb{E}_\xi\left[\inf_{\mathbf{x}\in\mathcal{H}} f(\mathbf{x},\xi) - \inf_{\mathbf{x}\in\mathcal{H}^*} f(\mathbf{x},\xi)\right]}_{\mathcal{E}_{\text{App}}} + \underbrace{\mathbb{E}_\xi\left[f(\mathbf{x}^*_{(N)},\xi) - \inf_{\mathbf{x}\in\mathcal{H}} f(\mathbf{x},\xi)\right]}_{\mathcal{E}_{\text{Est}}} + \underbrace{\mathbb{E}_\xi\left[f(\mathbf{x}_t,\xi) - f(\mathbf{x}^*_{(N)},\xi)\right]}_{\mathcal{E}_{\text{Exp-Opt}}},\end{aligned} \tag{4}$$

where we use colors to highlight the differences between terms.

In this decomposition, $\mathcal{E}_{\text{App}}$ measures the difference between the best $\mathbf{x}^*$ obtained from $\mathcal{H}$ and that from the true hypothesis space $\mathcal{H}^*$, meaning that it is due to the improper choice of the learning model; $\mathcal{E}_{\text{Est}}$ reflects the effect of finite samples: the more samples we have, the smaller estimation error is, when $N \to \infty$, $\mathcal{E}_{\text{Est}} \to 0$; the last term $\mathcal{E}_{\text{Exp-Opt}}$ measures the finite algorithmic precision of an iterative optimization algorithm. Various trade-offs between these three components are needed to achieve a good balance such that a learner can retain a low excess error (Bottou and Bousquet [2008]).

1.2.2 Lower Bounds for $\mathcal{E}_{\text{Est}} + \mathcal{E}_{\text{Exp-Opt}}$

We assume that a proper hypothesis class $\mathcal{H}$ is already chosen and presented to the learning algorithm, hence $\mathcal{E}_{\text{App}}$ in the decomposition (4) is fixed. We will focus on the problem of how to minimize the estimation error and optimization error, *especially when the number of training samples N is large*. The purpose is to demonstrate that for large scale learning problems with massive data samples, the large N should be viewed as assets other than liabilities.

To achieve this goal, we derive in the following a tight worst-case lower bound for $\mathcal{E}_{\text{Est}} + \mathcal{E}_{\text{Exp-Opt}}$. As we will see in the following chapters, this lower bound is met by the proposed stochastic algorithms.

The sum of the two errors of our main concern can be further decomposed and bounded as follows.

$$\begin{aligned}
&\mathcal{E}_{\text{Est}} + \mathcal{E}_{\text{Exp-Opt}} \\
=&\mathbb{E}_\xi\big[f(\mathbf{x}^*_{(N)},\xi) - \inf_{\mathbf{x}\in\mathcal{H}} f(\mathbf{x},\xi)\big] + \mathbb{E}_\xi\big[f(\mathbf{x}_t,\xi) - f(\mathbf{x}^*_{(N)},\xi)\big] \\
=&\mathbb{E}_\xi f(\mathbf{x}_t,\xi) - \mathbb{E}_\xi f(\mathbf{x}^*_{\mathcal{H}},\xi) \\
=&\mathbb{E}\big[\mathbb{E}_\xi f(\mathbf{x}_t,\xi) - \widehat{f}(\mathbf{x}_t)\big] + \mathbb{E}\big[\widehat{f}(\mathbf{x}^*_{(N)}) - \mathbb{E}_\xi f(\mathbf{x}^*_{\mathcal{H}},\xi)\big] + \mathbb{E}\big[\widehat{f}(\mathbf{x}_t) - \widehat{f}(\mathbf{x}^*_{(N)})\big] \\
\geq&\mathbb{E}\big[\mathbb{E}_\xi f(\mathbf{x}_t,\xi) - \widehat{f}(\mathbf{x}_t)\big] + \mathbb{E}\big[\widehat{f}(\mathbf{x}^*_{(N)}) - \mathbb{E}_\xi f(\mathbf{x}^*_{(N)},\xi)\big] + \mathbb{E}\big[\widehat{f}(\mathbf{x}_t) - \widehat{f}(\mathbf{x}^*_{(N)})\big],
\end{aligned} \tag{5}$$

where in the last inequality we used the fact that $\mathbf{x}^*_{\mathcal{H}}$ is always the minimizer of $\mathbb{E}_\xi f(\mathbf{x},\xi)$ in $\mathcal{H}$, hence $\mathbb{E}_\xi f(\mathbf{x}^*_{\mathcal{H}},\xi) \leq \mathbb{E}_\xi f(\mathbf{x}^*_{(N)},\xi)$. We can observe that the first two items in the RHS of (5) have similar forms: the difference between the expectation and the empirical mean. To find a worse-case lower bound for these terms, we use the classic results of concentration inequalities from the literature of empirical process (e.g. van der Vaart and Wellner [1996]). In particular, we include the following result for the completeness of this dissertation. It essentially follows Theorem 2.3 of (Bartlett and Mendelson [2006]).

Theorem 1. *(Bartlett and Mendelson [2006]) For any interger $N \geq \frac{1}{\sigma^2_{\mathcal{H}}}$ there exists constant $c > 0$ for which the following holds:*

$$\sup_{\mathbf{x}\in\mathcal{H}} \mathbb{E}\big|\mathbb{E}_\xi f(\mathbf{x},\xi) - \widehat{f}(\mathbf{x})\big| \geq \frac{c\sigma_{\mathcal{H}}}{\sqrt{N}}, \tag{6}$$

where $\sigma_{\mathcal{H}}^2 \equiv \sup_{\mathbf{x}\in\mathcal{H}} Var_{\xi\sim\mathcal{P}}[f(\mathbf{x},\xi)]$ *is the variance of* f*, and* $\widehat{f}(\mathbf{x}) \equiv \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x},\xi_i)$.

This theorem states that the maximum deviation between the expectation $\mathbb{E}_\xi f(\mathbf{x},\xi)$ and the empirial term $\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x},\xi_i)$ cannot approach zero at a rate faster than $\frac{1}{\sqrt{N}}$ if no additional structural assumption is made. Hence the sum of the first two terms on the RHS of (5) is also lower bounded by $\frac{c\sigma_{\mathcal{H}}}{\sqrt{N}}$.

The third term on the RHS of (5) is the expected optimization error. If we are given a fixed dataset $\xi_1,\ldots,\xi_N$, then $\widehat{f}(\mathbf{x}_t) - \widehat{f}(\mathbf{x}^*_{(N)})$ is just the deterministic optimization error made by the current solution $\mathbf{x}_t$. Depends on the class that the optimization problem belongs to, there are also lower bounds for this term. We give some of these bounds appeared in the optimization literature. Most of these work is pioneered by Arkadi Nemirovski (Nemirovski and Yudin [1983]) and Yurii Nesterov (Nesterov [2004]).

The first lower bound of optimization error applies to the most general case where the least amount of assumptions are made: f is a convex function but not necessarily differentiable, and an iterative optimization algorithm has only access to the first-order oracles $f'(\mathbf{x}_k)$ that is an arbitrary subgradient of f. In this case,

Theorem 2. *For any constants* $R > 0$ *and* $M > 0$*, there exists a convex function* $f : \mathbb{R}^D \to \mathbb{R}$ *that is* M*-Lipschitz continuous on the ball* $B(\mathbf{x}_0, R)$ *such that* $\forall t > 0, \forall \mathbf{x}^* \in B(\mathbf{x}_0, R)$*,*

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \geq \frac{MR}{2(1+\sqrt{t+1})}, \tag{7}$$

for any optimization scheme that generate a sequence $\{\mathbf{x}_k\}_{k=1:t}$ *satisfying*

$$\mathbf{x}_k \in \mathbf{x}_0 + \mathrm{Lin}\{f'(\mathbf{x}_0),\ldots,f'(\mathbf{x}_{k-1})\},\ \forall 0 < k \leq t.$$

This lower bound is tight, since the subgradient descent method can retain a rate of convergence upper bounded by this rate, up to a constant factor.

Nonsmooth convex function is the most general class of our interest, but it suffers from the slow $1/\sqrt{t}$ rate. However, when a convex function is Lipschitz smooth, the lower bound of the convergence rate can be significantly improved to $1/t^2$, as stated by the following theorem. When f is differentiable, the first-order oracle becomes the gradient of the function. We denote it as $\nabla f(\mathbf{x}_k)$ to distinguish it from the subgradient $f'(\mathbf{x}_k)$.

Theorem 3. *For any $\mathbf{x}_0 \in \mathbb{R}^D$ there exits an L-Lipschitz smooth convex function $f : \mathbb{R}^D \to \mathbb{R}$ such that $\forall t > 0$, $\mathbf{x}^* \equiv \arg\min_{\mathbf{x}} f(\mathbf{x})$,*

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \geq \frac{3L\|\mathbf{x}^* - \mathbf{x}_0\|^2}{32(k+1)^2} \tag{8}$$

for any optimization scheme that generate a sequence $\{\mathbf{x}_k\}_{k=1:t}$ satisfying

$$\mathbf{x}_k \in \mathbf{x}_0 + \mathrm{Lin}\{\nabla f(\mathbf{x}_0), \ldots, \nabla f(\mathbf{x}_{k-1})\},\ \forall 0 < k \leq t.$$

The classic gradient descent can only retain an $O(1/t)$ rate. However, the above $O(1/t^2)$ can be achieved by Nesterov's optimal method (Nesterov [1983]). This method is also known as the accelerated gradient descent.

When a function is both L-Lipschitz smooth and μ-strongly convex, a first-order method can enjoy a fast linear convergence rate.

Theorem 4. *For any $\mathbf{x}_0 \in \mathbb{R}^D$ there exits an L-Lipschitz smooth and μ-strongly convex function $f : \mathbb{R}^D \to \mathbb{R}$ such that $\forall t > 0$, $\mathbf{x}^* \equiv \arg\min_{\mathbf{x}} f(\mathbf{x})$,*

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \geq \left(1 - \frac{2}{1 + \sqrt{L/\mu}}\right)^{2t} \|\mathbf{x}^* - \mathbf{x}_0\|^2 \tag{9}$$

for any optimization scheme that generate a sequence $\{\mathbf{x}_k\}_{k=1:t}$ satisfying

$$\mathbf{x}_k \in \mathbf{x}_0 + \mathrm{Lin}\{\nabla f(\mathbf{x}_0), \ldots, \nabla f(\mathbf{x}_{k-1})\},\ \forall 0 < k \leq t.$$

Cobmining the lower bounds established for the estimation error $\mathcal{E}_{\mathrm{Est}}$ (Theorem 1) and the optimization error $\mathcal{E}_{\mathrm{Opt}}$ (Theorem 2, 3 and 4), we are ready to present the following concept of *optimal learning algorithms*.

Definition 1. *Given a fixed distribution $\mathcal{P}$ and an objective function f, any supervised learning algorithm that achieves the lower bounds for the estimation error and the optimization error is statistically optimal.*

The above definition only considers the iteration complexity of a learning algorithm. However, in real-world applications, the time complexity is typically of our major concern. This is especially important for large-scale problems, where the system cannot afford scanning the dataset for many iterations. In extreme cases, even a single sweep is too expensive.

Based on these considerations, we define the computationally optimal learning algorithm as follows.

Definition 2. *Given a fixed distribution $\mathcal{P}$, an objective function f and a time budget $T = N$, any supervised learning algorithm that achieves the lowest $\mathcal{E}_{Est} + \mathcal{E}_{Exp\text{-}Opt}$ within time T is computationally optimal.*

1.2.3 Batch Learning v.s. Stochastic Learning

It has been shown in e.g. Boucheron et al. [2005], Bartlett and Mendelson [2006] and Bartlett [2008] that the $1/\sqrt{N}$ lower bound of the estimation error (6) can be achieved by the principle of the empirical risk minimization (ERM) (Vapnik [1998]). In machine learning ERM is also known as the batch learning method, expressed by the following deterministic optimization:

$$\min_{\mathbf{x}\in\mathcal{X}} \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}, \xi_i). \tag{10}$$

This scheme is essentially the same as the sample average approximation (SAA), one of the two major methods for stochastic programming (Shapiro et al. [2009]). According to Definition 1, if the ERM objective is solve by a deterministic optimization algorithm that achieves the lower bound for optimization error $\mathcal{E}_{\text{Opt}}$, then it is statistically optimal. However, a statistically optimal algorithm is not necessarily computationally optimal. For batch learning algorithms (e.g. gradient descent or quasi-Newton methods), typically all the N training samples are involved in the operations of a single iteration, resulting an $O(N)$ time complexity. Despite of the potentially fast convergence rates for the optimization error, the number of iterations that a batch algorithm can run within a fixed time budget is relatively small if the number of data samples is large. This might results in a very large optimization error, even higher than the estimation error that has a much slower convergence rate.

Stochastic learning algorithms typically refer to those where a single data sample (or a mini-batch samples) is involved in each iteration of the optimization process. In many stochastic algorithms, the $1/\sqrt{N}$ estimation error can be achieved. However, their optimization errors might decrease slowly, comparing with batch optimization algorithms.

Even with slow convergence rates of $\mathcal{E}_{\text{Opt}}$, the computational cost for each iteration is only $O(1)$, indicating that given a time budget $T = N$, stochastic algorithms can be executed for T iterations, while batch algorithm can only be executed for 1 iteration. Suppose a statistically optimal batch algorithm achieves the optimal linear convergence rate, since $\left(1 - \sqrt{\frac{\mu}{L}}\right)^1 >> \frac{1}{T}$, it is NOT computationally optimal. Theoretically, given N large enough such that the i.i.d. Assumption 1 holds, stochastic algorithms will almost always yield a lower generalization error. In the follows we use a simple example to illustrate this phenomenon.

We use the following linear regression problem as our objective:

$$\min_{\mathbf{x}} \mathbb{E}_{\xi_i \equiv \{A_i, \mathbf{b}_i\}} \|A_i \mathbf{x} - \mathbf{b}_i\|_2^2, \tag{11}$$

and the corresponding ERM is as follows:

$$\min_{\mathbf{x}} \frac{1}{N} \sum_{i=1}^{N} \|A_i \mathbf{x} - \mathbf{b}_i\|_2^2, \tag{12}$$

Each example $A_i \in \mathbb{R}^D$ is sampled from a D dimensional gaussian $\mathcal{N}(\mathbf{0}, I)$. The "true" objective $\mathbf{x}^*$ is also generated from $\mathcal{N}(\mathbf{0}, I)$. For noise-free experiments, the labels are generated by $\mathbf{b}_i = A_i^T \mathbf{x}^*$, while for noisy cases, we set

$$\mathbf{b}_i = A_i^T \mathbf{x}^* + \epsilon,$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma I)$. The deterministic gradient descent (GD) is used as the batch learning algorithms, and the stochastic gradient descent (SGD) is used as the stochastic learning algorithm. To test the generalization error, we use 1/3 of the samples for testing, and the rest are for training.

In Fig.3, we compare the testing error for GD and SGD using the noise-free dataset. The dataset sizes are $N = 40, 80, 160, 200, 400, 800, 1600, 2000, 4000, 8000$ and 10000. To ensure that the two algorithms are given the same time budget, in this experiment we run 1 iteration for GD, and N iterations for SGD. This figure clearly shows that the testing error of SGD decreases much faster than that of GD. The larger N is, the larger the difference. Since the data is noisy free, the estimation error should be 0. The slow convergence of GD

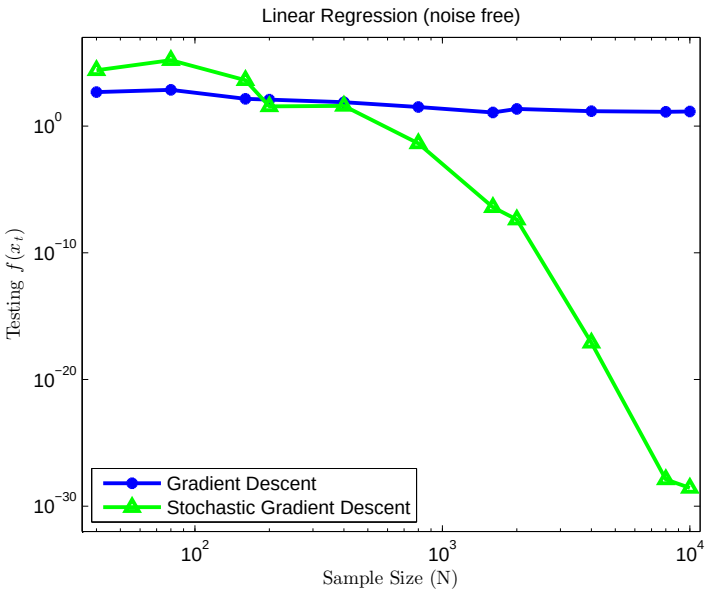

Figure 3: Testing values v.s. number of data samples (N). Solve least squares linear regression (noise free) with batch algorithm (gradient descent) and stochastic algorithm (stochastic gradient descent).

is due to the high optimization error, as discussed above. The fast convergence of SGD exhibits the optimization error (almost linear rate).

In Fig.4, we compare the performance of GD and SGD with two noisy datasets of different noise levels: $\sigma = 10^{-3}$ (Left) and $\sigma = 10^{-1}$ (Right). As in the previous experiment, we run 1 iteration for GD, and N iterations for SGD. We have several observations. First, like in the noise-free case, SGD outperforms GD for all situations when N is larger than 500. Second, the batch algorithm's performance is almost the same for the two noise levels. This indicates that the optimization error is indeed very large, and dominates the estimation error. Third, for both noise levels, SGD converges to the near-optimal errors (although the errors are different) after running almost the same number of iterations (around 2000). This indicates that the estimation error is now the dominating factor for SGD, which is different from the noise-free case.

In the third set of experiments, instead of running only 1 iteration for GD, we run it for 10, 20 and 50 batch iterations. The corresponding number of iterations for SGD are thus $10N$, $20N$ and $50N$ (i.e. 10, 20 and 50 epochs).

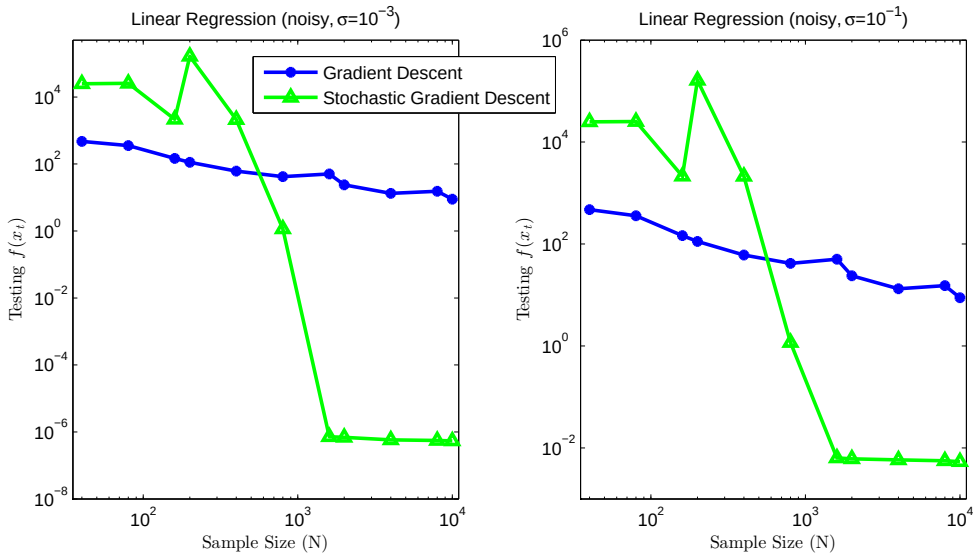

Figure 4: Testing values v.s. number of data samples (N). Solve noisy least squares linear regression with batch and stochastic algorithms. Left: $\sigma = 10^{-3}$. Right: $\sigma = 10^{-1}$.

In all the three experiments shown in Fig.5, the dataset is noise free. Although running stochastic algorithms for multiple epochs does not strictly satisfy the i.i.d. assumption, SGD still outperforms GD in all settings. The comparison of these three subfigures also reveals that when GD's number of iterations increases, its optimization error decreases, and are more and more comparable with that of SGD. However, even after 100 iterations, the generalization capability of GD is still not as good as SGD.

When the dataset's noise level is $\sigma = 10^{-3}$, we compare GD and SGD under three settings, with 5, 10 and 20 epochs each, and the results are plotted in Fig.6. Unlike the noise-free cases (Fig.5), the testing errors of GD and SGD are very close when the number of epochs is larger than 10. This is due to two reasons. First, the optimization error of GD is much lower after 10 or 20 iterations, hence the estimation error begins to dominate. Second, the i.i.d. assumption of SGD do not hold. However, when N is large (e.g. 10^4) and the noise level is higher than $\sigma = 10^{-3}$, a single stochastic sweep over the dataset is already enough to achieve the best prediction accuracy (Fig.4).

If the noise level decreases from 10^{-3} to 10^{-6} (Fig.7), the advantage of SGD becomes significant even for 10 epochs.

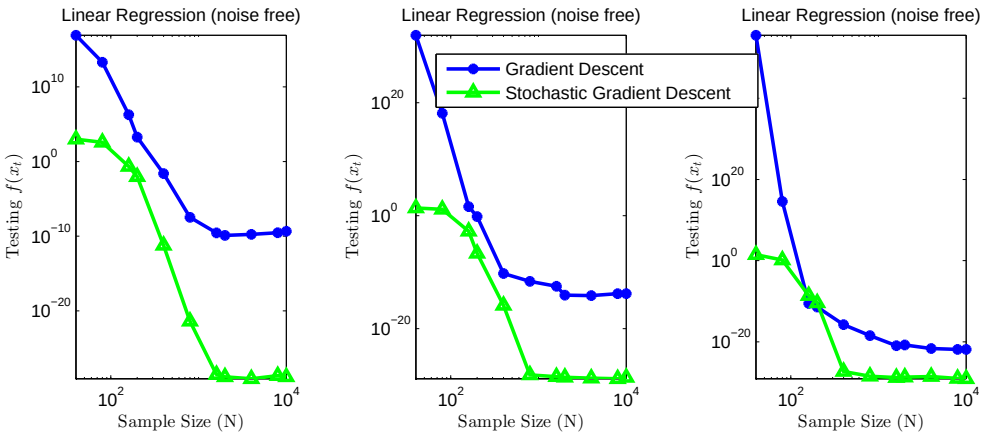

Figure 5: Testing values v.s. number of data samples (N). Solve least squares linear regression (noise free) with batch and stochastic algorithms. Left: 10 epochs. Middle: 20 epochs. Right: 50 epochs.

The above discussions and comparisons clearly demonstrate the trade-off between the estimation error and the optimization error. The final convergence behavior of the testing accuracy is dominated by the slower one.

1.2.4 Distributed Learning

In the previous section we analyzed the error sources of batch and stochastic algorithms. We illustrated by examples that in many situations stochastic algorithms exhibit very competitive generalization capacities with a much lower time complexity. However in many real-world applications, batch learning is still desired for various reasons. For example, sometimes a deterministic repeatable training process is preferred to avoid the variations of stochastic algorithms' performance. In these scenarios, the data-distributed learning is a natural way to deal with large-scale problems.

There are generally two classes of methods for the distributed learning in the literature. The first class includes the gradient-based primal methods: e.g. the distributed subgradient descent methods (Nedic and Ozdaglar [2009], Dekel et al. [2011]) and the distributed dual averaging methods (Duchi et al. [2010], Agarwal and Duchi [2011], Duchi et al. [2012]). To

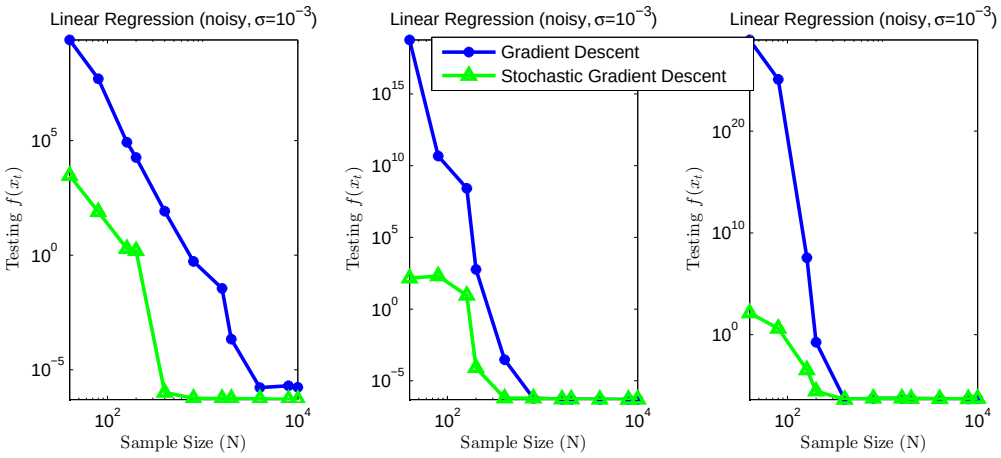

Figure 6: Testing values v.s. number of data samples (N). Solve noisy ($\sigma = 10^{-3}$) least squares linear regression with batch and stochastic algorithms. Left: 5 epochs. Middle: 10 epochs. Right: 20 epochs.

solve the batch ERM problem (the $1/N$ term is omitted)

$$\min_{\mathbf{x}\in\mathcal{X}} \sum_{i=1}^{N} f(\mathbf{x}, \xi_i), \tag{13}$$

the most straightforward method is to distribute the data samples $\{\xi\}_{i=1}^{N}$ evenly to S worker nodes (slaves):

$$\min_{\mathbf{x}\in\mathcal{X}} \sum_{s=1}^{S} \sum_{i=1}^{N_s} f(\mathbf{x}, \xi_i). \tag{14}$$

At iteration t, each worker calculates its partial (sub)gradients based on the N/S samples and send the vectors to the master node. The master receives all the S partial results, aggregate (average and project to $\mathcal{X}$) and broadcast the new solution $\mathbf{x}_{t+1}$ to all the workers and proceed with the next iteration. If all the operations are synchronous, this simple parallelization will have the same convergence behavior as the single-node-based gradient descent, and is S times faster. However, the communication and computation cost on the master node is the bottleneck of this method. In some applications, this master-slave communication topology is even not available, and all the workers might be just loosely connected with each other (Predd et al. [2007]).

The second class are the primal-dual methods based on the augmented Lagrangian method (Zhu et al. [2009]) or the alternating direction method of multipliers (ADMM) (Boyd et al. [2010], Mateos et al. [2010], Mota et al. [2012]). In gradient-based methods,

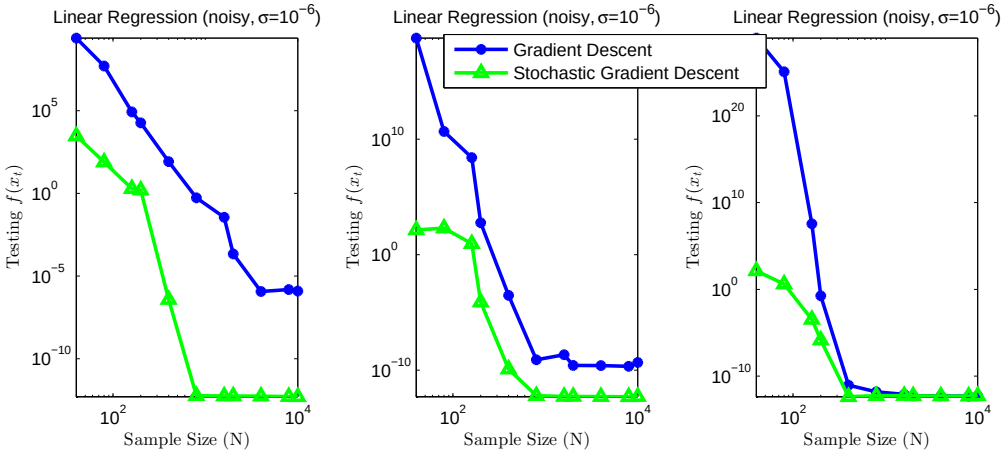

Figure 7: Testing values v.s. number of data samples (N). Solve noisy ($\sigma = 10^{-6}$) least squares linear regression with batch and stochastic algorithms. Left: 5 epochs. Middle: 10 epochs. Right: 20 epochs.

the (sub)gradients are transmitted and aggregated in the hope that all workers will asymptotically obtain information from all data samples. While for the second class, the consensus requirements are *explicitly* encoded as constraints, as given by the following formulation.

$$\begin{aligned} &\min_{\mathbf{x}_s \in \mathcal{X}} \sum_{s=1}^{S} \sum_{i=1}^{N_s} f(\mathbf{x}_s, \xi_i) \\ &\text{s.t. } \mathbf{x}_1 = \mathbf{x}_2 = \ldots = \mathbf{x}_S, \end{aligned} \tag{15}$$

The decentralized learning problem can be easily modeled by the consensus equality constraints.

We propose to solve problem (15) by ADMM in parallel. To take advantage of ADMM's capacities in dealing with separable functions, we divide the workers into two groups and design a *bipartite graph* for communication. Bipartite graphs are general enough to subsume many popular communication topologies. For example the master-slave setting is an unbalanced bipartite graph where the master is on one part, and the slaves are on the other. Trees and hypercubes are also bipartite examples that are very popular in general distributed computation (Bertsekas and Tsitsiklis [1997]).

We focus on the convergence behavior of the proposed consensus ADMM algorithm and we want to investigate how it will be affected by the various factors of our problem. One of the central themes in distributed learning is the question "What is the *optimal*

communication topology?" To reach a definitive answer to this question, one still needs to overcome major hurdles because the convergence behavior of ADMM in this context not only depends on the communication topology, but also on the penalty parameter β used in the augmented Lagrangian. The main focus of our work is to characterize the interplay between these factors.

1.3 Outline and Main Contributions

In this dissertation we investigate the three main sources of the generalization error: the approximation error, originated from the improper hypothesis space and problem models, the estimation error, a quantity that depends on the variation and the number of data examples, and the optimization error that directly depends on the specific optimization technique being used in the learning process. A deeper study on the lower bounds of the estimation and optimization errors leads to the concepts of optimal learning algorithms. A statistically optimal algorithm achieves these lower bounds without considering the computational cost. A computationally optimal algorithm could achieve the lower bounds with only a single sweep of the data. Guided by these analysis, we are able to derive several scalable optimization algorithms for various machine learning problems, where the structures of these problems are explored.

In Chapter II we first investigate stochastic methods for solving nonsmooth problems in machine learning. Among these problems the support vector machine is one of the most important examples that is widely used in many applications. This work is motivated by the fact that the lower bound of the optimization error for minimizing nonsmooth functions is $O(1/\sqrt{t})$, which is at least of the same order as the estimation error lower bound. We therefore propose a stochastic smoothing method to approximate the original nonsmooth function with a Lipschitz smooth surrogate. By carefully controlling the degree of the smoothness and the approximation we are able to achieve the lower bounds for both the estimation and optimization errors. The fast rates are confirmed by empirical comparisons, in which the derived accelerated nonsmooth stochastic gradient descent (ANSGD) significantly outperforms previous subgradient descent algorithms including SGD and averaged

SGD in several real-world applications.

In Chapter III we are interested in a common problem structure in machine learning: the separability of the objective functions. This property subsumes the very general pattern for the regularized risk minimization problems. We study a family of convex optimization problems where our objective functions are stochastic and composite. Our starting point is the classic alternating direction method of multipliers (ADMM) that provides a very flexible framework in dealing with separabilities. The proposed stochastic ADMM algorithm takes the advantages of both the scalability and the separability. It applies to a more general class of convex and nonsmooth objective functions, beyond the smooth and separable least squares loss used in lasso. A novel model named graph-guided SVM (GGSVM) is proposed with which one can easily encode complex relations between features as a graph-lasso prior. GGSVM can be easily solved by the stochastic ADMM, and it exhibits significantly higher prediction accuracies than the traditional SVM without using the graph-lasso prior.

In Chapter IV we extend the scalability of stochastic algorithms to nonlinear machine learning models, i.e. the kernelized support vector machine, where the problem is formulated as a constrained dual quadratic optimization. The simplex constraint can be handled by the classic Frank-Wolfe method (also known as the constrained gradient). The proposed stochastic Frank-Wolfe method maintains a stochastic set of support vectors. In each iteration the incoming random sample is greedily determined to be included in the support vector set or not, and the updates on the dual variables are all of closed-forms. SFW achieves comparable or even better accuracies than state-of-the-art batch and online algorithms, and are significantly faster.

In Chapter V we consider the data-distributed deterministic learning. This is a very important problem in many large-scale machine learning systems where data samples are distributed over hundreds or thousands of general purpose servers. Locally accessing data is typically faster than the remote access due to the latency of network communication and limited bandwidth. We formulate the distributed learning problem as a consensus constrained optimization problem and solve it using the general methodology of Alternating Direction Method of Multipliers (ADMM) (Glowinski and Marroco [1975], Gabay and

Mercier [1976]). We used bipartite communication topologies to take advantage of ADMM's capacities in dealing with separable functions. We identify the three degrees of freedom in implementing this method: communication topology, penalty parameter β and the order for updating variables. In order to investigate the joint effects of these factors, we provided an analysis of ADMM's convergence behavior. The analysis demonstrates that all the primal and dual variables enjoy a linear rate of convergence. Due to the difficulty in obtaining a very sharp rate from which the optimal β^* can be derived, we proposed a strategy for choosing β adaptively, with an underestimated initial guess β_0 that is derived from our bound. Numerical experiments show that β^* is achieved at a point where the norms of primal and dual residuals are close and decrease at the fastest rate. With β^*, the complete bipartite and the master-slave graphs converge fastest, followed by bi-regular graphs. The proposed strategy of adaptive β is very efficient.

In Chapter VI we conclude this dissertation and present several promising research directions that deserve long-term investigations.

CHAPTER II

STOCHASTIC SMOOTHING

2.1 Introduction

In this chapter we investigate the nonsmoothness of functions in machine learning. This property is a central issue in machine learning computations, since many important models minimize nonsmooth convex functions. For example, using the nonsmooth hinge loss yields sparse support vector machines; regressors can be made robust to outliers by using the nonsmooth absolute loss other than the squared loss; the $l1$-norm is widely used in sparse reconstructions. In spite of the attractive properties, nonsmooth functions are theoretically more difficult to optimize than smooth functions (Nemirovski and Yudin [1983]). In this chapter we focus on minimizing nonsmooth functions where the functions are either stochastic (stochastic optimization), or learning samples are provided incrementally (online learning).

Smoothness and strong-convexity are typically certificates of the existence of fast global solvers. Nesterov's deterministic smoothing method (Nesterov [2005b]) deals with the difficulty of nonsmooth functions by approximating them with smooth functions, for which optimal methods (Nesterov [2004]) can be applied. It converges as $f(\mathbf{x}_t) - \min_{\mathbf{x}} f(\mathbf{x}) \leq O(1/t)$ after t iterations. If a nonsmooth function is strongly convex, this rate can be improved to $O(1/t^2)$ using the excessive gap technique (Nesterov [2005a]).

In this chapter, we extend Nesterov's smoothing method to the stochastic setting by proposing a stochastic smoothing method for nonsmooth functions. Combining this with a stochastic version of the optimal gradient descent method, we introduce and analyze a new algorithm named Accelerated Nonsmooth Stochastic Gradient Descent (ANSGD), for a class of functions that include the popular ML methods of interest.

To our knowledge ANSGD is the first stochastic first-order algorithm that can achieve the optimal $O(1/t)$ rate for minimizing nonsmooth loss functions without Polyak's averaging

(Polyak and Juditsky [1992]). In comparison, the classic SGD converges in $O(\ln t/t)$ for non-smooth strongly convex functions (Shalev-Shwartz et al. [2007]), and is usually not robust (Nemirovski et al. [2009]). Even with Polyak's averaging (Bach and Moulines [2011], Xu [2011]), there are cases where SGD's convergence rate still can not be faster than $O(\ln t/t)$ (Shamir [2011]). Numerical experiments on real-world datasets also indicate that ANSGD converges much faster in comparing with these state-of-the-art algorithms.

A perturbation-based smoothing method is recently proposed for stochastic nonsmooth minimization (Duchi et al. [2011]). This work achieves similar iteration complexities as ours, in a parallel computation scenario. In serial settings, ANSGD enjoys better and optimal bounds.

In machine learning, many problems can be cast as minimizing a composition of a loss function and a regularization term. Before proceeding to the algorithm, we first describe a different setting of "composite minimizations" that we will pursue in this chapter, along with our notations and assumptions.

2.1.1 A Different "Composite Setting"

In the classic *black-box* setting of first-order stochastic algorithms (Nemirovski et al. [2009]), the structure of the objective function $\min_{\mathbf{x}}\{f(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi}) : \boldsymbol{\xi} \sim P\}$ is unknown. In each iteration t, an algorithm can only access the first-order stochastic oracle and obtain a subgradient $f'(\mathbf{x}, \boldsymbol{\xi}_t)$. The basic assumption is that $f'(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\xi}} f'(\mathbf{x}, \boldsymbol{\xi})$ for any $\mathbf{x}$, where the random vector $\boldsymbol{\xi}$ is from a fixed distribution P.

The *composite setting* (also known as *splitting* (Lions and Mercier [1979])) is an extension of the black-box model. It was proposed to exploit the structure of objective functions. Driven by applications of sparse signal reconstruction, it has gained significant interest from different communities (Daubechies et al. [2004], Beck and Teboulle [2009], Nesterov [2007a]). Stochastic variants have also been proposed recently (Lan [2010], Lan and Ghadimi [2011], Duchi and Singer [2009], Hu et al. [2009], Xiao [2010]). A stochastic composite function $\Phi(\mathbf{x}) \equiv f(\mathbf{x}) + g(\mathbf{x})$ is the sum of a smooth stochastic convex function $f(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$ and a nonsmooth (but simple and deterministic) function $g()$. To minimize Φ, previous work

construct the following model iteratively:

$$\langle \nabla f(\mathbf{x}_t, \boldsymbol{\xi}_t), \mathbf{x} - \mathbf{x}_t \rangle + \frac{1}{\eta_t} D(\mathbf{x}, \mathbf{x}_t) + g(\mathbf{x}), \tag{16}$$

where $\nabla f(\mathbf{x}_t, \boldsymbol{\xi}_t)$ is a gradient, $D(\cdot, \cdot)$ is a proximal function (typically a Bregman divergence) and η_t is a stepsize.

A successful application of the composite idea typically relies on the assumption that model (16) is easy to minimize. If $g()$ is very simple, e.g. $\|\mathbf{x}\|_1$ or the nuclear norm, it is straightforward to obtain the minimum in analytic forms. However, this assumption does not hold for many other applications in machine learning, where many loss functions (not the regularization term, here the nonsmooth $g()$ becomes the nonsmooth loss function) are nonsmooth, and do not enjoy separability properties (Wright et al. [2009]). This includes important examples such as hinge loss, absolute loss, and ϵ-insensitive loss.

In this chapter, we tackle this problem by studying a new stochastic composite setting: $\min_{\mathbf{x}} \Phi(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$, where loss function $f()$ is convex and nonsmooth, while $g()$ is convex and L_g-Lipschitz smooth:

$$g(\mathbf{x}) \leq g(\mathbf{y}) + \langle \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{L_g}{2} \|\mathbf{x} - \mathbf{y}\|^2. \tag{17}$$

For clarity, in this chapter we focus on unconstrained minimizations. Without loss of generality, we assume that both $f()$ and $g()$ are stochastic: $f(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$ and $g(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\xi}} g(\mathbf{x}, \boldsymbol{\xi})$, where $\boldsymbol{\xi}$ has distribution P. If either one is deterministic, its $\boldsymbol{\xi}$ is then dropped. To make our algorithm and analysis more general, we assume that $g()$ is μ-strongly convex: $\forall \mathbf{x}, \mathbf{y}$,

$$g(\mathbf{x}) \geq g(\mathbf{y}) + \langle \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2. \tag{18}$$

If it is not strongly convex, one can simply take $\mu = 0$.

The main idea of our algorithm again stems from exploiting the structures of $f()$ and $g()$. In Section 2.2 we propose to form a smooth stochastic approximation of $f()$, such that the optimal methods (Nesterov [2004]) can be applied to attain optimal convergence rates. The convergence of our proposed algorithm is analyzed in Section 3.3, and a batch-to-online conversion is also proposed. Two popular machine learning problems are chosen as our examples in Section 2.4, and numerical evaluations are presented in Section 5.6.

2.2 Approach

2.2.1 Stochastic Smoothing Method

An important breakthrough in nonsmooth minimization was made by Nesterov in a series of works (Nesterov [2005b,a, 2007b]). By exploiting function structures, Nesterov shows that in many applications, minimizing a well-structured nonsmooth function $f(\mathbf{x})$ can be formulated as an equivalent saddle-point form

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{u}\in\mathcal{U}} \Big[\langle A\mathbf{x}, \mathbf{u}\rangle - Q(\mathbf{u})\Big], \tag{19}$$

where $\mathbf{u} \in \mathbb{R}^m$, $\mathcal{U} \subseteq \mathbb{R}^m$ is a convex set, A is a linear operator mapping $\mathbb{R}^D \to \mathbb{R}^m$ and $Q(\mathbf{u})$ is a continuous convex function. Inserting a non-negative ζ-strongly convex function $\omega(\mathbf{u})$ in (19) one obtains a smooth approximation of the original nonsmooth function

$$\hat{f}(\mathbf{x}, \gamma) \equiv \max_{\mathbf{u}\in\mathcal{U}} \Big[\langle A\mathbf{x}, \mathbf{u}\rangle - Q(\mathbf{u}) - \gamma\omega(\mathbf{u})\Big], \tag{20}$$

where $\gamma > 0$ is a fixed *smoothness parameter* which is crucial in the convergence analysis. The key property of this approximation is:

Lemma 1. *(Nesterov [2005b])Function $\hat{f}(\mathbf{x}, \gamma)$ is convex and continuously differentiable, and its gradient is Lipschitz continuous with constant $L_{\hat{f}} \equiv \frac{\|A\|^2}{\gamma\zeta}$, where*

$$\|A\| \equiv \max_{\mathbf{x},\mathbf{u}} \{\langle A\mathbf{x}, \mathbf{u}\rangle : \|\mathbf{x}\| = 1, \|\mathbf{u}\| = 1\}. \tag{21}$$

Nesterov's smoothing method was originally proposed for deterministic optimization. A major drawback of this method is that the number of iterations N must be known beforehand, such that the algorithm can set a proper smoothness parameter $\gamma = O(\frac{2\|A\|}{N+1})$ to ensure convergence. This makes it unsuitable for algorithms that runs forever, or whose number of iterations is not known. Following his work we propose to extend this smoothing method to stochastic optimization. Our stochastic smoothing differs from the deterministic one in the operator A and smoothness parameter γ, where both will be time-varying.

We assume that the nonsmooth part $f(\mathbf{x}, \boldsymbol{\xi})$ of the stochastic composite function $\Phi()$ is well structured, i.e. for a specific realization $\boldsymbol{\xi}_t$, it has an equivalent form like the max function in (19):

$$f(\mathbf{x}, \boldsymbol{\xi}_t) = \max_{\mathbf{u}\in\mathcal{U}} \Big[\langle A_{\boldsymbol{\xi}_t}\mathbf{x}, \mathbf{u}\rangle - Q(\mathbf{u})\Big], \tag{22}$$

where $A_{\boldsymbol{\xi}_t}$ is a stochastic linear operator associated with $\boldsymbol{\xi}_t$. We construct a smooth approximation of this function as:

$$\hat{f}(\mathbf{x}, \boldsymbol{\xi}_t, \gamma_t) \equiv \max_{\mathbf{u} \in \mathcal{U}} \Big[\langle A_{\boldsymbol{\xi}_t} \mathbf{x}, \mathbf{u} \rangle - Q(\mathbf{u}) - \gamma_t \omega(\mathbf{u}) \Big], \tag{23}$$

where γ_t is a time-varying smoothness parameter only associated with iteration index t, and is independent of $\boldsymbol{\xi}_t$. Function $\omega()$ is non-negative and ζ-strongly convex. Due to Lemma 1, $\hat{f}(\mathbf{x}, \boldsymbol{\xi}_t, \gamma_t)$ is $\frac{\|A_{\boldsymbol{\xi}_t}\|^2}{\gamma_t \zeta}$-Lipschitz smooth. It follows that

Lemma 2. $\forall \mathbf{x}, \mathbf{y}, t$, $\mathbb{E}_{\boldsymbol{\xi}} \hat{f}(\mathbf{x}, \boldsymbol{\xi}, \gamma_t) \leq \mathbb{E}_{\boldsymbol{\xi}} \hat{f}(\mathbf{y}, \boldsymbol{\xi}, \gamma_t) + \mathbb{E}_{\boldsymbol{\xi}} \langle \nabla \hat{f}(\mathbf{y}, \boldsymbol{\xi}, \gamma_t), \mathbf{x} - \mathbf{y} \rangle + \frac{\mathbb{E}_{\boldsymbol{\xi}} \|A_{\boldsymbol{\xi}}\|^2}{\gamma_t \zeta} \|\mathbf{x} - \mathbf{y}\|^2$.

We have the following observation about our composite objective $\Phi()$, which relates the reduction of the original and approximated function values.

Lemma 3. *For any* $\mathbf{x}, \mathbf{x}_t, t$,

$$\Phi(\mathbf{x}_t) - \Phi(\mathbf{x}) \leq \mathbb{E}_{\boldsymbol{\xi}} \Big[\hat{f}(\mathbf{x}_t, \boldsymbol{\xi}, \gamma_t) + g(\mathbf{x}_t, \boldsymbol{\xi}) \Big] - \mathbb{E}_{\boldsymbol{\xi}} \Big[\hat{f}(\mathbf{x}, \boldsymbol{\xi}, \gamma_t) + g(\mathbf{x}, \boldsymbol{\xi}) \Big] + \gamma_t D_{\mathcal{U}}, \tag{24}$$

where $D_{\mathcal{U}} \equiv \max_{\mathbf{u} \in \mathcal{U}} \omega(\mathbf{u})$.

Proof.

$$\begin{aligned}
&\Phi(\mathbf{x}_t) - \Phi(\mathbf{x}) \\
&= [f(\mathbf{x}_t) - f(\mathbf{x})] + [g(\mathbf{x}_t) - g(\mathbf{x})] \\
&= \mathbb{E}_{\boldsymbol{\xi}} [f(\mathbf{x}_t, \boldsymbol{\xi})] + \mathbb{E}_{\boldsymbol{\xi}} [-f(\mathbf{x}, \boldsymbol{\xi}) + g(\mathbf{x}_t, \boldsymbol{\xi}) - g(\mathbf{x}, \boldsymbol{\xi})] \\
&= \mathbb{E}_{\boldsymbol{\xi}} \max_{\mathbf{u} \in \mathcal{U}} \Big\{ [\langle A_{\boldsymbol{\xi}} \mathbf{x}_t, \mathbf{u} \rangle - Q(\mathbf{u}) - \gamma_t \omega(\mathbf{u})] + \gamma_t \omega(\mathbf{u}) \Big\} + \mathbb{E}_{\boldsymbol{\xi}} [-f(\mathbf{x}, \boldsymbol{\xi}) + g(\mathbf{x}_t, \boldsymbol{\xi}) - g(\mathbf{x}, \boldsymbol{\xi})] \\
&\leq \mathbb{E}_{\boldsymbol{\xi}} \max_{\mathbf{u} \in \mathcal{U}} [\langle A_{\boldsymbol{\xi}} \mathbf{x}_t, \mathbf{u} \rangle - Q(\mathbf{u}) - \gamma_t \omega(\mathbf{u})] + \max_{\mathbf{u} \in \mathcal{U}} [\gamma_t \omega(\mathbf{u})] + \mathbb{E}_{\boldsymbol{\xi}} [-f(\mathbf{x}, \boldsymbol{\xi}) + g(\mathbf{x}_t, \boldsymbol{\xi}) - g(\mathbf{x}, \boldsymbol{\xi})] \\
&= \mathbb{E}_{\boldsymbol{\xi}} \Big[\hat{f}(\mathbf{x}_t, \boldsymbol{\xi}, \gamma_t) \Big] + \gamma_t D_{\mathcal{U}} + \mathbb{E}_{\boldsymbol{\xi}} [-f(\mathbf{x}, \boldsymbol{\xi}) + g(\mathbf{x}_t, \boldsymbol{\xi}) - g(\mathbf{x}, \boldsymbol{\xi})] \\
&\leq \mathbb{E}_{\boldsymbol{\xi}} \Big[\hat{f}(\mathbf{x}_t, \boldsymbol{\xi}, \gamma_t) - \hat{f}(\mathbf{x}, \boldsymbol{\xi}, \gamma_t) \Big] + \mathbb{E}_{\boldsymbol{\xi}} [g(\mathbf{x}_t, \boldsymbol{\xi}) - g(\mathbf{x}, \boldsymbol{\xi})] + \gamma_t D_{\mathcal{U}}.
\end{aligned} \tag{25}$$

The last inequality is due to the non-negativity of $\omega()$ and definitions of f (22) and $\hat{f}$ (23). □

2.2.2 Accelerated Nonsmooth SGD (ANSGD)

We are now ready to present our algorithm ANSGD (Algorithm 1). This stochastic algorithm is obtained by applying Nesterov's optimal method to our smooth surrogate function, and thus has a similar form to that of his original deterministic method (Nesterov [2004](p.78)). However, our convergence analysis is more straightforward, and does not rely on the concept of estimate sequences. Hence it is easier to identify proper series $\gamma_t, \eta_t, \alpha_t$ and θ_t that are crucial in achieving fast rates of convergence. These series will be determined in our main results (Thm.5 and 6).

Algorithm 1 Accelerated Nonsmooth Stochastic Gradient Descent (ANSGD)

INPUT: series $\gamma_t,\ \eta_t,\ \theta_t \geq 0$ and $0 \leq \alpha_t \leq 1$;
OUTPUT: $\mathbf{x}_{t+1}$;
0. Initialize $\mathbf{x}_0$ and $\mathbf{v}_0$;
for $t = 0, 1, 2, \ldots$ **do**
1. $\mathbf{y}_t \leftarrow \frac{(1-\alpha_t)(\mu+\theta_t)\mathbf{x}_t + \alpha_t\theta_t\mathbf{v}_t}{\mu(1-\alpha_t)+\theta_t}$
2. $\hat{f}_{t+1}(\mathbf{x}) \leftarrow \max_{\mathbf{u}\in\mathcal{U}} \left[\langle A_{\boldsymbol{\xi}_{t+1}}\mathbf{x}, \mathbf{u}\rangle - Q(\mathbf{u}) - \gamma_{t+1}\omega(\mathbf{u})\right]$
3. $\mathbf{x}_{t+1} \leftarrow \mathbf{y}_t - \eta_t\left[\nabla\hat{f}_{t+1}(\mathbf{y}_t) + \nabla g_{t+1}(\mathbf{y}_t)\right]$
4. $\mathbf{v}_{t+1} \leftarrow \frac{\theta_t\mathbf{v}_t + \mu\mathbf{y}_t - \left[\nabla\hat{f}_{t+1}(\mathbf{y}_t) + \nabla g_{t+1}(\mathbf{y}_t)\right]}{\mu+\theta_t}$
end for

2.3 Convergence Analysis

To clarify our presentation, we use Table 1 to list some notations that will be used throughout the chapter.

Table 1: Some notations.

Symbol	Meaning
$\hat{f}_t(\mathbf{x}),\ g_t(\mathbf{x})$	$\hat{f}(\mathbf{x}, \boldsymbol{\xi}_t, \gamma_t),\ g(\mathbf{x}, \boldsymbol{\xi}_t)$
$\nabla\hat{f}_t(\mathbf{x}),\ \nabla g_t(\mathbf{x})$	$\nabla\hat{f}(\mathbf{x}, \boldsymbol{\xi}_t, \gamma_t),\ \nabla g(\mathbf{x}, \boldsymbol{\xi}_t)$
L_t	$L_g + \frac{\|A_{\boldsymbol{\xi}_t}\|^2}{\gamma_t\zeta}$
$\sigma_t(\mathbf{x})$	$[\nabla\hat{f}_t(\mathbf{x}) + \nabla g_t(\mathbf{x})] - \mathbb{E}_{\boldsymbol{\xi}_t}[\nabla\hat{f}_t(\mathbf{x}) + \nabla g_t(\mathbf{x})]$
σ^2	$\mathbb{E}\max_t \|\sigma_{t+1}(\mathbf{y}_t)\|^2$
Δ_t	$\mathbb{E}_{\boldsymbol{\xi}_t}\left[\hat{f}_t(\mathbf{x}_t) + g_t(\mathbf{x}_t)\right] - \mathbb{E}_{\boldsymbol{\xi}_t}\left[\hat{f}_t(\mathbf{x}) + g(\mathbf{x})\right]$
Γ_{t+1}	$\langle\sigma_{t+1}(\mathbf{y}_t), \alpha_t\mathbf{x} + (1-\alpha_t)\mathbf{x}_t - \mathbf{y}_t\rangle$
D_t^2	$\frac{1}{2}\mathbb{E}\|\mathbf{x} - \mathbf{v}_t\|^2$

Our convergence rates are based on the following main lemma, which bounds the progressive reduction Δ_t of the smoothed function value. Actually Line 1, 3, and 4 of Alg.1 are also derived from the proof of this lemma.

Lemma 4. *Let γ_t be monotonically decreasing. Applying algorithm ANSGD to nonsmooth composite function $\Phi()$, we have $\forall \mathbf{x}$ and $\forall t \geq 0$,*

$$\begin{aligned} &\Delta_{t+1} \leq (1-\alpha_t)\Delta_t + (1-\alpha_t)(\gamma_t - \gamma_{t+1})D_{\mathcal{U}} + \\ &\Gamma_{t+1} + \frac{\alpha_t}{2}\left[\theta_t\|\mathbf{x}-\mathbf{v}_t\|^2 - (\mu+\theta_t)\|\mathbf{x}-\mathbf{v}_{t+1}\|^2\right] + \\ &\eta_t pq + \left[\frac{\alpha_t}{2(\mu+\theta_t)} + \frac{L_{t+1}}{2}\eta_t^2 - \eta_t\right] q^2 \end{aligned} \tag{26}$$

where $p \equiv \|\sigma_{t+1}(\mathbf{y}_t)\|$ and $q \equiv \|\nabla \hat{f}_{t+1}(\mathbf{y}_t) + \nabla g_{t+1}(\mathbf{y}_t)\|$.

Before proceeding to the proof of this main lemma, we present two auxiliary results. For clarity, in the following lemmas and proofs we use the following notations to denote the smoothly approximated composite function and its expectation:

$$F_t(\mathbf{x}, \gamma_t) \equiv \hat{f}_t(\mathbf{x}) + g_t(\mathbf{x}) = \hat{f}(\mathbf{x}, \boldsymbol{\xi}_t, \gamma_t) + g(\mathbf{x}, \boldsymbol{\xi}_t) \tag{27}$$

and

$$F(\mathbf{x}, \gamma_t) \equiv \mathbb{E}_{\boldsymbol{\xi}_t} F_t(\mathbf{x}, \gamma_t). \tag{28}$$

The first lemma is on the smoothly approximated function and the smoothness parameter γ_t.

Lemma 5. *If γ_t is monotonically decreasing with t, for any $\mathbf{x}$ and $t \geq 0$,*

$$F(\mathbf{x}, \gamma_t) \leq F(\mathbf{x}, \gamma_{t+1}) \leq F(\mathbf{x}, \gamma_t) + (\gamma_t - \gamma_{t+1})D_{\mathcal{U}}, \tag{29}$$

where $D_{\mathcal{U}} \equiv \max_{\mathbf{u}\in\mathcal{U}} \omega(\mathbf{u})$.

Proof. The left inequality is obvious, since $\gamma_t \geq \gamma_{t+1}$ and $\omega(\mathbf{u})$ is nonnegative. For the right

inequality,

$$
\begin{aligned}
&F(\mathbf{x}, \gamma_{t+1}) - F(\mathbf{x}, \gamma_t) \\
&= \mathbb{E}_{\boldsymbol{\xi}} \hat{f}(\mathbf{x}, \boldsymbol{\xi}, \gamma_{t+1}) - \mathbb{E}_{\boldsymbol{\xi}} \hat{f}(\mathbf{x}, \boldsymbol{\xi}, \gamma_t) \\
&= \max_{\mathbf{u} \in \mathcal{U}} \left[\langle \mathbb{E}_{\boldsymbol{\xi}} A_{\boldsymbol{\xi}} \mathbf{x}, \mathbf{u} \rangle - Q(\mathbf{u}) - \gamma_{t+1} \omega(\mathbf{u}) \right] - \max_{\mathbf{u} \in \mathcal{U}} \left[\langle \mathbb{E}_{\boldsymbol{\xi}} A_{\boldsymbol{\xi}} \mathbf{x}, \mathbf{u} \rangle - Q(\mathbf{u}) - \gamma_t \omega(\mathbf{u}) \right] \\
&\leq \max_{\mathbf{u} \in \mathcal{U}} \left\{ \left[\langle \mathbb{E}_{\boldsymbol{\xi}} A_{\boldsymbol{\xi}} \mathbf{x}, \mathbf{u} \rangle - Q(\mathbf{u}) - \gamma_{t+1} \omega(\mathbf{u}) \right] - \left[\langle \mathbb{E}_{\boldsymbol{\xi}} A_{\boldsymbol{\xi}} \mathbf{x}, \mathbf{u} \rangle - Q(\mathbf{u}) - \gamma_t \omega(\mathbf{u}) \right] \right\} \\
&= \max_{\mathbf{u} \in \mathcal{U}} \left[(\gamma_t - \gamma_{t+1}) \omega(\mathbf{u}) \right].
\end{aligned} \tag{30}
$$

□

The second lemma is about proximal methods using Bregman divergence as prox-functions, which is a direct result of optimality conditions. It appeared in Lan and Ghadimi [2011](Lemma 2), and is an extension of the "3-point identity" (Chen and Teboulle [1993](Lemma 3.1)).

Lemma 6. *(Lan and Ghadimi [2011]) Let $l(\mathbf{x})$ be a convex function. Let scalars $s_1, s_2 \geq 0$. For any vectors $\mathbf{u}$ and $\mathbf{v}$, denote their Bregman divergence as $D(\mathbf{u}, \mathbf{v})$. If $\forall \mathbf{x}, \mathbf{u}, \mathbf{v}$*

$$
\mathbf{x}^* = \arg\min_{\mathbf{x}} l(\mathbf{x}) + s_1 D(\mathbf{u}, \mathbf{x}) + s_2 D(\mathbf{v}, \mathbf{x}), \tag{31}
$$

then

$$
l(\mathbf{x}) + s_1 D(\mathbf{u}, \mathbf{x}) + s_2 D(\mathbf{v}, \mathbf{x}) \geq l(\mathbf{x}^*) + s_1 D(\mathbf{u}, \mathbf{x}^*) + s_2 D(\mathbf{v}, \mathbf{x}^*) + (s_1 + s_2) D(\mathbf{x}^*, \mathbf{x}). \tag{32}
$$

We are now ready to prove Lemma 4.

Proof of Lemma 4. Due to Lemma 2 and Lipschitz-smoothness of $g(\mathbf{x})$, $F(\mathbf{x}, \gamma_{t+1})$ has a

Lipschitz smooth constant $L_{F_{t+1}} \equiv \frac{\mathbb{E}_{\boldsymbol{\xi}}\|A_{\boldsymbol{\xi}}\|^2}{\gamma_{t+1}\zeta} + L_g$. It follows that

$$
\begin{aligned}
&F(\mathbf{x}_{t+1}, \gamma_{t+1}) \\
&\leq F(\mathbf{y}_t, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t \rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \\
&= (1-\alpha_t)F(\mathbf{y}_t, \gamma_{t+1}) + \alpha_t F(\mathbf{y}_t, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t \rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \\
&= (1-\alpha_t)F(\mathbf{y}_t, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t) \rangle + \\
&\quad \alpha_t F(\mathbf{y}_t, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t) \rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \\
&\leq (1-\alpha_t)F(\mathbf{x}_t, \gamma_{t+1}) + \alpha_t F(\mathbf{y}_t, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t) \rangle + \\
&\quad \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2,
\end{aligned}
\tag{33}
$$

where the last inequality is due to the convexity of $F()$. Subtracting $F(\mathbf{x}, \gamma_{t+1})$ from both sides of the above inequality we have:

$$
\begin{aligned}
&F(\mathbf{x}_{t+1}, \gamma_{t+1}) - F(\mathbf{x}, \gamma_{t+1}) \leq (1-\alpha_t)F(\mathbf{x}_t, \gamma_{t+1}) - F(\mathbf{x}, \gamma_{t+1}) \\
&\quad + \alpha_t F(\mathbf{y}_t, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t) \rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \\
&\leq (1-\alpha_t)\big[F(\mathbf{x}_t, \gamma_t) + (\gamma_t - \gamma_{t+1})D_{\mathcal{U}}\big] - F(\mathbf{x}, \gamma_{t+1}) \\
&\quad + \alpha_t F(\mathbf{y}_t, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t) \rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \\
&\leq (1-\alpha_t)\big[F(\mathbf{x}_t, \gamma_t) - F(\mathbf{x}, \gamma_t)\big] - \alpha_t F(\mathbf{x}, \gamma_{t+1}) + (1-\alpha_t)(\gamma_t - \gamma_{t+1})D_{\mathcal{U}} \\
&\quad + \alpha_t F(\mathbf{y}_t, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t) \rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2,
\end{aligned}
\tag{34}
$$

where the last two inequalities are due to Lemma 5.

Denoting $\Delta_t \equiv F(\mathbf{x}_t, \gamma_t) - F(\mathbf{x}, \gamma_t)$ and $\sigma_t(\mathbf{x}) \equiv \nabla F_t(\mathbf{x}, \gamma_t) - \nabla F(\mathbf{x}, \gamma_t)$ we can rewrite

(34) as:

$$
\begin{aligned}
&\Delta_{t+1} - (1-\alpha_t)\Delta_t - (1-\alpha_t)(\gamma_t - \gamma_{t+1})D_{\mathcal{U}} \\
&\leq \alpha_t F(\mathbf{y}_t, \gamma_{t+1}) - \alpha_t F(\mathbf{x}, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t)\rangle \\
&\quad + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \\
&\overset{(18)}{\leq} \alpha_t F(\mathbf{y}_t, \gamma_{t+1}) - \alpha_t \left[F(\mathbf{y}_t, \gamma_{t+1}) + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x} - \mathbf{y}_t\rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}_t\|^2\right] + \\
&\quad \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t)\rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \\
&= -\alpha_t \left[\langle \nabla F_{t+1}(\mathbf{y}_t, \gamma_{t+1}) - \sigma_{t+1}(\mathbf{y}_t), \mathbf{x} - \mathbf{y}_t\rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}_t\|^2\right] + \\
&\quad \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t)\rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \\
&= -\alpha_t \left[\langle \nabla F_{t+1}(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x} - \mathbf{y}_t\rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}_t\|^2 + \frac{\theta_t}{2}\|\mathbf{x} - \mathbf{v}_t\|^2\right] + \frac{\alpha_t\theta_t}{2}\|\mathbf{x} - \mathbf{v}_t\|^2 + \\
&\quad \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t)\rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 + \langle \sigma_{t+1}(\mathbf{y}_t), \alpha_t(\mathbf{x} - \mathbf{y}_t)\rangle \\
&\leq -\alpha_t \left[\langle \nabla F_{t+1}(\mathbf{y}_t, \gamma_{t+1}), \mathbf{v}_{t+1} - \mathbf{y}_t\rangle + \frac{\mu}{2}\|\mathbf{v}_{t+1} - \mathbf{y}_t\|^2 + \frac{\theta_t}{2}\|\mathbf{v}_{t+1} - \mathbf{v}_t\|^2 + \frac{\mu + \theta_t}{2}\|\mathbf{x} - \mathbf{v}_{t+1}\|^2\right] \\
&\quad + \frac{\alpha_t\theta_t}{2}\|\mathbf{x} - \mathbf{v}_t\|^2 + \langle \nabla F(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x}_{t+1} - \mathbf{y}_t - (1-\alpha_t)(\mathbf{x}_t - \mathbf{y}_t)\rangle + \frac{L_{F_{t+1}}}{2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \\
&\quad + \langle \sigma_{t+1}(\mathbf{y}_t), \alpha_t(\mathbf{x} - \mathbf{y}_t)\rangle,
\end{aligned}
\tag{35}
$$

where the last inequality is due to Lemma 6 (taking $D(\mathbf{u}, \mathbf{v}) = \frac{1}{2}\|\mathbf{u} - \mathbf{v}\|^2$) and the definition of $\mathbf{v}_{t+1}$:

$$
\mathbf{v}_{t+1} \equiv \arg\min_{\mathbf{x}} \langle \nabla F_{t+1}(\mathbf{y}_t, \gamma_{t+1}), \mathbf{x} - \mathbf{y}_t\rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}_t\|^2 + \frac{\theta_t}{2}\|\mathbf{x} - \mathbf{v}_t\|^2. \tag{36}
$$

Minimizing the above directly leads to Line 4 of Alg.1:

$$
\mathbf{v}_{t+1} = \frac{\theta_t\mathbf{v}_t + \mu\mathbf{y}_t - \nabla F_{t+1}(\mathbf{y}_t, \gamma_{t+1})}{\mu + \theta_t}. \tag{37}
$$

Base on this updating rule, it is easy to verify the following inequality:

$$
\begin{aligned}
&-\alpha_t \left[\frac{\mu}{2}\|\mathbf{v}_{t+1} - \mathbf{y}_t\|^2 + \frac{\theta_t}{2}\|\mathbf{v}_{t+1} - \mathbf{v}_t\|^2\right] \\
&\leq -\frac{\alpha_t}{2}\left[\frac{\mu\theta_t}{\mu + \theta_t}\|\mathbf{v}_t - \mathbf{y}_t\|^2 + \frac{1}{\mu + \theta_t}\|\nabla F_{t+1}(\mathbf{y}_t, \gamma_{t+1})\|^2\right] \\
&\leq \frac{-\alpha_t}{2(\mu + \theta_t)}\|\nabla F_{t+1}(\mathbf{y}_t, \gamma_{t+1})\|^2.
\end{aligned}
\tag{38}
$$

To set $\mathbf{x}_{t+1}$ (Line 3 of Alg.1), we follow the classic stochastic gradient descent, such that $\|\mathbf{x}_{t+1}-\mathbf{y}_t\|^2$ can be bounded in terms of $\|\nabla F_{t+1}(\mathbf{y}_t,\gamma_{t+1})\|^2$: $\mathbf{x}_{t+1}=\mathbf{y}_t-\eta_t\nabla F_{t+1}(\mathbf{y}_t,\gamma_{t+1})$.

Hence

$$\|\mathbf{x}_{t+1}-\mathbf{y}_t\|^2=\eta_t^2\|\nabla F_{t+1}(\mathbf{y}_t,\gamma_{t+1})\|^2, \tag{39}$$

and

$$\begin{aligned}&\langle\nabla F(\mathbf{y}_t,\gamma_{t+1}),\mathbf{x}_{t+1}-\mathbf{y}_t\rangle=\langle\nabla F_{t+1}(\mathbf{y}_t,\gamma_{t+1})-\sigma_{t+1}(\mathbf{y}_t),\mathbf{x}_{t+1}-\mathbf{y}_t\rangle\\&\leq-\eta_t\|\nabla F_{t+1}(\mathbf{y}_t,\gamma_{t+1})\|^2+\eta_t\|\sigma_{t+1}(\mathbf{y}_t)\|\cdot\|\nabla F_{t+1}(\mathbf{y}_t,\gamma_{t+1})\|.\end{aligned} \tag{40}$$

Inserting (37,38,39 and 40) into (35) we have

$$\begin{aligned}&\Delta_{t+1}\leq(1-\alpha_t)\Delta_t+(1-\alpha_t)(\gamma_t-\gamma_{t+1})D_{\mathcal{U}}+\\&\frac{\alpha_t}{2}\left[\theta_t\|\mathbf{x}-\mathbf{v}_t\|^2-(\mu+\theta_t)\|\mathbf{x}-\mathbf{v}_{t+1}\|^2\right]+\langle\sigma_{t+1}(\mathbf{y}_t),\alpha_t(\mathbf{x}-\mathbf{y}_t)+(1-\alpha_t)(\mathbf{x}_t-\mathbf{y}_t)\rangle+\\&\eta_t\|\sigma_{t+1}(\mathbf{y}_t)\|\cdot\|\nabla F_{t+1}(\mathbf{y}_t,\gamma_{t+1})\|+\left[\frac{\alpha_t}{2(\mu+\theta_t)}+\frac{L_{t+1}}{2}\eta_t^2-\eta_t\right]\|\nabla F_{t+1}(\mathbf{y}_t,\gamma_{t+1})\|^2+\\&\left\langle\nabla F_{t+1}(\mathbf{y}_t,\gamma_{t+1}),\frac{-\alpha_t\theta_t(\mathbf{v}_t-\mathbf{y}_t)}{\mu+\theta_t}-(1-\alpha_t)(\mathbf{x}_t-\mathbf{y}_t)\right\rangle.\end{aligned} \tag{41}$$

Taking the last term $\frac{-\alpha_t\theta_t(\mathbf{v}_t-\mathbf{y}_t)}{\mu+\theta_t}-(1-\alpha_t)(\mathbf{x}_t-\mathbf{y}_t)=0$ recovers the updating rule of $\mathbf{y}_t$ (Line 1 of Alg.1). Hence our result follows. □

2.3.1 How to Choose Stepsizes η_t

In the RHS of (26), nonnegative scalars $p,q\geq0$ are data-dependent, and could be arbitrarily large. Hence we need to set proper stepsizes η_t such that the last two terms in (26) are non-positive. One might conjecture that: there exist a series $c_t\geq0$ such that

$$\eta_t pq+\left[\frac{\alpha_t}{2(\mu+\theta_t)}+\frac{L_{t+1}}{2}\eta_t^2-\eta_t\right]q^2\leq c_t p^2. \tag{42}$$

It is easy to verify that if we take $\eta_t=\frac{\alpha_t}{\mu+\theta_t}$ and any series $c_t\geq\frac{\alpha_t}{2(\mu+\theta_t-\alpha_t L_{t+1})}\geq0$, then (42) is satisfied. To retain a tight bound, we take

$$c_t=\frac{\alpha_t}{2(\mu+\theta_t-\alpha_t L_{t+1})}. \tag{43}$$

Taking expectation on both sides of (26) and noticing that $\mathbb{E}_{\boldsymbol{\xi_{t+1}}|\boldsymbol{\xi}_{[t]}}\Gamma_{t+1}=0$, $\mathbb{E}_{\boldsymbol{\xi}_{t+1}}c_t\leq\frac{\alpha_t}{2(\mu+\theta_t-\alpha_t\mathbb{E}_{\boldsymbol{\xi}_{t+1}}L_{t+1})}$ due to Jensen's inequality, we have

Lemma 7. $\forall \mathbf{x}$ *and* $\forall t \geq 0$,

$$\begin{aligned}\mathbb{E}\Delta_{t+1} \leq (1-\alpha_t)\mathbb{E}\Delta_t + \alpha_t\theta_t D_t^2 - \alpha_t(\mu+\theta_t)D_{t+1}^2 \\ + \frac{\alpha_t}{2(\mu+\theta_t-\alpha_t\mathbb{E}L_{t+1})}\sigma^2 + (1-\alpha_t)(\gamma_t-\gamma_{t+1})D_{\mathcal{U}},\end{aligned} \tag{44}$$

The optimal convergence rates of our algorithm differs according to the fact of μ (positive or not). They are presented separately in the following two subsections, where the choices of $\gamma_t,\ \theta_t,\ \alpha_t$ will also be determined.

2.3.2 Optimal Rates for Composite Minimizations when $\mu = 0$

When $\mu = 0$, $g()$ is only convex and L_g-Lipschitz smooth, but not assumed to be strongly convex.

Theorem 5. *Take* $\alpha_t = \frac{2}{t+2}$, $\gamma_{t+1} = \alpha_t$, $\theta_t = L_g\alpha_t + \frac{\Omega}{\sqrt{\alpha_t}} + \frac{\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta}$ *and* $\eta_t = \frac{\alpha_t}{\theta_t}$ *in Alg.1, where* Ω *is a constant. We have* $\forall \mathbf{x}$ *and* $\forall t \geq 0$,

$$\mathbb{E}\left[\Phi(\mathbf{x}_{t+1}) - \Phi(\mathbf{x})\right] \leq \frac{4L_g D^2}{(t+2)^2} + \frac{2\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2 D^2/\zeta + 4D_{\mathcal{U}}}{t+2} + \frac{\sqrt{2}(\Omega D^2 + \sigma^2/\Omega)}{\sqrt{t+2}}, \tag{45}$$

where $D^2 \equiv \max_i D_i^2$.

Proof. It is easy to verify that by taking $\alpha_t = \frac{2}{t+2}$, $\gamma_{t+1} = \alpha_t$ and $\theta_t = L_g\alpha_t + \frac{\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta} + \frac{\Omega}{\sqrt{\alpha_t}}$, we have $\forall t > 1$:

$$(1-\alpha_{t-1})(\gamma_{t-1}-\gamma_t) \leq \gamma_t - \gamma_{t+1}, \tag{46}$$

and

$$(1-\alpha_t)\frac{\alpha_{t-1}}{2(\theta_{t-1}-\alpha_{t-1}\mathbb{E}L_t)} \leq \frac{\alpha_t}{2(\theta_t-\alpha_t\mathbb{E}L_{t+1})}. \tag{47}$$

Next we define and bound weighted sums of D_t^2 that will be used later.

$$\begin{aligned}\Psi(t) := \left[\alpha_t\theta_t - (1-\alpha_t)\alpha_{t-1}\theta_{t-1}\right]D_t^2 + (1-\alpha_t)\left[\alpha_{t-1}\theta_{t-1} - (1-\alpha_{t-1})\alpha_{t-2}\theta_{t-2}\right]D_{t-1}^2 + \\ (1-\alpha_t)(1-\alpha_{t-1})\left[\alpha_{t-2}\theta_{t-2} - (1-\alpha_{t-2})\alpha_{t-3}\theta_{t-3}\right]D_{t-2}^2 + \cdots,\end{aligned} \tag{48}$$

where replacing α_t and θ_t by their definitions we have $\forall t$:

$$\alpha_t\theta_t - (1-\alpha_t)\alpha_{t-1}\theta_{t-1} = \frac{4L_g}{(t+1)^2(t+2)^2} + \frac{2\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2/\zeta}{(t+1)(t+2)} + \frac{\sqrt{2}\left[(t+1)\sqrt{t+2} - t\sqrt{t+1}\right]\Omega}{(t+1)(t+2)} \tag{49}$$

Substituting (49) into (48) and using invoking the definition of D^2 we have $\forall t$:

$$
\begin{aligned}
\Psi(t) &\leq 4L_g D^2 \left[\frac{1}{(t+1)^2(t+2)^2} + \frac{t(t+1)}{(t+1)(t+2)}\frac{1}{t^2(t+1)^2} + \frac{(t-1)t}{(t+1)(t+2)}\frac{1}{(t-1)^2t^2} + \cdots\right] \\
&+ \frac{2\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2 D^2}{\zeta}\left[\frac{1}{(t+1)(t+2)} + \frac{t(t+1)}{(t+1)(t+2)}\frac{1}{t(t+1)} + \frac{(t-1)t}{(t+1)(t+2)}\frac{1}{(t-1)t} + \cdots\right] \\
&+ \sqrt{2}\Omega D^2\Bigg[\frac{(t+1)\sqrt{t+2} - t\sqrt{t+1}}{(t+1)(t+2)} + \frac{t(t+1)}{(t+1)(t+2)}\frac{t\sqrt{t+1} - (t-1)\sqrt{t}}{t(t+1)} + \\
&\quad \frac{(t-1)t}{(t+1)(t+2)}\frac{(t-1)\sqrt{t} - (t-2)\sqrt{t-1}}{(t-1)t} + \cdots\Bigg] \\
&= \frac{4L_g D^2}{(t+1)(t+2)}\left[\left(\frac{1}{t+1} - \frac{1}{t+2}\right) + \left(\frac{1}{t} - \frac{1}{t+1}\right) + \left(\frac{1}{t-1} - \frac{1}{t}\right) + \cdots\right] \\
&+ \frac{2\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2 D^2}{\zeta}\left[\frac{1}{(t+1)(t+2)} + \frac{1}{(t+1)(t+2)} + \frac{1}{(t+1)(t+2)} + \cdots\right] \\
&+ \frac{\sqrt{2}\Omega D^2}{(t+1)(t+2)}\left[(t+1)\sqrt{t+2} - t\sqrt{t+1} + t\sqrt{t+1} - (t-1)\sqrt{t} + (t-1)\sqrt{t} - (t-2)\sqrt{t-1} + \cdots\right] \\
&\leq \alpha_t\theta_t D^2.
\end{aligned}
\tag{50}
$$

Since $\mu = 0$, by recursively applying (44) and $1 - \alpha_0 = 0$ we have

$$
\begin{aligned}
\mathbb{E}\Delta_{t+1} &\leq (1-\alpha_t)\mathbb{E}\Delta_t + \alpha_t\theta_t\left(D_t^2 - D_{t+1}^2\right) + \frac{\alpha_t}{2(\theta_t - \alpha_t\mathbb{E}L_{t+1})}\sigma^2 + (1-\alpha_t)(\gamma_t - \gamma_{t+1})D_{\mathcal{U}} \\
&\leq (1-\alpha_t)(1-\alpha_{t-1})\mathbb{E}\Delta_{t-1} + \alpha_t\theta_t\left(D_t^2 - D_{t+1}^2\right) + (1-\alpha_t)\,\alpha_{t-1}\theta_{t-1}\left(D_{t-1}^2 - D_t^2\right) + \\
&\quad \frac{2\alpha_t}{2(\theta_t - \alpha_t\mathbb{E}L_{t+1})}\sigma^2 + 2(1-\alpha_t)(\gamma_t - \gamma_{t+1})D_{\mathcal{U}} \\
&\leq \cdots \\
&\overset{(48)}{\leq} \prod_{i=0}^{t}(1-\alpha_i)\Delta_0 + \Psi(t) + \frac{(t+1)\alpha_t}{2(\theta_t - \alpha_t\mathbb{E}L_{t+1})}\sigma^2 + (t+1)(1-\alpha_t)(\gamma_t - \gamma_{t+1})D_{\mathcal{U}} \\
&\overset{(50)}{\leq} \alpha_t\theta_t D^2 + \frac{\sigma^2}{\theta_t - \alpha_t\mathbb{E}L_{t+1}} + \frac{2D_{\mathcal{U}}}{t+2} \\
&= \left[\alpha_t^2\mathbb{E}L_{t+1} + \Omega\sqrt{\alpha_t}\right]D^2 + \frac{\sqrt{\alpha_t}\sigma^2}{\Omega} + \frac{2D_{\mathcal{U}}}{t+2}.
\end{aligned}
\tag{51}
$$

Combining with Lemma 3 we have $\forall \mathbf{x}$

$$
\begin{aligned}
\mathbb{E}\left[\Phi(\mathbf{x}_{t+1}) - \Phi(\mathbf{x})\right] &\leq \left[\alpha_t^2\mathbb{E}L_{t+1} + \Omega\sqrt{\alpha_t}\right]D^2 + \frac{\sqrt{\alpha_t}\sigma^2}{\Omega} + \frac{2D_{\mathcal{U}}}{t+2} + \gamma_{t+1}D_{\mathcal{U}} \\
&\leq \alpha_t^2 L_g D^2 + \left(\gamma_{t+1} + \frac{2}{t+2}\right)D_{\mathcal{U}} + \alpha_t^2\frac{\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\gamma_{t+1}\zeta}D^2 + \sqrt{\alpha_t}\left(\Omega D^2 + \frac{\sigma^2}{\Omega}\right).
\end{aligned}
\tag{52}
$$

Taking $\gamma_{t+1} = \alpha_t = \frac{2}{t+2}$ our result follows. □

In this result, the variance bound is optimal up to a constant factor (Agarwal et al. [2012]). The dominating factor is still due to the stochasticity, but not affected by the nonsmoothness of $f()$. Taking the parameter $\Omega = \sigma/D$, this last term becomes $\frac{2\sqrt{2}D\sigma}{\sqrt{t+2}}$. This bound is better than that of stochastic gradient descent or stochastic dual averaging (Dekel et al. [2010]) for minimizing L-Lipschitz smooth functions, whose rate is $O\left(\frac{LD_0^2}{t} + \frac{D_0^2+\sigma^2}{\sqrt{t}}\right)$; without the smooth function $g()$, our bound is of the same order as it, keeping in mind that our rate is for nonsmooth minimizations. This fact underscores the potential of using stochastic optimal methods for nonsmooth functions. In a time budget t, the optimization errors for both smooth function g and nonsmooth function f are optimal up to constant factors, according to Definition 2, Algorithm 1 is *computationally optimal*.

The diminishing smoothness parameter $\gamma_t = \frac{2}{t+2}$ indicates that initially a smoother approximation is preferred, such that the solution does not change wildly due to the nonsmoothness and stochasticity. Eventually the approximated function should be closer and closer to the original nonsmooth function, such that the optimality can be reached. Some concrete examples are given in Fig.8.

The $\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2$ in our bound is a theoretical constant. In Sec.2.4 we demonstrate a sampling method, and it turns out to work quite well in estimating $\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2$.

2.3.3 Nearly Optimal Rates for Strongly Convex Minimizations

When $\mu > 0$, $g()$ is strongly convex, and the convergence rate of ANSGD can be improved to $O(1/t)$.

Theorem 6. *Take* $\alpha_t = \frac{2}{t+1}$, $\gamma_{t+1} = \alpha_t$, $\theta_t = L_g\alpha_t + \frac{\mu}{2\alpha_t} + \frac{\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta} - \mu$ *and* $\eta_t = \frac{\alpha_t}{\mu+\theta_t}$ *in Alg.1. Denote*

$$C \equiv \max\left\{\frac{4\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta\mu}, 2\left(\frac{L_g}{\mu}\right)^{1/3}\right\}. \tag{53}$$

We have $\forall \mathbf{x}$ *and* $\forall t \geq 0$,

$$\mathbb{E}\left[\Phi(\mathbf{x}_{t+1}) - \Phi(\mathbf{x})\right] \leq \frac{6.58L_g\tilde{D}^2}{t(t+1)} + \mathcal{B} + \frac{4D_{\mathcal{U}}}{t+1} + \frac{\sigma^2}{\mu(t+1)}, \tag{54}$$

where

$$\mathcal{B} \equiv \begin{cases} \frac{2\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2 \tilde{D}^2/\zeta}{t+1} & \text{if } 0 \le t < C, \\ \frac{2(C-2)\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2 \tilde{D}^2/\zeta}{t(t+1)} & \text{if } t \ge C, \end{cases} \tag{55}$$

and $\tilde{D}^2 \equiv \max_{0 \le i \le \min\{t,C\}} D_i^2$.

Proof. It is easy to verify that by taking $\alpha_t = \frac{2}{t+1}$, we have $\forall t \ge 1$

$$(1-\alpha_{t-1})(\gamma_{t-1}-\gamma_t) \le \gamma_t - \gamma_{t+1}. \tag{56}$$

and

$$(1-\alpha_t)\alpha_{t-1}^2 \le \alpha_t^2 \tag{57}$$

Denote

$$S_t := \alpha_t\theta_t - (1-\alpha_t)(\alpha_{t-1}\theta_{t-1} + \mu\alpha_{t-1}). \tag{58}$$

Taking $\theta_t = L_g\alpha_t + \frac{\mu}{2\alpha_t} + \frac{\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta} - \mu$ it is easy to verify that $\forall t \ge 1$:

$$S_t = 4L_g\frac{1}{(t+1)^2t^2} + \frac{2\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta}\left[\frac{1}{t} - \frac{1}{t+1}\right] - \frac{\mu}{t+1}. \tag{59}$$

We want to find the smallest iteration index C such that: when $t \ge C$, $S_t \le 0$. Without any knowledge about L_g and $\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2$, minimizing S_t w.r.t t does not yield an analytic form of C. Hence we simply let

$$4L_g\frac{1}{(t+1)^2t^2} \le \frac{\mu}{2(t+1)}, \tag{60}$$

and

$$\frac{2\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta}\left[\frac{1}{t} - \frac{1}{t+1}\right] \le \frac{\mu}{2(t+1)}. \tag{61}$$

Inequality (60) is satisfied when

$$t \ge 2\left(\frac{L_g}{\mu}\right)^{1/3}, \tag{62}$$

and (61) is satisfied when

$$t \ge \frac{4\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta\mu}. \tag{63}$$

Combining these two we reach the definition of C in (53). Next we proceed to prove the bound.

As defined in the theorem, we denote $\tilde{D}^2 = \max_{0\le i\le \min(t,C)} D_i^2$. By recursively applying (44) for $0 \le i \le t$ and noticing that $S_t \le 0\ \forall t \ge C$, $1-\alpha_1 = 0$ we have

$$
\begin{aligned}
\mathbb{E}\Delta_{t+1} &\overset{(56)}{\le} \prod_{i=0}^{t}(1-\alpha_i)\Delta_0 + (t+1)(1-\alpha_t)(\gamma_t-\gamma_{t+1})D_{\mathcal{U}} + \\
&\quad \left[(\alpha_t\theta_t)D_t^2 - (\alpha_t\theta_t+\mu\alpha_t)D_{t+1}^2\right] + \\
&\quad (1-\alpha_t)\left[(\alpha_{t-1}\theta_{t-1})D_{t-1}^2 - (\alpha_{t-1}\theta_{t-1}+\mu\alpha_{t-1})D_t^2\right] + \\
&\quad (1-\alpha_t)(1-\alpha_{t-1})\left[(\alpha_{t-2}\theta_{t-2})D_{t-2}^2 - (\alpha_{t-2}\theta_{t-2}+\mu\alpha_{t-2})D_{t-1}^2\right] + \\
&\quad \cdots + \prod_{i=1}^{t}(1-\alpha_i)\left[(\alpha_0\theta_0)D_0^2 - (\alpha_0\theta_0+\mu\alpha_0)D_1^2\right] + \\
&\quad \frac{\sigma^2}{\mu}\left[\alpha_t^2 + (1-\alpha_t)\alpha_{t-1}^2 + \cdots + \prod_{i=1}^{t}(1-\alpha_i)\alpha_0^2\right] \\
&\overset{(57)}{\le} \frac{2D_{\mathcal{U}}}{t+1} + \tilde{D}^2\prod_{i=C-1}^{t}(1-\alpha_i)\left[\alpha_{C-2}\theta_{C-2} - (1-\alpha_{C-2})(\alpha_{C-3}\theta_{C-3}+\mu\alpha_{C-3})\right] + \\
&\quad \tilde{D}^2\prod_{i=C-2}^{t}(1-\alpha_i)\left[\alpha_{C-3}\theta_{C-3} - (1-\alpha_{C-3})(\alpha_{C-4}\theta_{C-4}+\mu\alpha_{C-4})\right] + \\
&\quad \cdots + \tilde{D}^2\prod_{i=2}^{t}(1-\alpha_i)\left[\alpha_1\theta_1 - (1-\alpha_1)(\alpha_0\theta_0+\mu\alpha_0)\right] + \frac{t\alpha_t^2\sigma^2}{\mu}
\end{aligned}
\tag{64}
$$

Applying (59) by ignoring the $-\frac{\mu}{t+1}$ term to the above inequality we can bound the coefficients of L_g and $\frac{\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta}$ parts separately as follows.

When $t \ge C$, for the L_g part:

$$
\begin{aligned}
&\frac{\prod_{i=C-1}^{t}(1-\alpha_i)}{(C-1)^2(C-2)^2} + \frac{\prod_{i=C-2}^{t}(1-\alpha_i)}{(C-2)^2(C-3)^2} + \frac{\prod_{i=C-3}^{t}(1-\alpha_i)}{(C-3)^2(C-4)^2} + \cdots + \frac{\prod_{i=2}^{t}(1-\alpha_i)}{2^2\cdot 1^2} \\
&= \frac{1}{(t+1)t}\left[\frac{1}{(C+2)(C+1)} + \frac{1}{(C+1)C} + \frac{1}{C(C-1)} + \cdots + \frac{1}{2\cdot 1}\right] \\
&\le \frac{1}{(t+1)t}\sum_{i=1}^{C+1}\frac{1}{i^2} \le \frac{\pi^2}{6t(t+1)}
\end{aligned}
\tag{65}
$$

For the $\frac{\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta}$ part:

$$\begin{aligned}
&\prod_{i=C-1}^{t}(1-\alpha_i)\left(\frac{1}{C-2}-\frac{1}{C-1}\right)+\prod_{i=C-2}^{t}(1-\alpha_i)\left(\frac{1}{C-3}-\frac{1}{C-2}\right)+\\
&\cdots+\prod_{i=2}^{t}(1-\alpha_i)\left(1-\frac{1}{2}\right)\\
&=\frac{C-1}{(t+1)t}-\frac{C-2}{(t+1)t}+\frac{C-2}{(t+1)t}-\frac{C-3}{(t+1)t}+\cdots+\frac{2}{(t+1)t}-\frac{1}{(t+1)t}\\
&=\frac{C-1}{(t+1)t}-\frac{1}{(t+1)t}=\frac{C-2}{t(t+1)}.
\end{aligned} \tag{66}$$

Combining with Lemma 3 and taking $\gamma_{t+1}=\alpha_t=\frac{2}{t+1}$ we have $\forall \mathbf{x}$:

$$\begin{aligned}
\mathbb{E}\left[\Phi(\mathbf{x}_{t+1})-\Phi(\mathbf{x})\right] &\leq \frac{2D_{\mathcal{U}}}{t+1}+\frac{2\pi^2L_g\tilde{D}^2}{3t(t+1)}+\frac{2(C-2)\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2\tilde{D}^2/\zeta}{t(t+1)}+\frac{\sigma^2}{\mu(t+1)}+\gamma_{t+1}D_{\mathcal{U}}\\
&=\frac{2\pi^2L_g\tilde{D}^2}{3t(t+1)}+\frac{2(C-2)\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2\tilde{D}^2/\zeta}{t(t+1)}+\frac{4D_{\mathcal{U}}}{t+1}+\frac{\sigma^2}{\mu(t+1)}.
\end{aligned} \tag{67}$$

When $0\leq t\leq C$, one can simply put $C=t$ in the above, and this completes our proof. □

Note that C is the smallest iteration index for which one can retain $1/t^2$ rates for the $\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2$ part ($\mathcal{B}$). Without any knowledge about L_g, μ and $\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2$, one can set a parameter Ω and take $\theta_t=L_g\alpha_t+\frac{\mu}{2\alpha_t}+\frac{\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\Omega\zeta}-\mu$ in the algorithm. In our experiments, we observe that one can take Ω fairly large (of $O(\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2)$), meaning that C can be very small (O(1)), and $\mathcal{B}$ is $O(\frac{1}{t^2})$ for *all* t. In this sense, strongly convex ANSGD is almost parameter-free. Without the $O(1/t)$ rate of $D_{\mathcal{U}}$, all terms in our bound are optimal. This is why our rate is called "nearly" optimal. In practice, $D_{\mathcal{U}}$ is usually small, and it will be dominated by the last term $\frac{\sigma^2}{\mu(t+1)}$.

2.3.4 Batch-to-Online Conversion

The performance of an online learning (online convex minimization) algorithm is typically measured by *regret*, which can be expressed as

$$R(t)\equiv\sum_{i=0}^{t-1}\left[\Phi(\mathbf{x}_i,\boldsymbol{\xi}_{i+1})-\Phi(\mathbf{x}_t^*,\boldsymbol{\xi}_{i+1})\right], \tag{68}$$

where $\mathbf{x}_t^*\equiv\arg\min_{\mathbf{x}}\sum_{i=0}^{t-1}\left[\Phi(\mathbf{x},\boldsymbol{\xi}_{i+1})\right]$. In the learning theory literature, many approaches are proposed which use online learning algorithms for batch learning (stochastic optimization), called "online-to-batch" (O-to-B) conversions. For convex functions, many of these

approaches employ an "averaged" solution as the final solution.

On the contrary, we show that stochastic optimization algorithms can also be used *directly* for online learning. This "batch-to-online" (B-to-O) conversion is almost free of any additional effort: under i.i.d. assumptions of data, one can use any stochastic optimization algorithm for online learning.

Proposition 1. *For any* $t \geq 0$*,*

$$\mathbb{E}_{\boldsymbol{\xi}_{[t]}} R(t) \leq \sum_{i=0}^{t-1} \mathbb{E}_{\boldsymbol{\xi}_{[i]}} \left[\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}^*)\right] + \mathbb{E}_{\boldsymbol{\xi}_{[t]}} \sum_{i=0}^{t-1} \left[\Phi(\mathbf{x}_t^*) - \Phi(\mathbf{x}_t^*, \boldsymbol{\xi}_{i+1})\right] \tag{69}$$

where $\mathbf{x}^* \equiv \arg\min_{\mathbf{x}} \Phi(\mathbf{x})$ *and* $\mathbf{x}_t^* \equiv \arg\min_{\mathbf{x}} \sum_{i=0}^{t-1} \left[\Phi(\mathbf{x}, \boldsymbol{\xi}_{i+1})\right]$.

Proof.

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\xi}_{[t]}} R(t) = \mathbb{E}_{\boldsymbol{\xi}_{[t]}} \sum_{i=0}^{t-1} \left[\Phi(\mathbf{x}_i, \boldsymbol{\xi}_{i+1}) - \Phi(\mathbf{x}_t^*, \boldsymbol{\xi}_{i+1})\right] \\
&= \mathbb{E}_{\boldsymbol{\xi}_{[t]}} \sum_{i=0}^{t-1} \Big\{ \left[\Phi(\mathbf{x}_i, \boldsymbol{\xi}_{i+1}) - \Phi(\mathbf{x}^*)\right] + \left[\Phi(\mathbf{x}^*) - \Phi(\mathbf{x}_t^*, \boldsymbol{\xi}_{i+1})\right] \Big\} \\
&= \sum_{i=0}^{t-1} \mathbb{E}_{\boldsymbol{\xi}_{[i+1]}} \left[\Phi(\mathbf{x}_i, \boldsymbol{\xi}_{i+1}) - \Phi(\mathbf{x}^*)\right] + \mathbb{E}_{\boldsymbol{\xi}_{[t]}} \sum_{i=0}^{t-1} \left[\Phi(\mathbf{x}^*) - \Phi(\mathbf{x}_t^*)\right] \\
&\quad + \mathbb{E}_{\boldsymbol{\xi}_{[t]}} \sum_{i=0}^{t-1} \left[\Phi(\mathbf{x}_t^*) - \Phi(\mathbf{x}_t^*, \boldsymbol{\xi}_{i+1})\right] \\
&\leq \sum_{i=0}^{t-1} \mathbb{E}_{\boldsymbol{\xi}_{[i+1]}} \left[\Phi(\mathbf{x}_i, \boldsymbol{\xi}_{i+1}) - \Phi(\mathbf{x}^*)\right] + \mathbb{E}_{\boldsymbol{\xi}_{[t]}} \sum_{i=0}^{t-1} \left[\Phi(\mathbf{x}_t^*) - \Phi(\mathbf{x}_t^*, \boldsymbol{\xi}_{i+1})\right] \\
&= \sum_{i=0}^{t-1} \mathbb{E}_{\boldsymbol{\xi}_{[i]}} \left[\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}^*)\right] + \mathbb{E}_{\boldsymbol{\xi}_{[t]}} \sum_{i=0}^{t-1} \left[\Phi(\mathbf{x}_t^*) - \Phi(\mathbf{x}_t^*, \boldsymbol{\xi}_{i+1})\right].
\end{aligned}$$

□

When $\Phi()$ is convex, the second term in (69) can be bounded by applying standard results in uniform convergence (e.g. Boucheron et al. [2005]): $\sum_{i=1}^{t-1} \Phi(\mathbf{x}_t^*) - \Phi(\mathbf{x}_t^*, \boldsymbol{\xi}_{i+1}) = O(\sqrt{t})$. Together with summing up the RHS of (45), we can obtain an $O(\sqrt{t})$ regret bound. When $\Phi()$ is strongly convex, the second term in (69) can be bounded using (Shalev-Shwartz et al. [2009]): $\sum_{i=1}^{t-1} \Phi(\mathbf{x}_t^*) - \Phi(\mathbf{x}_t^*, \boldsymbol{\xi}_{i+1}) = O(\ln t)$. Together with summing up the RHS of (54), an $O(\ln t)$ regret bound is achieved. The $O(\sqrt{t})$ and $O(\ln t)$ regret bounds are known

Using our proposed ANSGD for online learning by B-to-O achieves the same (optimal) regret bounds as state-of-the-art algorithms designated for online learning. However, using

O-to-B, one can only retain an $O(\ln t/t)$ rate of convergence for stochastic strongly convex optimization. From this perspective, O-to-B is inferior to B-to-O. The sub-optimality of O-to-B is also discussed in Hazan and Kale [2011].

2.4 Examples

In this section, two nonsmooth functions are given as examples. We will show how these functions can be stochastically approximated, and how to calculate parameters used in our algorithm.

2.4.1 Hinge Loss SVM Classification

Hinge loss is a convex surrogate of the $0-1$ loss. Denote a sample-label pair as $\boldsymbol{\xi} \equiv \{\mathbf{s}, l\} \sim P$, where $\mathbf{s} \in \mathbb{R}^D$ and $l \in \mathbb{R}$. Hinge loss can be expressed as $f_{\text{hinge}}(\mathbf{x}) \equiv \max\{0, 1 - l\mathbf{s}^T\mathbf{x}\}$. It has been widely used for SVM classifiers where the objective is $\min \Phi(\mathbf{x}) = \min \mathbb{E}_{\boldsymbol{\xi}} f_{\text{hinge}}(\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{x}\|^2$. Note that the regularization term $g(\mathbf{x}) = \frac{\lambda}{2}\|\mathbf{x}\|^2$ is λ-strongly convex, hence according to Thm.6, ANSGD enjoys $O(1/(\lambda t))$ rates. Taking $\omega(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|^2$ in (23), it is easy to check that the smooth stochastic approximation of hinge loss is

$$\hat{f}_{\text{hinge}}(\mathbf{x}, \boldsymbol{\xi}_t, \gamma_t) = \max_{0 \le u \le 1} \left\{ u\left(1 - l_t \mathbf{s}_t^T \mathbf{x}\right) - \gamma_t \frac{u^2}{2} \right\}. \tag{70}$$

This maximization is simple enough such that we can obtain an equivalent smooth representation:

$$\hat{f}_{\text{hinge}}(\mathbf{x}, \boldsymbol{\xi}_t, \gamma_t) = \begin{cases} 0 & \text{if } l_t \mathbf{s}_t^T \mathbf{x} \ge 1, \\ \frac{(1 - l_t \mathbf{s}_t^T \mathbf{x})^2}{2\gamma_t} & \text{if } 1 - \gamma_t \le l_t \mathbf{s}_t^T \mathbf{x} < 1, \\ 1 - l_t \mathbf{s}_t^T \mathbf{x} - \frac{\gamma_t}{2} & \text{if } l_t \mathbf{s}_t^T \mathbf{x} < 1 - \gamma_t. \end{cases} \tag{71}$$

Several examples of $\hat{f}_{\text{hinge}}$ with varying γ_t are plotted in Fig.8(left) in comparing with the hinge loss.

Here $\mathbf{u}$ is a scalar, hence it is straightforward to calculate $\frac{\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2}{\zeta}$, which will be used to generate sequences θ_t. In binary classification, suppose $l \in \{1, -1\}$. Using definition (21), one only needs to calculate $\mathbb{E}(\max_{\|\mathbf{x}\|=1} \mathbf{s}_t^T \mathbf{x})^2$. Practically one can take a small subset of k random samples $\mathbf{s}_i$ (e.g. $k = 100$), and calculate the sample average of the squared norms $\frac{1}{k}\sum_{i=1}^k \|\mathbf{s}_i\|^2$. This yields $\frac{1}{k}\sum_{i=1}^k (\max_{\|\mathbf{x}\|=1} \mathbf{s}_i^T \mathbf{x})^2$, an estimate of $\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2$.

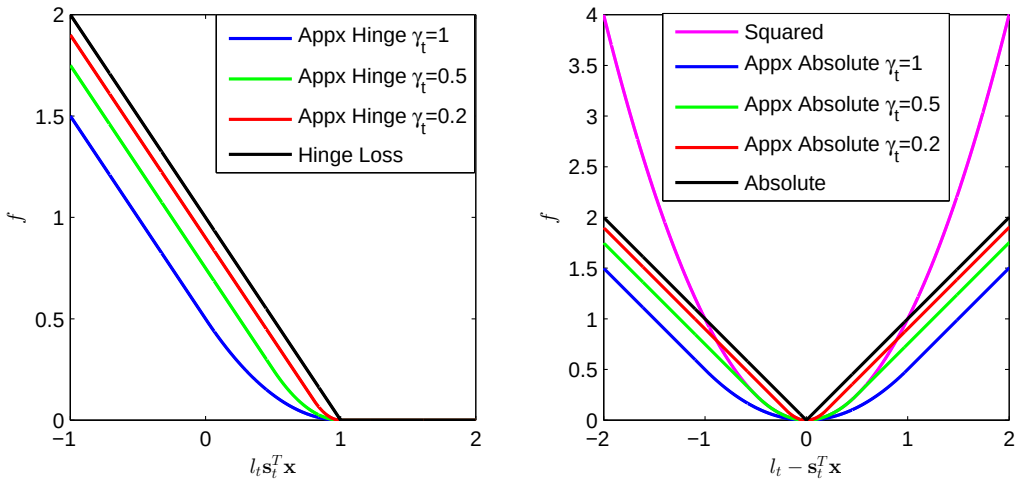

Figure 8: Left: Hinge loss and its smooth approximations. Right: Absolute loss and its smooth approximations.

2.4.2 Absolute Loss Robust Regression

Absolute loss is an alternative to the popular squared loss for robust regressions (Hastie et al. [2009]). Using same notations as Sec.2.4.1 it can be expressed as $f_{abs}(\mathbf{x}) \equiv |l - \mathbf{s}^T\mathbf{x}|$. Taking $\omega(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|^2$ in (23), its smooth stochastic approximation can be expressed as

$$\hat{f}_{\text{abs}}(\mathbf{x}, \boldsymbol{\xi}_t, \gamma_t) = \max_{-1 \le u \le 1} \left\{ u(l_t - \mathbf{s}_t^T\mathbf{x}) - \gamma_t \frac{u^2}{2} \right\}. \tag{72}$$

Solving this maximization wrt u we obtain an equivalent form:

$$\hat{f}_{\text{abs}}(\mathbf{x}, \boldsymbol{\xi}_t, \gamma_t) = \begin{cases} l_t - \mathbf{s}_t^T\mathbf{x} - \frac{\gamma_t}{2} & \text{if } l_t - \mathbf{s}_t^T\mathbf{x} \ge \gamma_t, \\ \frac{(l_t - \mathbf{s}_t^T\mathbf{x})^2}{2\gamma_t} & \text{if } -\gamma_t \le l_t - \mathbf{s}_t^T\mathbf{x} < \gamma_t, \\ -(l_t - \mathbf{s}_t^T\mathbf{x}) - \frac{\gamma_t}{2} & \text{if } l_t - \mathbf{s}_t^T\mathbf{x} < -\gamma_t. \end{cases} \tag{73}$$

This approximation looks similar to the well-studied Huber loss (Huber [1964]), though they are different. Actually they share the same form only when $\gamma_t = 0.5$ (green curve in Fig.8 Right).

The parameter $\mathbb{E}\|A_{\boldsymbol{\xi}}\|^2$ can be estimated in a similar way as discussed in Sec.2.4.1.

2.5 Experimental Results

In this section, five publicly available datasets from various application domains will be used to evaluate the efficiency of ANSGD. Datasets "svmguide1", "real-sim", "rcv1" and "alpha" are for binary classifications, and "abalone" is for robust regressions.[1]

Following our examples in Sec.2.4, we will evaluate our algorithm using approximated hinge loss for classifications, and approximated absolute loss for regressions. Exact hinge and absolute losses will be used for subgradient descent algorithms that we will compare with, as described in the following section. All losses are squared-$l2$-norm-regularized. The regularization parameter λ is shown on each figure. When assuming strong-convexity, we take $\mu = \lambda$.

2.5.1 Algorithms for Comparison and Parameters

We compare ANSGD with three state-of-the-art algorithms. Each algorithm has a data-dependent tuning parameter, denoted by Ω (although they have different physical meanings). The best values of Ω are found based on a tuning subset of samples. Note that when assuming strong-convexity, our ANSGD is almost parameter-free. As discussed after Thm.6, our experiments indicate that the optimal Ω is taken such that $\frac{\mathbb{E}\|A\xi\|^2}{\Omega\zeta} \approx 1$, meaning that one can simply take $\theta_t = L_g\alpha_t + \frac{\mu}{2\alpha_t} + 1 - \mu$.

SGD. The classic stochastic approximation (Robbins and Monro [1951]) is adopted: $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta_t f'(\mathbf{x}_t)$, where $f'(\mathbf{x}_t)$ is the subgradient. When only assuming convexity ($\mu = 0$), we use stepsize $\eta_t = \frac{\Omega}{\sqrt{t}}$. When assuming strong-convexity, we follow the stepsize used in SGD2 (Bottou): $\eta_t = \frac{1}{\mu(t+\Omega)}$.

Averaged SGD. This is algorithmically the same as SGD, except that the averaged result $\bar{\mathbf{x}} \equiv \frac{1}{t}\sum_{i=1}^{t}\mathbf{x}_i$ is used for testing. We follow the stepsizes suggested by the recent work on the non-asymptotic analysis of SGD (Bach and Moulines [2011], Xu [2011]), where it is argued that Polyak's averaging combining with proper stepsizes yield optimal rates. When

[1] Dataset "alpha" is obtained from `ftp://largescale.ml.tu-berlin.de/largescale/`, and the other four datasets can be accessed via `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools`. Dataset "rcv1" comes with $20,242$ training samples and $677,399$ testing samples. For "svmguide1" and "real-sim", we randomly take 60% of the samples for training and 40% for testing. For "alpha" and "abalone", 80% are used for training, and the rest 20% are used for testing.

only assuming convexity, we use stepsizes $\eta_t = \frac{\Omega}{\sqrt{t}}$ (Bach and Moulines [2011]). When assuming strong convexity, the stepsize is taken as $\eta_t = \frac{1}{\Omega(1+\mu t/\Omega)^{3/4}}$ (Xu [2011]).

AC-SA. This approach (Lan [2010], Lan and Ghadimi [2011]) is interesting to compare because like ANSGD, it is another way of obtaining a stochastic algorithm based on Nesterov's optimal method, begging the question of whether it has similar behavior. Theoretically, according to Prop.8 and 9 in Lan and Ghadimi [2011], the bound for the nonsmooth part is of $O(1/\sqrt{t})$ for $\mu = 0$ and $O(1/t)$ for $\mu > 0$. In comparison, our nonsmooth part converges in $O(1/t)$ for $\mu = 0$ and $O(1/t^2)$ for $\mu > 0$. Numerically we observe that directly applying AC-SA to nonsmooth functions results in inferior performances.

2.5.2 Results

Due to the stochasticity of all the algorithms, for each setting of the experiments, we run the program for 10 times, and plot the mean and standard deviation of the results using error bars.

In the first set of experiments, we compare ANSGD with two subgradient-based algorithms SGD and Averaged SGD. Classification results are shown in Fig.9, 10, 11 and 12, and regression results are shown in Fig.13. In each figure, the left column is for algorithms without strongly convex assumptions, while in the right column the algorithms assume strong-convexity and take $\mu = \lambda$. For classification results, we plot function values over the testing set in the first row, and plot testing accuracies in the second row.

It is clear that in all these experiments, ANSGD's function values converges consistently faster than the other two SGD algorithms. In non-strongly convex experiments, it converges significantly faster than SGD and its averaged version. In strongly convex experiments, it still out performs, and is more robust than strongly convex SGD. Averaged SGD performs well in strongly convex settings, in terms of prediction accuracies, although its errors are still higher than ANSGD in the first three datasets. The only exception is in "alpha" (Fig.12), where Averaged SGD retains higher function values than ANSGD, but its accuracies are contradictorily higher in early stages. The reason might be that the inexact solution serves as an additional regularization factor, which cannot be predicted by the analysis of

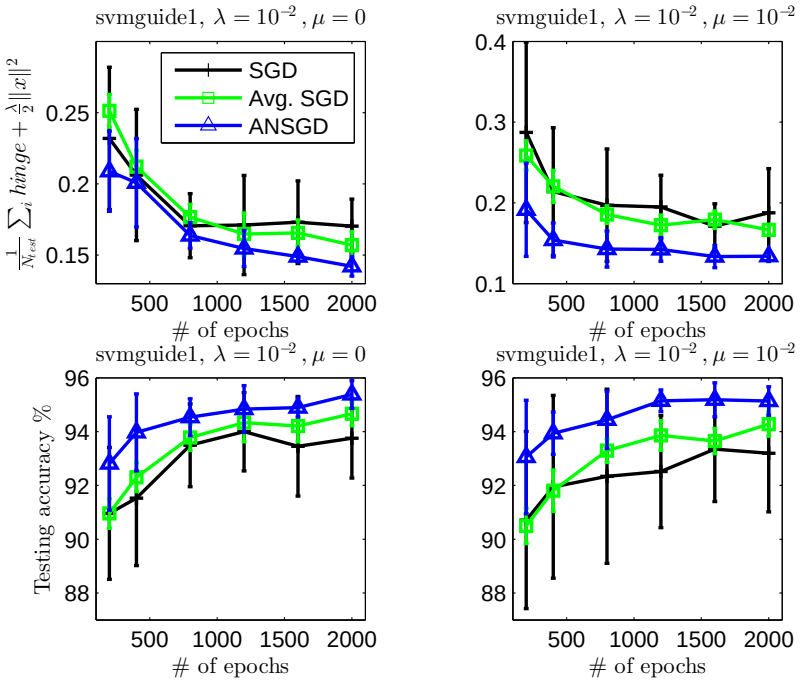

Figure 9: Classification with "svmguide1".

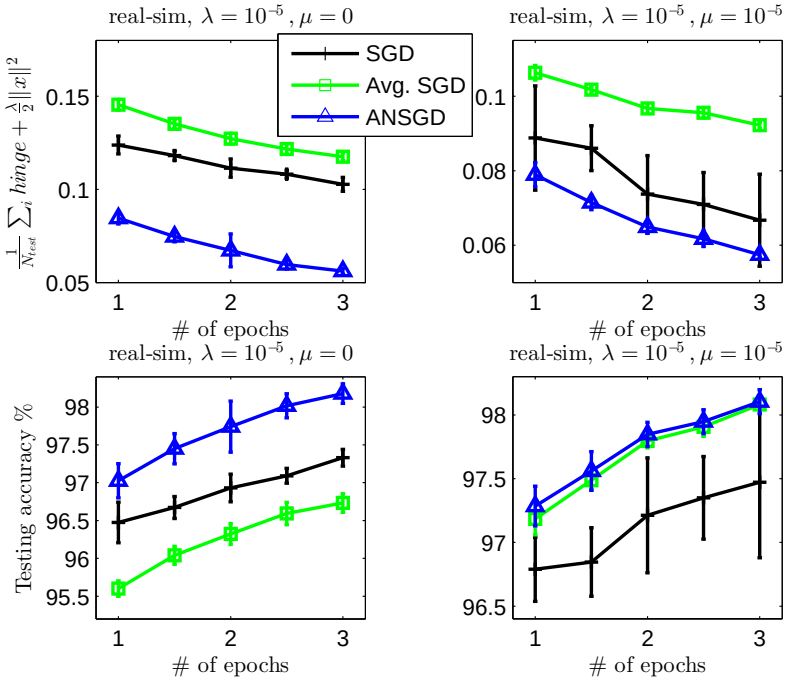

Figure 10: Classification with "real-sim".

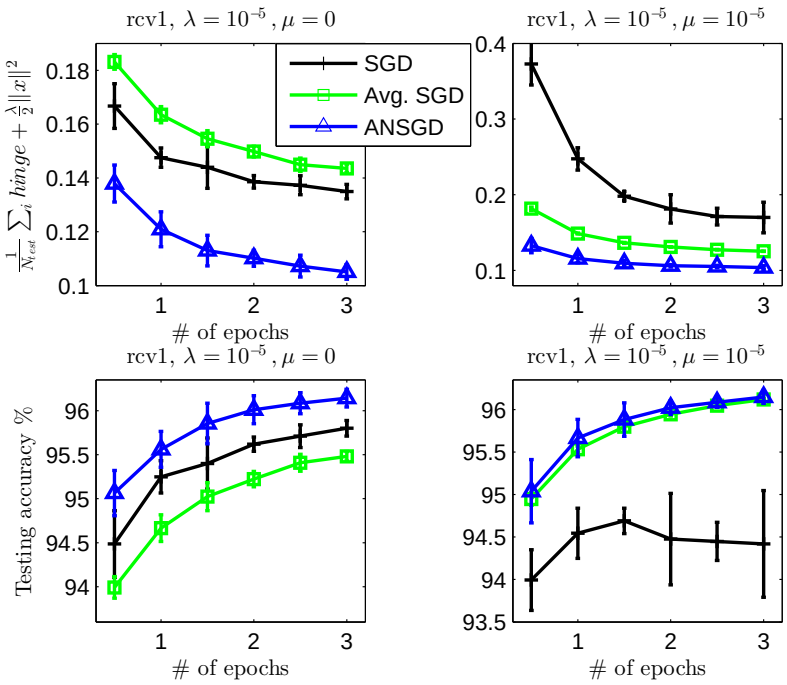

Figure 11: Classification with "rcv1".

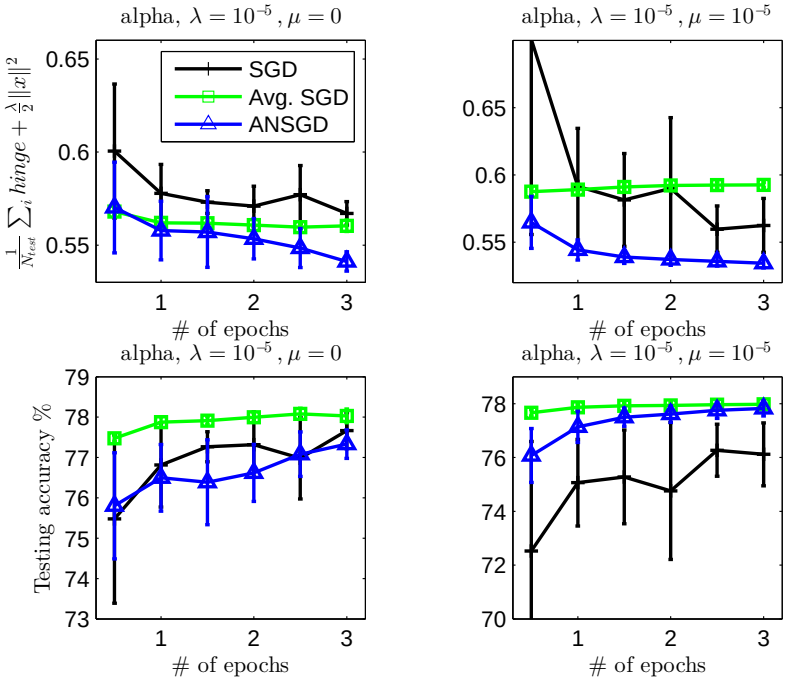

Figure 12: Classification with "alpha".

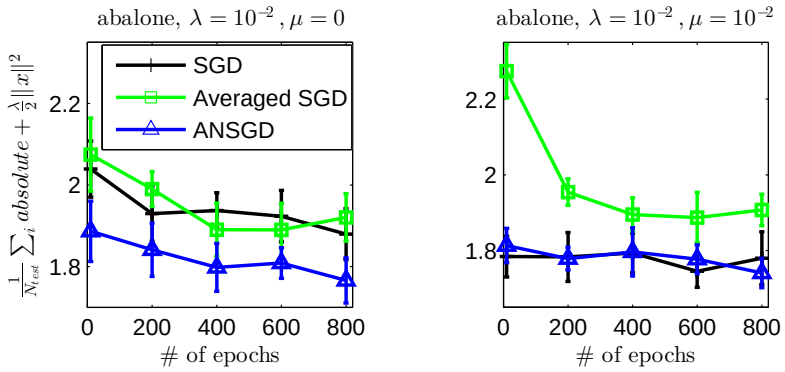

Figure 13: Regression with "abalone".

convergence rates.

In the second set of experiments, we compare ANSGD with AC-SA and its strongly convex version. Results are in Fig.14, 15, 16 and 17. In all experiments our ANSGD significantly outperforms AC-SA, and is much more stable. These experiments confirm the theoretically better rates discussed in Sec.2.5.1.

2.6 Conclusions of this Chapter

We introduce a different composite setting for nonsmooth functions. Under this setting we propose a stochastic smoothing method and a novel stochastic algorithm ANSGD. Convergence analysis show that it achieves (nearly) optimal rates under both convex and strongly convex assumptions. We also propose a "Batch-to-Online" conversion for online learning, and show that optimal regrets can be obtained.

We will extend our method to constrained minimizations, as well as cases when the approximated function $\hat{f}()$ is not easily obtained by maximizing $\mathbf{u}$. Nesterov's excessive gap technique has the "true" optimal $1/t^2$ bound, and we will investigate the possibility of integrating it in our algorithm. Exploiting links with statistical learning theories may also be promising.

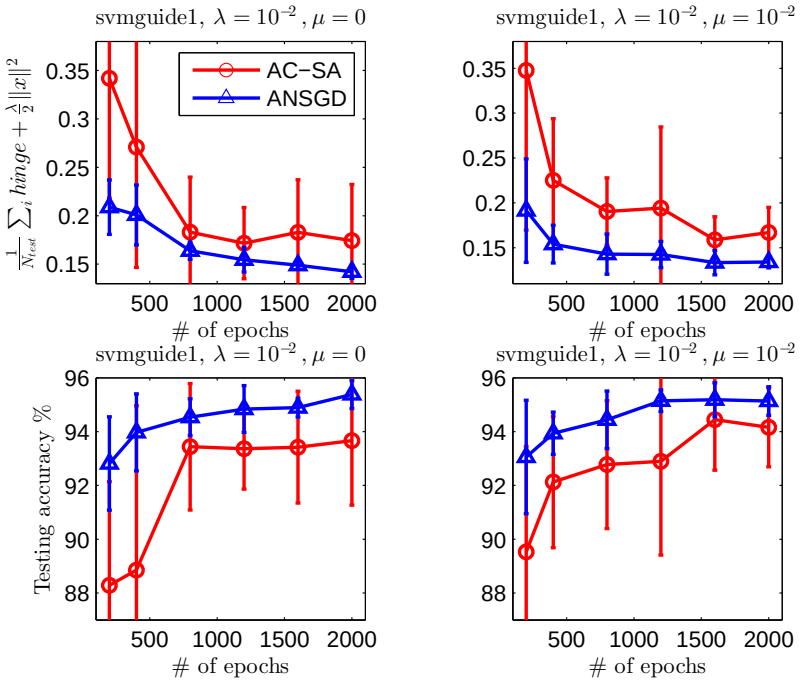

Figure 14: Classification with "svmguide1".

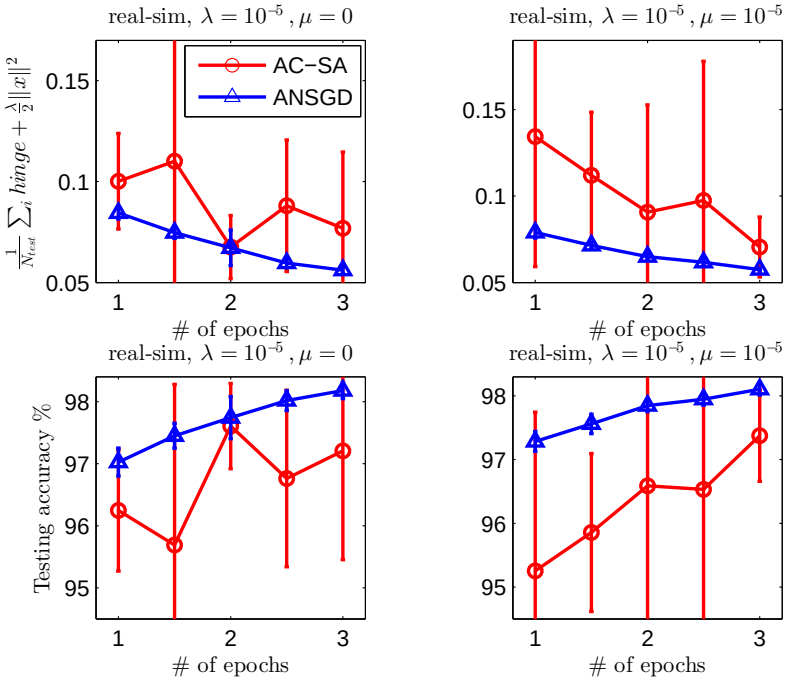

Figure 15: Classification with "real-sim".

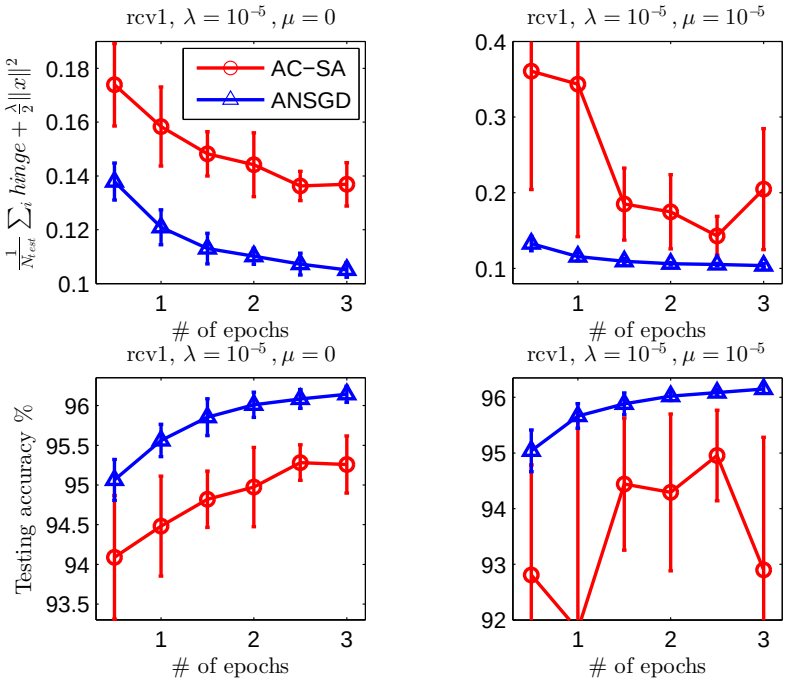

Figure 16: Classification with "rcv1".

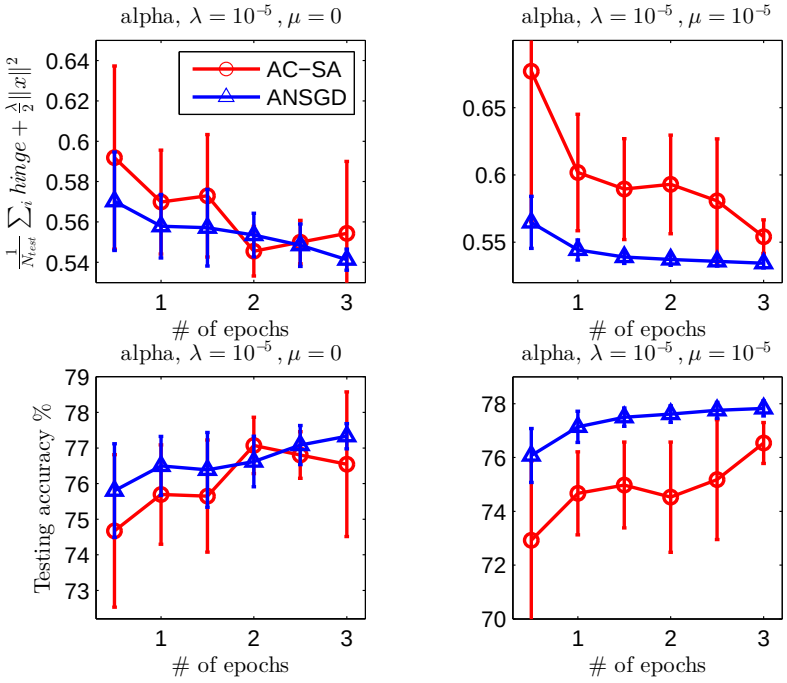

Figure 17: Classification with "alpha".

CHAPTER III

STOCHASTIC ALTERNATING DIRECTION METHOD OF MULTIPLIERS

3.1 Introduction

The Alternating Direction Method of Multipliers (ADMM) (Glowinski and Marroco [1975], Gabay and Mercier [1976]) is a very simple computational method for optimization proposed in 1970s. It stemmed from the augmented Lagrangian method (also known as the method of multipliers) dating back to late 1960s. The theoretical aspects of ADMM have been studied since 1980s, and its global convergence was established in the literature (Gabay [1983], Glowinski and Tallec [1989], Eckstein and Bertsekas [1992]). As reviewed in the comprehensive paper (Boyd et al. [2010]), with the ability of dealing with objective functions separately and synchronously, ADMM turned out to be a natural fit in the field of large-scale data-distributed machine learning and big-data related optimization, and therefore received significant amount of attention in the last few years. Considerable work was conducted thereafter. On the theoretical side, ADMM was shown to have an $O(1/N)$ rate of convergence for convex problems (Monteiro and Svaiter [2010], He and Yuan [2012a,b], Wang and Banerjee [2012]), where N stands for the number of iterations. When objective functions are strongly convex and Lipschitz smooth, linear convergence rates were reported very recently (Hong and Luo [2012], Deng and Yin [2012]). On the practical side, ADMM has been applied to a wide range of application domains, such as compressed sensing (Yang and Zhang [2011]), image restoration (Goldstein and Osher [2009]), video processing and matrix completion (Goldfarb et al. [2010]). Besides that, many variations of this classical method have been recently developed, such as linearized (Goldfarb et al. [2010], Zhang et al. [2011], Yang and Yuan [2012]), accelerated (Goldfarb et al. [2010]) and online (Wang and Banerjee [2012]) ADMM. However, most of these variants including the classic one implicitly assume full accessibilty of true data values, while in reality one can hardly ignore the

existence of noise. A more natural way of handling this issue is to consider unbiased or even biased observations of true data, which leads us to the stochastic setting.

3.1.1 Stochastic Setting for ADMM

In this chapter, we study a family of convex optimization problems where our objective functions are stochastic and composite. Specifically, we are interested in the following equality-constrained stochastic optimization:

$$\min_{\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y}} \mathbb{E}_{\boldsymbol{\xi}}\theta_1(\mathbf{x},\boldsymbol{\xi}) + \theta_2(\mathbf{y}) \text{ s.t. } A\mathbf{x} + B\mathbf{y} = \mathbf{b}, \tag{74}$$

where $\mathbf{x} \in \mathbb{R}^{d_1}, \mathbf{y} \in \mathbb{R}^{d_2}, A \in \mathbb{R}^{m\times d_1}, B \in \mathbb{R}^{m\times d_2}, \mathbf{b} \in \mathbb{R}^m$, $\mathcal{X}$ is a convex compact set, and $\mathcal{Y}$ is a closed convex set. We use the notation θ_1 for both the instance function value $\theta_1(\mathbf{x},\boldsymbol{\xi})$ and its expectation $\theta_1(\mathbf{x}) \equiv \mathbb{E}_{\boldsymbol{\xi}}\theta_1(\mathbf{x},\boldsymbol{\xi})$. We are able to draw a sequence of identical and independent (i.i.d.) observations from the random vector $\boldsymbol{\xi}$ that obeys a fixed but unknown distribution P. When $\boldsymbol{\xi}$ is deterministic, we can recover the traditional problem formulation of ADMM (Boyd et al. [2010]). In our most general setting, real-valued functions $\theta_1(\cdot)$ and $\theta_2(\cdot)$ are convex but not necessarily continuously differentiable. We will make additional assumptions in Section 3.4, in which we suggest more structural information on θ_1.

3.1.2 Motivations

The stochasticity of the proposed setting is inspired by the structural risk minimization principle (Vapnik [2000]). Under this principle, a statistical learning system's goal is to minimize the *regularized expected risk function*: $R(\mathbf{x}) \equiv \mathbb{E}_{\boldsymbol{\xi}}L(\mathbf{x},\boldsymbol{\xi}) + \Omega(\mathbf{x})$, where $L(\mathbf{x},\boldsymbol{\xi})$ is the *loss* incurred when applying prediction rule $\mathbf{x}$ on a sample $\boldsymbol{\xi}$, and Ω is a regularizer. In the batch learning setting, one uses a set of training samples to minimize the *regularized empirical risk function* $R_{\text{emp}}(\mathbf{x}) \equiv \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x},\boldsymbol{\xi}_i) + \Omega(\mathbf{x})$. With high probability, R and R_{emp} are close when the number of samples is large (Vapnik [2000]). However, to minimize R_{emp} one has to handle larger amount of samples which becomes less efficient under time and resource constraints. In the *stochastic* setting, in each iteration $\mathbf{x}$ is updated based on one noisy sample drawn from P instead of a finite training set. One obvious advantage is that the update costs much less time and resources than in the batch setting. Another

advantage we will show later in this paper is that, when carefully designed, our algorithm optimizes the expected risk *directly* with good rates of convergence.

The proposed stochastic ADMM setting fits perfectly with the regularized expected risk minimization. Putting it into our canonical form (131): $\theta_1(\mathbf{x}, \boldsymbol{\xi}) = L(\mathbf{x}, \boldsymbol{\xi})$, $\theta_2(\mathbf{y}) = \Omega(\mathbf{y})$, and the constraint becomes $\mathbf{x} = \mathbf{y}$. Beyond this simple formulation, the objective separation of ADMM is so flexible that one can use a more general linear constraint $A\mathbf{x} + B\mathbf{y} = \mathbf{b}$ to model the *complex* structural information encoded in the regularizer $\Omega(\mathbf{x})$. For example, if $\Omega(v_1, v_2) = |v_1 - v_2|$, we could add a variable v_{12}, a linear constraint $v_1 - v_2 = v_{12}$, and simply minimize $\Omega(v_{12}) = |v_{12}|$, which is easier to handle under our stochastic setting for ADMM. More examples will be given in Section 3.5.

3.1.3 Contributions of this Chapter

We propose a stochastic setting of the ADMM problem and also design the Stochastic ADMM algorithm to solve this problem. A key algorithmic feature of our Stochastic ADMM that distinguishes our method from previous ADMM and its variants is the first-order approximation of θ_1 that we used to modify the augmented Lagrangian. This simple modification is not only necessary for the convergence analysis of our stochastic method, but also makes our method applicable to a more general class of convex objective functions which might not have a closed-form solution in minimizing the augmented θ_1 directly. Moreover, the linearization makes the updates simpler and faster, as demonstrated by the examples in Section 3.5.

We establish convergence rates under various structural assumptions of θ_1: $O(1/\sqrt{t})$ for convex functions and $O(\log t/t)$ for strongly convex functions in terms of both the objective value and the feasibility violation. By contrast, recent research (He and Yuan [2012a,b], Wang and Banerjee [2012]) only show the convergence of ADMM *indirectly* in terms of the satisfaction of variational inequalities. We also demonstrate the usefulness of our algorithm with a novel application in Graph-Guided Support Vector Machine.

3.1.4 Related Work

A related setting named online ADMM was proposed in (Wang and Banerjee [2012]). In this setting, one does not assume $\boldsymbol{\xi}$ to be i.i.d., nor the objective to be stochastic, and the minimization of *regret* is concerned:

$$R(\mathbf{x}_{[1:t]}) \equiv \sum_{k=1}^{t}[\theta_1(\mathbf{x}_k, \boldsymbol{\xi}_k) + \theta_2(\mathbf{y}_k)] - \inf_{A\mathbf{x}+B\mathbf{y}=\mathbf{b}} \sum_{k=1}^{t}[\theta_1(\mathbf{x}, \boldsymbol{\xi}_k) + \theta_2(\mathbf{y})].$$

Besides that, it also differs from our stochastic ADMM algorithmically: a nonlinearized θ_1 is used in online ADMM, while a linearized one is adopted in our algorithm.

In an independent work (Suzuki [2013]), the author also linearized θ_1, and proposed dual averaging and proximal gradient methods for problem (131). The proposed OPG-ADMM algorithm enjoys the same order of convergence rates as our stochastic ADMM.

3.1.5 Notations

Throughout this paper, we denote a subgradient of a function f as f'. When f is differentiable, we will use ∇f . We denote by

$$\theta(\mathbf{u}) \equiv \theta_1(\mathbf{x}) + \theta_2(\mathbf{y})$$

the sum of the stochastic and the deterministic functions. For simplicity and clarity, we will use the following notations to denote stacked vectors or tuples:

$$\begin{aligned} \mathbf{u} \equiv \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}, \mathbf{w} \equiv \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \boldsymbol{\lambda} \end{pmatrix}, \mathbf{w}_k \equiv \begin{pmatrix} \mathbf{x}_k \\ \mathbf{y}_k \\ \boldsymbol{\lambda}_k \end{pmatrix}, \bar{\mathbf{u}}_k \equiv \begin{pmatrix} \frac{1}{k}\sum_{i=0}^{k-1} \mathbf{x}_i \\ \frac{1}{k}\sum_{i=1}^{k} \mathbf{y}_i \end{pmatrix}, \\ \bar{\mathbf{w}}_k \equiv \begin{pmatrix} \frac{1}{k}\sum_{i=1}^{k} \mathbf{x}_i \\ \frac{1}{k}\sum_{i=1}^{k} \mathbf{y}_i \\ \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{\lambda}_i \end{pmatrix}, F(\mathbf{w}) \equiv \begin{pmatrix} -A^T\boldsymbol{\lambda} \\ -B^T\boldsymbol{\lambda} \\ A\mathbf{x} + B\mathbf{y} - \mathbf{b} \end{pmatrix}, \mathcal{W} \equiv \begin{pmatrix} \mathcal{X} \\ \mathcal{Y} \\ \mathbb{R}^m \end{pmatrix}. \end{aligned} \tag{75}$$

the For a positive semidefinite matrix $G \in \mathbb{R}^{d_1 \times d_1}$, we define the G-norm of a vector as $\|\mathbf{x}\|_G := \|G^{1/2}\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T G\mathbf{x}}$. We use $\langle \cdot, \cdot \rangle$ to denote the inner product in a finite dimensional Euclidean space. When there is no ambiguity, we often use $\|\cdot\|$ to denote the

Euclidean norm $\|\cdot\|_2$. For a differentiable function $\omega(\cdot)$, Bregman divergence is defined as $D(\mathbf{u},\mathbf{v}) \equiv \omega(\mathbf{u}) - \omega(\mathbf{v}) - \langle \nabla\omega(\mathbf{v}), \mathbf{u}-\mathbf{v}\rangle$.

We assume that the optimal solution of (131) exists, and is denoted as $\mathbf{u}_* \equiv (\mathbf{x}_*^T, \mathbf{y}_*^T)^T$. The following quantities appear frequently in our convergence analysis.

$$\begin{aligned} &\boldsymbol{\delta}_k \equiv \theta_1'(\mathbf{x}_{k-1}, \boldsymbol{\xi}_k) - \theta_1'(\mathbf{x}_{k-1}), \\ &D_{\mathcal{X}} \equiv \sup_{\mathbf{x}_a, \mathbf{x}_b \in \mathcal{X}} \|\mathbf{x}_a - \mathbf{x}_b\|, \quad D_{\mathbf{y}_*, B} \equiv \|B(\mathbf{y}_0 - \mathbf{y}_*)\|. \end{aligned} \tag{76}$$

3.1.6 Assumptions

Before presenting the algorithm and convergence results, we list the following assumptions that will be used in our statements. These assumptions provide bounds for the magnitude and variance of subgradients for the stochastic function.

Assumption 2. *For all* $\mathbf{x} \in \mathcal{X}$, $\mathbb{E}\Big[\|\theta_1'(\mathbf{x}, \boldsymbol{\xi})\|^2\Big] \le M^2$.

Assumption 3. *For all* $\mathbf{x} \in \mathcal{X}$,

$$\mathbb{E}\Big[\exp\Big\{\|\theta_1'(\mathbf{x}, \boldsymbol{\xi})\|^2 / M^2\Big\}\Big] \le \exp\{1\}.$$

Assumption 4. *For all* $\mathbf{x} \in \mathcal{X}$,

$$\mathbb{E}\left[\|\theta_1'(\mathbf{x}, \boldsymbol{\xi}) - \theta_1'(\mathbf{x})\|^2\right] \le \sigma^2.$$

3.2 Stochastic ADMM Algorithm

Directly solving problem (131) can be nontrivial, even if $\boldsymbol{\xi}$ is deterministic and the equality constraint is as simple as $\mathbf{x} - \mathbf{y} = \mathbf{0}$. For example, using the augmented Lagrangian method, one has to minimize the *augmented Lagrangian*:

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X}, \mathbf{y}\in\mathcal{Y}} \mathcal{L}_\beta(\mathbf{x}, \mathbf{y}, \lambda) \equiv \min_{\mathbf{x}\in\mathcal{X}, \mathbf{y}\in\mathcal{Y}} \Big[&\theta_1(\mathbf{x}) + \theta_2(\mathbf{y}) - \\ &\langle \boldsymbol{\lambda}, A\mathbf{x} + B\mathbf{y} - \mathbf{b}\rangle + \frac{\beta}{2}\|A\mathbf{x} + B\mathbf{y} - \mathbf{b}\|^2\Big], \end{aligned} \tag{77}$$

where β is a pre-defined penalty parameter. This problem is at least not easier than solving the original one. The (deterministic) ADMM (Alg.2) solves this problem in a one-sweep Gauss-Seidel manner: minimizing $\mathcal{L}_\beta$ w.r.t. $\mathbf{x}$ and $\mathbf{y}$ alternatively given the other fixed, followed by a penalty update over the Lagrangian multiplier $\boldsymbol{\lambda}$.

Algorithm 2 Deterministic ADMM

0. Initialize $\mathbf{y}_0$ and $\boldsymbol{\lambda}_0 = 0$.
for $k = 0, 1, 2, \ldots$ **do**
1. $\mathbf{x}_{k+1} \leftarrow \arg\min_{\mathbf{x}\in\mathcal{X}} \mathcal{L}_\beta(\mathbf{x}, \mathbf{y}_k, \boldsymbol{\lambda}_k)$.
2. $\mathbf{y}_{k+1} \leftarrow \arg\min_{\mathbf{y}\in\mathcal{Y}} \mathcal{L}_\beta(\mathbf{x}_{k+1}, \mathbf{y}, \boldsymbol{\lambda}_k)$.
3. $\boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{\lambda}_k - \beta\,(A\mathbf{x}_{k+1} + B\mathbf{y}_{k+1} - \mathbf{b})$.
end for

A variant deterministic algorithm named linearized ADMM replaces Line 1 of Alg.2 by

$$\mathbf{x}_{k+1} \leftarrow \arg\min_{\mathbf{x}\in\mathcal{X}}\Big[\theta_1(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}_k\|_G^2 + \frac{\beta}{2}\left\|(A\mathbf{x} + B\mathbf{y}_k - \mathbf{b}) - \boldsymbol{\lambda}_k/\beta\right\|^2\Big],$$

where $G \in \mathbb{R}^{d_1\times d_1}$ is positive semidefinite. This variant can be regarded as a generalization of the original ADMM. When $G = \mathbf{0}$, it is the same as Alg.2. When $G = rI_{d_1} - \beta A^T A$, it is equivalent to the following linearized proximal point method:

$$\mathbf{x}_{k+1} \leftarrow \arg\min_{\mathbf{x}\in\mathcal{X}}\Big\{\theta_1(\mathbf{x}) + \frac{r}{2}\|\mathbf{x} - \mathbf{x}_k\|^2 + \beta(\mathbf{x} - \mathbf{x}_k)^T\left[A^T(A\mathbf{x_k} + B\mathbf{y}_k - \mathbf{b} - \boldsymbol{\lambda}_k/\beta)\right]\Big\}.$$

Note that the linearization is only applied to the quadratic function $\|(A\mathbf{x} + B\mathbf{y}_k - \mathbf{b}) - \boldsymbol{\lambda}_k/\beta\|^2$, but not to θ_1. This approximation helps in some cases when Line 1 of Alg.2 does not produce a closed-form solution given the quadratic term. For example, let $\theta_1(\mathbf{x}) = \|\mathbf{x}\|_1$ and A not identity.

As given in Alg.3, we propose a *Stochastic Alternating Direction Method of Multipliers* (*Stochastic ADMM*) algorithm. Our algorithm shares some features with the classical and the linearized ADMM. One can see that Line 2 and 3 are essentially the same as before. However we have a different updating rule for $\mathbf{x}$ as shown in Line 1, where we define an *approximated augmented Lagrangian*:

$$\begin{aligned}\hat{\mathcal{L}}_{\beta,k}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) \equiv \theta_1(\mathbf{x}_k) + \langle\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}\rangle + \theta_2(\mathbf{y}) - \\ \langle\boldsymbol{\lambda}, A\mathbf{x} + B\mathbf{y} - \mathbf{b}\rangle + \frac{\beta}{2}\|A\mathbf{x} + B\mathbf{y} - \mathbf{b}\|^2 + \frac{\|\mathbf{x} - \mathbf{x}_k\|^2}{2\eta_{k+1}}.\end{aligned} \tag{78}$$

There are two differences between $\mathcal{L}_\beta$ (77) and $\hat{\mathcal{L}}_{\beta,k}$ (78). First, we replace $\theta_1(\mathbf{x})$ with a first-order approximation of $\theta_1(\mathbf{x}, \boldsymbol{\xi}_{k+1})$ at $\mathbf{x}_k$: $\theta_1(\mathbf{x}_k) + \mathbf{x}^T\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})$. This approximation has

the same flavour of the stochastic mirror descent (Nemirovski et al. [2009]) used for solving a one-variable stochastic convex problem. Second, similar to the linearized ADMM, we add an l_2-norm prox-function $\|\mathbf{x}-\mathbf{x}_k\|^2$ but scale it by a time-varying stepsize η_{k+1}. As we will see in Section 3.3, the choice of this stepsize is crucial in guaranteeing a convergence.

Algorithm 3 Stochastic ADMM

0. Initialize $\mathbf{x}_0, \mathbf{y}_0$ and set $\boldsymbol{\lambda}_0 = 0$.
for $k = 0, 1, 2, \ldots$ **do**
1. $\mathbf{x}_{k+1} \leftarrow \arg\min_{\mathbf{x}\in\mathcal{X}} \hat{\mathcal{L}}_{\beta,k}(\mathbf{x}, \mathbf{y}_k, \boldsymbol{\lambda}_k)$.
2. $\mathbf{y}_{k+1} \leftarrow \arg\min_{\mathbf{y}\in\mathcal{Y}} \hat{\mathcal{L}}_{\beta,k}(\mathbf{x}_{k+1}, \mathbf{y}, \boldsymbol{\lambda}_k)$.
3. $\boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{\lambda}_k - \beta\,(A\mathbf{x}_{k+1} + B\mathbf{y}_{k+1} - \mathbf{b})$.
end for

3.3 Main Results of Convergence Rates

In this section, we will show that our Stochastic ADMM given in Alg.3 exhibits a rate $O(1/\sqrt{t})$ of convergence in terms of both the objective value *and* the feasibility violation:

$$\mathbb{E}\Big[\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\|_2\Big] = O(1/\sqrt{t}).$$

Before we address the main theorem on convergence rates, we will start with the following simple lemma, which is a very useful result by implementing Bregman divergence as a prox-function in proximal methods.

Lemma 8. *Let $l(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ be a convex differentiable function with gradient $\mathbf{g}$. Let scalar $s \geq 0$. For any vector $\mathbf{u}$ and $\mathbf{v}$, denote their Bregman divergence as $D(\mathbf{u}, \mathbf{v})$. If $\forall \mathbf{u} \in \mathcal{X}$,*

$$\mathbf{x}^* \equiv \arg\min_{\mathbf{x}\in\mathcal{X}} l(\mathbf{x}) + sD(\mathbf{x}, \mathbf{u}), \tag{79}$$

then

$$\langle \mathbf{g}(\mathbf{x}^*), \mathbf{x}^* - \mathbf{x}\rangle \leq s\left[D(\mathbf{x}, \mathbf{u}) - D(\mathbf{x}, \mathbf{x}^*) - D(\mathbf{x}^*, \mathbf{u})\right].$$

Proof. Invoking the optimality condition for (135), we have

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

which is equivalent to

$$\begin{aligned}\langle \mathbf{g}(\mathbf{x}^*), \mathbf{x}^* - \mathbf{x} \rangle &\leq s \langle \nabla D(\mathbf{x}^*, \mathbf{u}), \mathbf{x} - \mathbf{x}^* \rangle \\ &= s \langle \nabla \omega(\mathbf{x}^*) - \nabla \omega(\mathbf{u}), \mathbf{x} - \mathbf{x}^* \rangle \\ &= s \left[D(\mathbf{x}, \mathbf{u}) - D(\mathbf{x}, \mathbf{x}^*) - D(\mathbf{x}^*, \mathbf{u}) \right].\end{aligned}$$

□

Utilizing the above lemma, we are able to obtain an upper bound of the variation of the Lagrangian function and its first order approximation based on each iteration points.

Lemma 9. $\forall \mathbf{w} \in \mathcal{W}, k \geq 0$ *we have*

$$\begin{aligned}&\theta_1(\mathbf{x}_k) + \theta_2(\mathbf{y}_{k+1}) - \theta(\mathbf{u}) + (\mathbf{w}_{k+1} - \mathbf{w})^T F(\mathbf{w}_{k+1}) \leq \\ &\frac{\eta_{k+1} \|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2}{2} + \frac{\|\mathbf{x}_k - \mathbf{x}\|^2 - \|\mathbf{x}_{k+1} - \mathbf{x}\|^2}{2\eta_{k+1}} + \\ &\frac{\beta}{2} \left(\|A\mathbf{x} + B\mathbf{y}_k - \mathbf{b}\|^2 - \|A\mathbf{x} + B\mathbf{y}_{k+1} - \mathbf{b}\|^2 \right) + \\ &\langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2\beta} \left(\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_k\|_2^2 - \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}\|_2^2 \right).\end{aligned} \tag{80}$$

Proof. Due to the convexity of θ_1 and using the definition of $\boldsymbol{\delta}_k$, we have

$$\theta_1(\mathbf{x}_k) - \theta_1(\mathbf{x}) \leq \langle \theta_1'(\mathbf{x}_k), \mathbf{x}_k - \mathbf{x} \rangle = \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x} \rangle + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k \rangle + \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_k - \mathbf{x}_{k+1} \rangle. \tag{81}$$

Applying Lemma 13 to Line 1 of Alg.3 and taking $D(\mathbf{u}, \mathbf{v}) = \frac{1}{2}\|\mathbf{v} - \mathbf{u}\|^2$, we have

$$\begin{aligned}&\left\langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}) + A^T \left[\beta(A\mathbf{x}_{k+1} + B\mathbf{y}_k - \mathbf{b}) - \boldsymbol{\lambda}_k \right], \mathbf{x}_{k+1} - \mathbf{x} \right\rangle \\ &\leq \frac{1}{2\eta_{k+1}} \left(\|\mathbf{x}_k - \mathbf{x}\|^2 - \|\mathbf{x}_{k+1} - \mathbf{x}\|^2 - \|\mathbf{x}_k - \mathbf{x}_{k+1}\|^2 \right)\end{aligned} \tag{82}$$

Combining (81) and (82) we have

$$
\begin{aligned}
&\theta_1(\mathbf{x}_k) - \theta_1(\mathbf{x}) + \left\langle \mathbf{x}_{k+1} - \mathbf{x}, -A^T\boldsymbol{\lambda}_{k+1} \right\rangle \\
&\overset{(81)}{\leq} \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x} \rangle + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k \rangle + \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_k - \mathbf{x}_{k+1} \rangle + \\
&\quad \left\langle \mathbf{x}_{k+1} - \mathbf{x}, A^T \left[\beta(A\mathbf{x}_{k+1} + B\mathbf{y}_{k+1} - \mathbf{b}) - \boldsymbol{\lambda}_k \right] \right\rangle \\
&= \left\langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}) + A^T \left[\beta(A\mathbf{x}_{k+1} + B\mathbf{y}_k - \mathbf{b}) - \boldsymbol{\lambda}_k \right], \mathbf{x}_{k+1} - \mathbf{x} \right\rangle + \\
&\quad \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k \rangle + \left\langle \mathbf{x} - \mathbf{x}_{k+1}, \beta A^T B(\mathbf{y}_k - \mathbf{y}_{k+1}) \right\rangle + \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_k - \mathbf{x}_{k+1} \rangle \\
&\overset{(82)}{\leq} \frac{1}{2\eta_{k+1}} \left(\|\mathbf{x}_k - \mathbf{x}\|^2 - \|\mathbf{x}_{k+1} - \mathbf{x}\|^2 - \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 \right) + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k \rangle + \\
&\quad \left\langle \mathbf{x} - \mathbf{x}_{k+1}, \beta A^T B(\mathbf{y}_k - \mathbf{y}_{k+1}) \right\rangle + \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_k - \mathbf{x}_{k+1} \rangle
\end{aligned} \tag{83}
$$

We handle the last two terms separately:

$$
\begin{aligned}
&\left\langle \mathbf{x} - \mathbf{x}_{k+1}, \beta A^T B(\mathbf{y}_k - \mathbf{y}_{k+1}) \right\rangle = \beta \langle A\mathbf{x} - A\mathbf{x}_{k+1}, B\mathbf{y}_k - B\mathbf{y}_{k+1} \rangle \\
&= \frac{\beta}{2} \left[\left(\|A\mathbf{x} + B\mathbf{y}_k - \mathbf{b}\|^2 - \|A\mathbf{x} + B\mathbf{y}_{k+1} - \mathbf{b}\|^2 \right) + \left(\|A\mathbf{x}_{k+1} + B\mathbf{y}_{k+1} - \mathbf{b}\|^2 - \|A\mathbf{x}_{k+1} + B\mathbf{y}_k - \mathbf{b}\|^2 \right) \right] \\
&\leq \frac{\beta}{2} \left(\|A\mathbf{x} + B\mathbf{y}_k - \mathbf{b}\|^2 - \|A\mathbf{x} + B\mathbf{y}_{k+1} - \mathbf{b}\|^2 \right) + \frac{1}{2\beta} \|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k\|^2
\end{aligned} \tag{84}
$$

and

$$
\langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_k - \mathbf{x}_{k+1} \rangle \leq \frac{\eta_{k+1} \|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2}{2} + \frac{\|\mathbf{x}_k - \mathbf{x}_{k+1}\|^2}{2\eta_{k+1}}, \tag{85}
$$

where the last step is due to Young's inequality. Inserting (142) and (85) into (83), we have

$$
\begin{aligned}
&\theta_1(\mathbf{x}_k) - \theta_1(\mathbf{x}) + \left\langle \mathbf{x}_{k+1} - \mathbf{x}, -A^T\boldsymbol{\lambda}_{k+1} \right\rangle \\
&\leq \frac{1}{2\eta_{k+1}} \left(\|\mathbf{x}_k - \mathbf{x}\|^2 - \|\mathbf{x}_{k+1} - \mathbf{x}\|^2 \right) + \frac{\eta_{k+1} \|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2}{2} + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k \rangle \\
&+ \frac{\beta}{2} \left(\|A\mathbf{x} + B\mathbf{y}_k - \mathbf{b}\|^2 - \|A\mathbf{x} + B\mathbf{y}_{k+1} - \mathbf{b}\|^2 \right) + \frac{1}{2\beta} \|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k\|^2,
\end{aligned} \tag{86}
$$

Due to the optimality condition of Line 2 in Alg.3 and the convexity of θ_2, we have

$$
\theta_2(\mathbf{y}_{k+1}) - \theta_2(\mathbf{y}) + \left\langle \mathbf{y}_{k+1} - \mathbf{y}, -B^T\boldsymbol{\lambda}_{k+1} \right\rangle \leq 0. \tag{87}
$$

Using Line 3 in Alg.3, we have

$$
\begin{aligned}
&\langle \boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}, A\mathbf{x}_{k+1} + B\mathbf{y}_{k+1} - \mathbf{b} \rangle \\
&= \frac{1}{\beta} \langle \boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}, \boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k+1} \rangle \\
&= \frac{1}{2\beta} \left(\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_k\|^2 - \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}\|^2 - \|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k\|^2 \right)
\end{aligned} \tag{88}
$$

Taking the summation of inequalities (86) (87) and (88), we obtain the result as desired.

□

In what follows, we will present our main theorem of the convergence in two fashions, both in terms of expectation and probability satisfication.

Theorem 7. *Let* $\eta_k = \frac{D_{\mathcal{X}}}{M\sqrt{2k}}$ *for all* $k \geq 1$. *Define*

$$M_1(t) \equiv \frac{\sqrt{2}D_{\mathcal{X}}M}{\sqrt{t}} \quad \textit{and} \quad M_2(t) \equiv \frac{\beta D^2_{\mathbf{y}_*,B} + \rho^2/\beta}{2t}. \tag{89}$$

Then $\forall \rho > 0$ *and* $t \geq 1$ *we have:*

1. *Under Assumption 2*

$$\mathbb{E}[\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\|] \leq M_1(t) + M_2(t). \tag{90}$$

2. *Under Assumption 2 and 3,* $\forall \Omega > 0$

$$\begin{aligned} &Prob\Big\{\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\| > \\ &(1 + \Omega/2 + 2\sqrt{2\Omega})M_1(t) + M_2(t)\Big\} \leq 2\exp\{-\Omega\}. \end{aligned} \tag{91}$$

Remark 1. *Observe that our proof techniques can also be adapted to the deterministic case where no noise takes place. We are able to obtain a similar result for the classic deterministic ADMM:*

$$\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\|_2 \leq \frac{\beta D^2_{\mathbf{y}_*,B}}{2t} + \frac{\rho^2}{2\beta t}.$$

The positive ρ in the preceding results controls the trade-off between the objective value reduction and the feasibility satisfaction. For a fixed ρ, one can set an optimal $\beta = \rho/D_{\mathbf{y}_*,B}$ such that the upper bound is minimized.

While the resulting $O(1/t)$ rate for the deterministic ADMM is the same as those in the existing literature, the above finding is an advance in the theoretical aspects of ADMM. Our convergence rate for general convex functions is proved in terms of both the objective value and the feasibility violation. By contrast, the existing literature (He and Yuan [2012a,b], Wang and Banerjee [2012]) only shows the convergence of ADMM in terms of the satisfaction of variational inequalities, which is not a direct measure of how fast an algorithm reaches the optimal solution.

3.4 Extensions

3.4.1 Strongly Convex θ_1

When function $\theta_1(\cdot)$ is strongly convex, the convergence rate of Stochastic ADMM can be improved to $O\left(\frac{\log t}{t}\right)$, as shown in the following result.

Theorem 8. *When θ_1 is μ-strongly convex with respect to $\|\cdot\|$, taking $\eta_k = \frac{1}{k\mu}$ in Alg.3, under Assumption 2 we have $\forall \rho > 0, t \geq 1$,*

$$\begin{aligned} &\mathbb{E}\left[\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\|_2\right] \\ &\leq \frac{M^2 \log t}{\mu t} + \frac{\mu D_{\mathcal{X}}^2}{2t} + \frac{\beta D_{\mathbf{y}_*,B}^2}{2t} + \frac{\rho^2}{2\beta t}. \end{aligned} \tag{92}$$

3.4.2 Lipschitz Smooth θ_1

Since the bounds given in Theorem 7 are related to the magnitude of subgradients, they do not provide any intuition of the performance in low-noise scenarios. With a Lipschitz smooth function θ_1, we are able to obtain convergence rates in terms of the variations of gradients, as stated in Assumption 4. Besides, under this assumption we are able to replace the unusual definition of $\bar{\mathbf{u}}_k$ in (75) with the following:

$$\bar{\mathbf{u}}_k \equiv \begin{pmatrix} \frac{1}{k}\sum_{i=1}^k \mathbf{x}_i \\ \frac{1}{k}\sum_{i=1}^k \mathbf{y}_i \end{pmatrix}. \tag{93}$$

Theorem 9. *When $\theta_1(\cdot)$ is L-Lipschitz smooth with respect to $\|\cdot\|$, taking $\eta_k = \frac{1}{L+\sigma\sqrt{2k}/D_{\mathcal{X}}}$ in Alg.3, under Assumption 4 we have $\forall \rho > 0, t \geq 1$,*

$$\begin{aligned} &\mathbb{E}\left[\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\|_2\right] \\ &\leq \frac{\sqrt{2}D_{\mathcal{X}}\sigma}{\sqrt{t}} + \frac{LD_{\mathcal{X}}^2}{2t} + \frac{\beta D_{\mathbf{y}_*,B}^2}{2t} + \frac{\rho^2}{2\beta t}. \end{aligned} \tag{94}$$

3.5 Examples and Numerical Evaluations

3.5.1 Lasso

As one of the many motivating examples given in the review of ADMM (Boyd et al. [2010]), the l_1-regularized sparse least squares problem, also known as *lasso*, fits the general class of (131) very naturally. The composite functions can be written as:

$$\theta_1(\mathbf{x}, \boldsymbol{\xi}) = \frac{1}{2}\left(l - \mathbf{x}^T\mathbf{s}\right)^2, \quad \theta_2(\mathbf{y}) = \gamma\|\mathbf{y}\|_1, \tag{95}$$

where the training sample $\boldsymbol{\xi}$ contains feature-label pair $\{\mathbf{s}, l\}$ and γ is a regularization parameter. The constraint simply becomes $A = I,\ B = -I,\ \mathbf{b} = \mathbf{0}$. Same as in the deterministic case, applying stochastic ADMM to l_1-regularized problem produces closed-form updating rules. The three updates for (95) are actually very simple:

$$\begin{aligned}
\mathbf{x}_{k+1} &\leftarrow \frac{(l_{k+1} - \mathbf{s}_{k+1}^T \mathbf{x}_k)\mathbf{s}_{k+1} + \boldsymbol{\lambda}_k + \beta \mathbf{y}_k + \mathbf{x}_k/\eta_{k+1}}{\beta + 1/\eta_{k+1}}, \\
\mathbf{y}_{k+1} &\leftarrow S_{\frac{\gamma}{\beta}}\left(\mathbf{x}_{k+1} - \boldsymbol{\lambda}_k/\beta\right), \\
\boldsymbol{\lambda}_{k+1} &\leftarrow \boldsymbol{\lambda}_k - \beta(\mathbf{x}_{k+1} - \mathbf{y}_{k+1}),
\end{aligned} \tag{96}$$

where the soft-thresholding operator $S_\alpha(\mathbf{x})$ is defined in the same way as in Boyd et al. [2010]:

$$S_\alpha(\mathbf{x}) \equiv \begin{cases} x_i - \alpha, & \text{if } x_i > \alpha \\ 0, & \text{if } |x_i| \leq \alpha \\ x_i + \alpha, & \text{if } x_i < -\alpha \end{cases}, \ \forall i.$$

Some vector-scaling operations can be saved by by replacing $\boldsymbol{\lambda}_k$ with $\beta\boldsymbol{\zeta}_k$ in (96):

$$\begin{aligned}
\mathbf{x}_{k+1} &\leftarrow \frac{(l_{k+1} - \mathbf{s}_{k+1}^T \mathbf{x}_k)\mathbf{s}_{k+1} + \beta(\boldsymbol{\zeta}_k + \mathbf{y}_k) + \mathbf{x}_k/\eta_{k+1}}{\beta + 1/\eta_{k+1}}, \\
\mathbf{y}_{k+1} &\leftarrow S_{\frac{\gamma}{\beta}}\left(\mathbf{x}_{k+1} - \boldsymbol{\zeta}_k\right), \\
\boldsymbol{\zeta}_{k+1} &\leftarrow \boldsymbol{\zeta}_k - (\mathbf{x}_{k+1} - \mathbf{y}_{k+1}).
\end{aligned}$$

For simple problems like lasso, it is indeed not necessary to formulate it as a two-variable equality-constrained optimization. Instead, we can directly minimize $\mathbb{E}(l - \mathbf{x}^T\mathbf{s})^2 + \gamma\|\mathbf{x}\|_1$ without any constraint. A popular class of methods for solving this composite-objective problem is called *proximal gradient* (Tseng [2008], Nemirovski and Yudin [1983]) or *proximal splitting* (Combettes and Pesquet [2011]), which was investigated in various communities (Daubechies et al. [2004], Combettes and Wajs [2005], Beck and Teboulle [2009], Nesterov [2007a], Wright et al. [2009]). Stochastic and online variants of these methods have also been developed recently, mainly in the large-scale machine learning and optimization literature (Langford et al. [2009], Lan [2010], Lan and Ghadimi [2011], Duchi and Singer [2009], Hu et al. [2009], Xiao [2010]). For comparison purposes, here we take the *online forward-backward splitting* method (FOBOS) (Combettes and Pesquet [2011], Duchi

and Singer [2009]) as a first example. The FOBOS can be regarded as a proximal method with linearization of θ_1:

$$\mathbf{x}_{k+1} \leftarrow \arg\min_{\mathbf{x}\in\mathcal{X}} \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}\rangle + \theta_2(\mathbf{x}) + \frac{\|\mathbf{x}-\mathbf{x}_k\|^2}{2\eta_{k+1}}. \tag{97}$$

Comparing this method with our Alg.3, we can see that (97) is actually a special Stochastic ADMM that enforces $\mathbf{x}_k = \mathbf{y}_k$ (hence $\boldsymbol{\lambda}_k = \boldsymbol{\zeta}_k = \mathbf{0}$) in every iteration k. Note that this constraint feasibility is easy to enforce only because lasso comes with an extremely simple constraint $\mathbf{x} = \mathbf{y}$. One of the most attractive features of (97) is its closed form solution for lasso in terms of soft-thresholding:

$$\mathbf{x}_{k+1} \leftarrow S_{\gamma\eta_{k+1}}\left[\mathbf{x}_k + \eta_{k+1}\left(l_{k+1} - \mathbf{s}_{k+1}^T\mathbf{x}_k\right)\mathbf{s}_{k+1}\right].$$

As we will see in our next example (Sec.3.5.2), with complex constraints, applying proximal splitting methods might not produce closed-form updates.

The second algorithm we are going to compare with is the *online ADMM* (Wang and Banerjee [2012]), which was proposed under a related but different setting of online learning. In this algorithm, the first-order approximation $\theta_1(\mathbf{x}_k) + \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}\rangle$ is replaced by the exact function $\theta_1(\mathbf{x}, \boldsymbol{\xi}_k)$, which is a very straightforward "stochastization" of the deterministic ADMM. Applying this algorithm to lasso yields the following update for $\mathbf{x}$:

$$\mathbf{x}_{k+1} \leftarrow \left[\mathbf{s}_{k+1}\mathbf{s}_{k+1}^T + (\beta + 1/\eta_{k+1})\, I\right]^{-1}\mathbf{u},$$

while $\mathbf{u} \equiv l_{k+1}\mathbf{s}_{k+1} + \beta(\boldsymbol{\zeta}_k + \mathbf{y}_k) + \frac{\mathbf{x}_k}{\eta_{k+1}}$, and the updates for $\mathbf{y}$ and $\boldsymbol{\zeta}$ remain the same as our stochastic ADMM. Comparing the $\mathbf{x}$ updates of online and stochastic ADMM, it is clear that the linearization of our algorithm results in a much simpler inner product calculation, while a rank-1 matrix inversion is required for the online ADMM. Even with the Sherman-Morrison formula, this inversion process is still slower than the stochastic ADMM.

In the following experiments, we investigate two real-world datasets to examine the efficiency of our algorithm. Table 2 shows the statistics of these datasets and parameters we used for lasso. The first dataset, Abalone, obtained from the UCI Machine Learning Repository[1], is used to predict the age of abalones from physical measurements. The second

[1] `http://archive.ics.uci.edu/ml/datasets/Abalone`

dataset, E2006-tf-idf, a part of the 10K-Corpus[2], is used to predict the volatility of stock returns, an empirical measure of the financial risk of a company. The features are tf-idf of unigrams extracted from the financial reports of companies during the years 1996-2006 (Kogan et al. [2009]).

The prediction results are shown in Fig.18 and 19. One can observe that all algorithms converge reasonably well, as expected from our discussions above. The stochastic ADMM performs slightly better than the other two in Abalone. For E2006-tf-idf, an acceptable accuracy is achieved with a fast sweep of merely $2,000$ samples, less than 25% of the entire dataset.

Table 2: Two real-world datasets and parameters for lasso.

Name	# of Training Samples	# of Testing Samples	# of Dim. (d)	γ	β	$\frac{D_{\mathcal{X}}}{M\sqrt{2}}$
Abalone	3,342	835	8	0.01	1	1
E2006-tf-idf	16,087	3,308	150,360	0.1	1	1

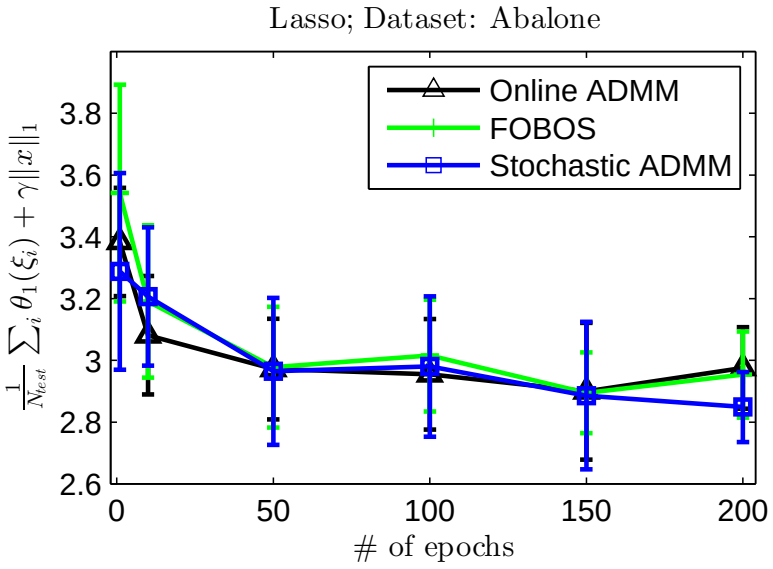

Figure 18: Lasso for Abalone Dataset.

[2] http://www.ark.cs.cmu.edu/10K/

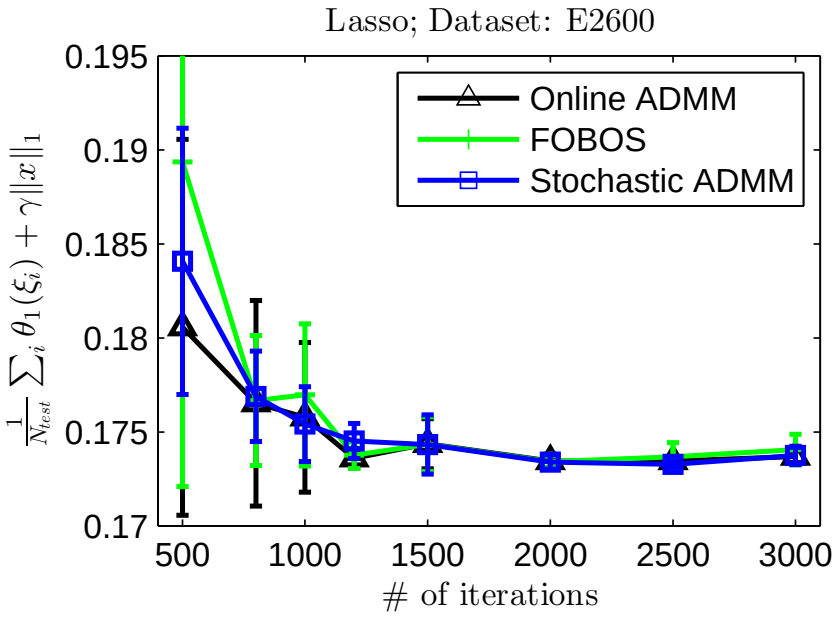

Figure 19: Lasso for E2600-tf-idf Dataset.

3.5.2 Graph-Guided SVM

Stochastic ADMM is more powerful for problems with complex equality constraints, for which proximal splitting methods such as FOBOS are no longer applicable, since there will be no closed-form for it. An important class of these problems is called the *generalized lasso* (Tibshirani and Taylor [2011]):

$$\min \mathbb{E}_{\boldsymbol{\xi}}[\frac{1}{2}\left(l - \mathbf{x}^T\mathbf{s}\right)^2] + \|F\mathbf{x}\|_1, \tag{98}$$

where the linear transformation $F \in \mathbb{R}^{f \times d_1}$ encodes the structural prior of a specific problem. When $F = I$, one recovers lasso. We can write (98) in our canonical form (131) with

$$\begin{aligned} &\theta_1(\mathbf{x}, \boldsymbol{\xi}) = \frac{1}{2}\left(l - \mathbf{x}^T\mathbf{s}\right)^2, \quad \theta_2(\mathbf{y}) = \|\mathbf{y}\|_1, \\ &A = F, \ B = -I, \ \mathbf{b} = \mathbf{0}. \end{aligned} \tag{99}$$

where $\boldsymbol{\xi} = \{\mathbf{s}, l\}$ is a feature-label pair.

As a concrete example of the generalized lasso, we evaluate our algorithm based on the *graph-guided fused lasso* (GFlasso) framework (Kim et al. [2009]), a graphical extension of the well-known *fused lasso* (Tibshirani et al. [2004]). Denote graph $\mathcal{G} \equiv \{\mathcal{V}, \mathcal{E}\}$, in which

$\mathcal{V} = \{x_1, \ldots, x_d\}$ is a set of the d variables of $\mathbf{x}$ and $\mathcal{E}$ is the set of edges among $\mathcal{V}$. Each edge $\{i, j\}$ is assigned with a weight w_{ij}. The optimization of GFlasso can thus be formulated as:

$$\min \mathbb{E}_{\boldsymbol{\xi}}[\frac{1}{2}\left(l - \mathbf{x}^T\mathbf{s}\right)^2] + \gamma\|\mathbf{x}\|_1 + \nu \sum_{\{i,j\}\in\mathcal{E}} w_{ij}|x_i - x_j|. \tag{100}$$

The only difference between GFlasso and lasso is the last term, referred as the fusion penalty (Tibshirani et al. [2004]), which penalizes the differences among variables connected in $\mathcal{G}$. A carefully designed fusion penalty helps in further reducing the risk of overfitting of our model over the training data. To implement Alg.3 to this problem, we only need to formulate the linear transformation F, which is very simple for GFlasso: $F_{ki} = w_{ij}$ and $F_{kj} = -w_{ij}$ for any edge $\{i, j\}_k \in \mathcal{E}$.

According to our convergence analysis, the loss θ_1 and regularizator θ_2 are allowed to be any convex functions. To meet the goal of classification, we replace the least squares loss in (100) by a nonsmooth *hinge loss* $L(\mathbf{x}, \boldsymbol{\xi}) \equiv \max\{0, 1 - l\mathbf{s}^T\mathbf{x}\}$ and the ℓ_1-norm by an Euclidean norm to enforce the maximum margin. The resulting combination is also known as support vector machine (SVM). With the additional graph-guided fusion penalty, we name our formulation *Graph-Guided SVM (GGSVM)*:

$$\begin{aligned} &\min \mathbb{E}_{\boldsymbol{\xi}} L(\mathbf{x}, \boldsymbol{\xi}) + \frac{\gamma}{2}\|\mathbf{x}\|_2^2 + \nu\|\mathbf{y}\|_1 \\ &\text{s.t. } F\mathbf{x} - \mathbf{y} = \mathbf{0}. \end{aligned} \tag{101}$$

Before presenting the penalty term, we first give an algorithmic solution of (101). Applying our stochastic ADMM to GGSVM, we obtain the following updates:

$$\begin{aligned} \mathbf{x}_{k+1} &\leftarrow \arg\min \mathbf{x}^T L'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}) + \gamma\mathbf{x}^T\mathbf{x}_k + \\ &\quad \frac{\beta}{2}\|F\mathbf{x} - \mathbf{y}_k - \boldsymbol{\lambda}_k/\beta\|_2^2 + \frac{\|\mathbf{x} - \mathbf{x}_k\|_2^2}{2\eta_{k+1}}, \\ \mathbf{y}_{k+1} &\leftarrow \arg\min \nu\|\mathbf{y}\|_1 + \frac{\beta}{2}\|F\mathbf{x}_{k+1} - \mathbf{y} - \boldsymbol{\lambda}_k/\beta\|_2^2, \\ \boldsymbol{\lambda}_{k+1} &\leftarrow \boldsymbol{\lambda}_k - \beta(F\mathbf{x}_{k+1} - \mathbf{y}_{k+1}). \end{aligned} \tag{102}$$

Without the graph-guided regularization, the stochastic ADMM becomes exactly the same as the classic stochastic gradient descent (SGD): $\mathbf{x}_{k+1} \leftarrow \arg\min \mathbf{x}^T L'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}) + \gamma\mathbf{x}^T\mathbf{x}_k + \frac{\|\mathbf{x}-\mathbf{x}_k\|_2^2}{2\eta_{k+1}}$.

The first two updates of (102) have close-forms:

$$\begin{aligned}\mathbf{x}_{k+1} \leftarrow & \left(\frac{I}{\eta_{k+1}} + \beta F^T F\right)^{-1} \Big[F^T(\beta \mathbf{y}_k + \boldsymbol{\lambda}_k) \\ & + (1/\eta_{k+1} - \gamma)\mathbf{x}_k - L'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\Big], \\ \mathbf{y}_{k+1} \leftarrow & S_{\frac{\nu}{\beta}}\left(F\mathbf{x}_{k+1} - \frac{\boldsymbol{\lambda}_k}{\beta}\right).\end{aligned} \tag{103}$$

Note that this simple $\mathbf{x}$-update is exactly the benefit that stochastic ADMM brings. In contrast, neither the classic ADMM nor its variants have closed-forms due to the nonseparable form of the hinge loss.

In each $\mathbf{x}$-update of (103), due to the time-varying η_{k+1}, one has to solve a symmetric linear system with a different system matrix. This can be carried out using standard methods, e.g. conjugate gradient, where the sparsity of $F^T F$ can help in reducing the time complexity. However, for large-scale problems we can remove this computational burden completely by replacing η_{k+1} with a fixed η_t, if we want to run t iterations. This indeed leads to a convergent algorithm, although the proof is not shown in Section 3.3 due to limited space. By this means we only need to solve the linear system *once*, and save the result for successive iterations.

The data is the publicly available 20newsgroups dataset[3], which contains binary occurrences of 100 popular words counted from $16,242$ newsgroup postings. On the top level of these postings are 4 main categories: computer, recreation, science and talks. We are interested in a multi-class classification task: to predict the category that a posting belongs to. We split the original data into a training set and a testing set. In each posting category, 80% postings are used for training and the rest 20% for testing. We use the one-vs-rest scheme for the multi-class classification.

The graphical structures we want to explore are the dependencies among these 100 words. Specifically, if two words i and j are strongly dependent, the difference between x_i and x_j in the linear predictor $\mathbf{x} \in \mathbb{R}^{100}$ should be penalized. In order to obtain F, we use the *sparse inverse covariance selection* (Banerjee et al. [2008]) (also known as *graphical lasso* (Friedman and Tibshirani [2007], Boyd et al. [2010])) and determine the sparsity

[3] `http://www.cs.nyu.edu/~roweis/data.html`

pattern of the inverse covariance matrix Σ^{-1}. By properly thresholding the components of Σ^{-1} to 0 and 1, we obtain the affinity matrix of $\mathcal{G}$ and plot the relations of these 100 words accordingly in Fig. 20. For simplicity, we take all the weights in F to be 1 and -1 whenever there is an edge.

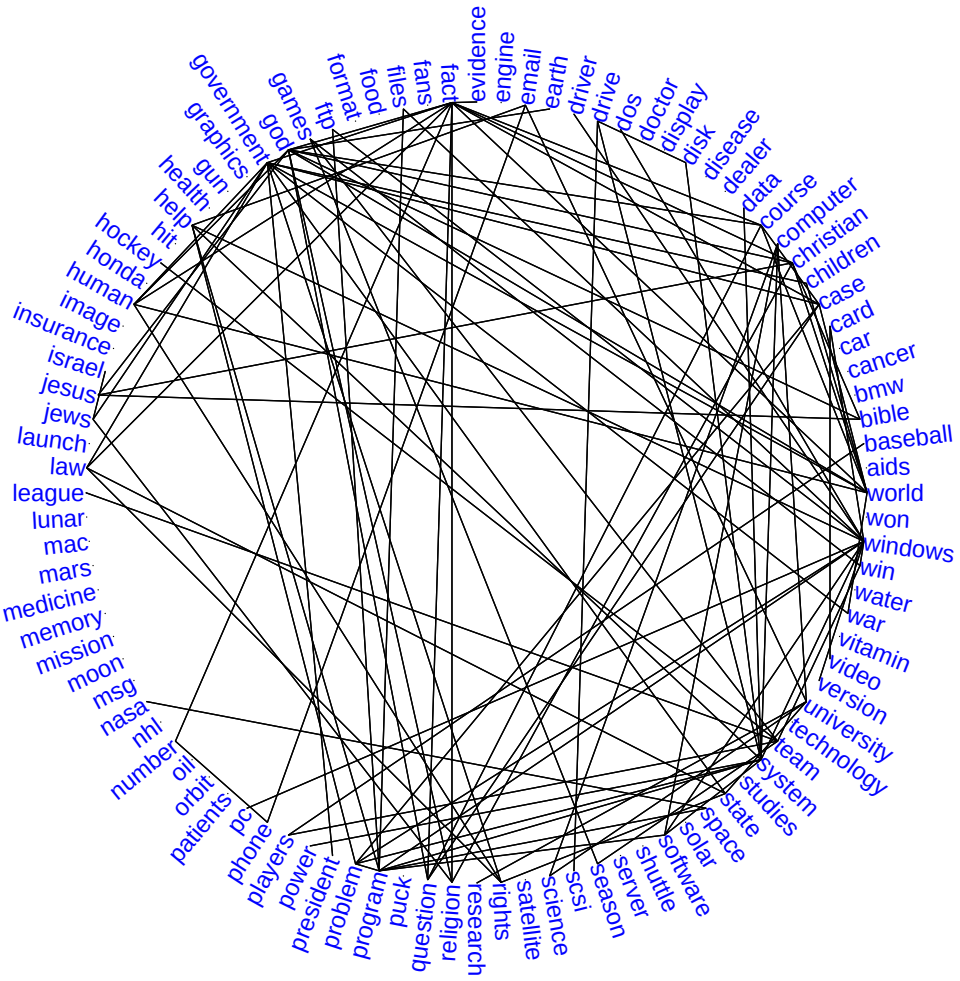

Figure 20: Graph of relations among 100 popular words in 20newsgroups dataset.

We compare the prediction accuracies with and without graphical regularization. Fig. 21 shows the experimental results. The x-axis stands for the number of epochs for stochastic algorithms. For this dataset, each epoch means $12,994$ iterations. We calculate the mean and the standard deviation of all the accuracies based on 10 runs of experiments under the same setting. This figure clearly indicates that GGSVM outperforms the classical SVM consistently in every setting. After a single epoch, which corresponds to 1 iteration for the

deterministic ADMM, the prediction accuracy is already very close to the best. This is a further evidence for the efficiency of our stochastic ADMM.

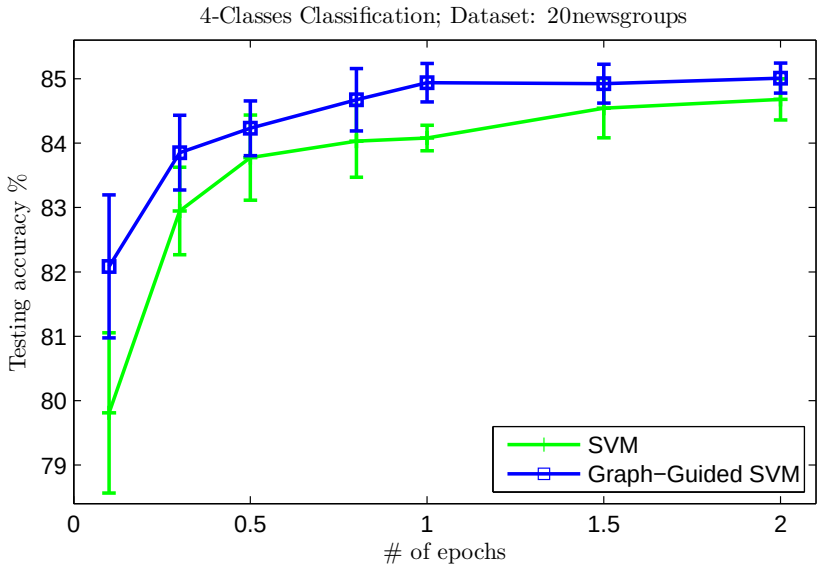

Figure 21: Accuracies for multi-class classification.

3.6 Conclusions of this Chapter

In this paper, we have proposed the stochastic setting for ADMM along with our stochastic linearized ADMM algorithm. As a benefit of the first-order approximation on the stochastic function, our algorithm is applicable to a very broad class of problems even with functions that have no closed-form solution to the subproblem of minimizing the augmented θ_1. We have also established convergence rates under various structural assumptions of θ_1: $O(1/\sqrt{t})$ for convex functions and $O(\log t/t)$ for strongly convex functions. We are working on integrating Nesterov's optimal first-order methods (Nesterov [2004]) to our algorithm, which will help in achieving optimal convergence rates. More interesting and challenging applications will be carried out in our future work.

3.7 Appendix

3.7.1 Proof of Theorem 1

Proof. (i). Invoking convexity of $\theta_1(\cdot)$ and $\theta_2(\cdot)$ and the monotonicity of operator $F(\cdot)$, we have $\forall \mathbf{w} \in \mathcal{W}$:

$$\begin{aligned}
&\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}) + (\bar{\mathbf{w}}_t - \mathbf{w})^T F(\bar{\mathbf{w}}_t) \\
&\leq \frac{1}{t}\sum_{k=1}^{t}\Big[\theta_1(\mathbf{x}_{k-1}) + \theta_2(\mathbf{y}_k) - \theta(\mathbf{u}) + (\mathbf{w}_k - \mathbf{w})^T F(\mathbf{w}_k)\Big] \\
&= \frac{1}{t}\sum_{k=0}^{t-1}\Big[\theta_1(\mathbf{x}_k) + \theta_2(\mathbf{y}_{k+1}) - \theta(\mathbf{u}) + (\mathbf{w}_{k+1} - \mathbf{w})^T F(\mathbf{w}_{k+1})\Big]
\end{aligned} \tag{104}$$

Applying Lemma 9 at the optimal solution $(\mathbf{x}, \mathbf{y}) = (\mathbf{x}_*, \mathbf{y}_*)$, we can derive from (104) that, $\forall \boldsymbol{\lambda}$

$$\begin{aligned}
&\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + (\bar{\mathbf{x}}_t - \mathbf{x}_*)^T(-A^T\bar{\boldsymbol{\lambda}}_t) + (\bar{\mathbf{y}}_t - \mathbf{y}_*)^T(-B^T\bar{\boldsymbol{\lambda}}_t) + (\bar{\boldsymbol{\lambda}}_t - \boldsymbol{\lambda})^T(A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}) \\
&\overset{(80)}{\leq} \frac{1}{t}\sum_{k=0}^{t-1}\left[\frac{\eta_{k+1}\|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2}{2} + \frac{1}{2\eta_{k+1}}\left(\|\mathbf{x}_k - \mathbf{x}_*\|^2 - \|\mathbf{x}_{k+1} - \mathbf{x}_*\|^2\right) + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_* - \mathbf{x}_k\rangle\right] \\
&\quad + \frac{1}{t}\left(\frac{\beta}{2}\|A\mathbf{x}_* + B\mathbf{y}_0 - \mathbf{b}\|^2 + \frac{1}{2\beta}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_0\|^2\right) \\
&\leq \frac{1}{t}\sum_{k=0}^{t-1}\left[\frac{\eta_{k+1}\|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2}{2} + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_* - \mathbf{x}_k\rangle\right] + \frac{1}{t}\left(\frac{D_{\mathcal{X}}^2}{2\eta_t} + \frac{\beta}{2}D_{\mathbf{y}_*,B}^2 + \frac{1}{2\beta}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_0\|_2^2\right)
\end{aligned} \tag{105}$$

The above inequality is true for all $\boldsymbol{\lambda} \in \mathbb{R}^m$, hence it also holds in the ball $\mathcal{B}_0 = \{\boldsymbol{\lambda} : \|\boldsymbol{\lambda}\|_2 \leq \rho\}$. Combing with the fact that the optimal solution must also be feasible, it follows that

$$\begin{aligned}
&\max_{\boldsymbol{\lambda}\in\mathcal{B}_0}\Big\{\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + (\bar{\mathbf{x}}_t - \mathbf{x}_*)^T(-A^T\bar{\boldsymbol{\lambda}}_t) + (\bar{\mathbf{y}}_t - \mathbf{y}_*)^T(-B^T\bar{\boldsymbol{\lambda}}_t) + (\bar{\boldsymbol{\lambda}}_t - \boldsymbol{\lambda})^T(A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b})\Big\} \\
&= \max_{\boldsymbol{\lambda}\in\mathcal{B}_0}\Big\{\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \bar{\boldsymbol{\lambda}}_t^T(A\mathbf{x}_* + B\mathbf{y}_* - b) - \boldsymbol{\lambda}^T(A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b})\Big\} \\
&= \max_{\boldsymbol{\lambda}\in\mathcal{B}_0}\Big\{\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) - \boldsymbol{\lambda}^T(A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b})\Big\} \\
&= \theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\|_2
\end{aligned} \tag{106}$$

Taking an expectation over (106) and using (105) we have:

$$\begin{aligned}
&\mathbb{E}\left[\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\|_2\right] \\
&\leq \mathbb{E}\left[\frac{1}{t}\sum_{k=0}^{t-1}\left(\frac{\eta_{k+1}\|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2}{2} + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_* - \mathbf{x}_k\rangle\right) + \frac{1}{t}\left(\frac{D_{\mathcal{X}}^2}{2\eta_t} + \frac{\beta}{2}D_{\mathbf{y}_*,B}^2\right)\right] \\
&\quad + \mathbb{E}\left[\max_{\boldsymbol{\lambda}\in\mathcal{B}_0}\left\{\frac{1}{2\beta t}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_0\|_2^2\right\}\right] \\
&\leq \frac{1}{t}\left(\frac{M^2}{2}\sum_{k=1}^{t}\eta_k + \frac{D_{\mathcal{X}}^2}{2\eta_t}\right) + \frac{\beta D_{\mathbf{y}_*,B}^2}{2t} + \frac{\rho^2}{2\beta t} + \frac{1}{t}\sum_{k=0}^{t-1}\mathbb{E}\left[\langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_* - \mathbf{x}_k\rangle\right] \\
&= \frac{1}{t}\left(\frac{M^2}{2}\sum_{k=1}^{t}\eta_k + \frac{D_{\mathcal{X}}^2}{2\eta_t}\right) + \frac{\beta D_{\mathbf{y}_*,B}^2}{2t} + \frac{\rho^2}{2\beta t} \\
&\leq \frac{\sqrt{2}D_{\mathcal{X}}M}{\sqrt{t}} + \frac{\beta D_{\mathbf{y}_*,B}^2}{2t} + \frac{\rho^2}{2\beta t}
\end{aligned}$$

In the second last step, we use the fact that $\mathbf{x}_k$ is independent of $\boldsymbol{\xi}_{k+1}$, hence

$$\mathbb{E}_{\boldsymbol{\xi}_{k+1}|\boldsymbol{\xi}_{[1:k]}}\langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_* - \mathbf{x}_k\rangle = \left\langle \mathbb{E}_{\boldsymbol{\xi}_{k+1}|\boldsymbol{\xi}_{[1:k]}}\boldsymbol{\delta}_{k+1}, \mathbf{x}_* - \mathbf{x}_k\right\rangle = 0.$$

(ii) From the steps in the proof of part (i), it follows that,

$$\begin{aligned}
&\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\| \\
&\leq \frac{1}{t}\sum_{k=0}^{t-1}\frac{\eta_{k+1}\ \|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2}{2} + \frac{1}{t}\sum_{k=0}^{t-1}\langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_* - \mathbf{x}_k\rangle + \frac{1}{t}\left(\frac{D_{\mathcal{X}}^2}{2\eta_t} + \frac{\beta}{2}D_{\mathbf{y}_*,B}^2 + \frac{\rho^2}{2\beta}\right) \quad (107) \\
&\equiv A_t + B_t + C_t
\end{aligned}$$

Note that random variables A_t and B_t are dependent on $\boldsymbol{\xi}_{[t]}$.

Claim 1. *For $\Omega_1 > 0$,*

$$Prob\left(A_t \geq (1+\Omega_1)\frac{M^2}{2t}\sum_{k=1}^{t}\eta_k\right) \leq \exp\{-\Omega_1\}. \quad (108)$$

Let $\alpha_k \equiv \frac{\eta_k}{\sum_{k=1}^{t}\eta_k}\ \forall k = 1, \ldots, t$, then $0 \leq \alpha_k \leq 1$ and $\sum_{k=1}^{t}\alpha_k = 1$. Using the fact that $\{\boldsymbol{\delta}_k, \forall k\}$ are independent and applying Assumption 3, one has

$$\begin{aligned}
&\mathbb{E}\left[\exp\left\{\sum_{k=1}^{t}\alpha_k\|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2/M^2\right\}\right] \\
&= \prod_{k=1}^{t}\mathbb{E}\left[\exp\left\{\alpha_k\|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2/M^2\right\}\right] \\
&\leq \prod_{k=1}^{t}\left(\mathbb{E}\left[\exp\left\{\|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2/M^2\right\}\right]\right)^{\alpha_k} \quad \text{(Jensen's Inequality)} \\
&\leq \prod_{k=1}^{t}\left(\exp\{1\}\right)^{\alpha_k} = \exp\left\{\sum_{k=1}^{t}\alpha_k\right\} = \exp\{1\}
\end{aligned}$$

Hence, by Markov's Inequality, we can get

$$\begin{aligned}&\text{Prob}\left(A_t \geq (1+\Omega_1)\frac{M^2}{2t}\sum_{k=1}^{t}\eta_k\right)\\&\leq \exp\left\{-(1+\Omega_1)\right\}\mathbb{E}\left[\exp\left\{\sum_{k=1}^{t}\alpha_k\|\theta_1'(\mathbf{x}_k,\boldsymbol{\xi}_{k+1})\|^2/M^2\right\}\right]\leq \exp\{-\Omega_1\}.\end{aligned}$$

We have therefore proved Claim 1.

Claim 2. *For $\Omega_2 > 0$,*

$$\textit{Prob}\left(B_t > 2\Omega_2\frac{D_{\mathcal{X}}M}{\sqrt{t}}\right)\leq \exp\left\{-\frac{\Omega_2^2}{4}\right\}. \tag{109}$$

In order to prove this claim, we adopt the following facts in Nemirovski's paper (Nemirovski et al. [2009]).

Lemma 10. *Given that for all $k = 1,\ldots,t$, ζ_k is a deterministic function of $\boldsymbol{\xi}_{[k]}$ with $\mathbb{E}\left[\zeta_k|\boldsymbol{\xi}_{[k-1]}\right] = 0$ and $\mathbb{E}\left[\exp\{\zeta_k^2/\sigma_k^2\}|\boldsymbol{\xi}_{[k-1]}\right]\leq\exp\{1\}$, we have*

1. *For $\gamma\geq 0$, $\mathbb{E}\left[\exp\{\gamma\zeta_k\}|\boldsymbol{\xi}_{[k-1]}\right]\leq\exp\{\gamma^2\sigma_k^2\},\forall k=1,\ldots,t$*
2. *Let $S_t=\sum_{k=1}^{t}\zeta_k$, then $\textit{Prob}\{S_t > \Omega\sqrt{\sum_{k=1}^{t}\sigma_k^2}\}\leq\exp\left\{-\frac{\Omega^2}{4}\right\}$.*

Using this result by setting $\zeta_k = \langle\boldsymbol{\delta}_k, \mathbf{x}_* - \mathbf{x}_{k-1}\rangle$, $S_t = \sum_{k=1}^{t}\zeta_k$, and $\sigma_k = 2D_{\mathcal{X}}M, \forall k$, we can verify that $\mathbb{E}\left[\zeta_k|\boldsymbol{\xi}_{[k-1]}\right] = 0$ and

$$\mathbb{E}\left[\exp\{\zeta_k^2/\sigma_k^2\}|\boldsymbol{\xi}_{[k-1]}\right]\leq\mathbb{E}\left[\exp\{D_{\mathcal{X}}^2\|\boldsymbol{\delta}_k\|^2/\sigma_k^2\}|\boldsymbol{\xi}_{[k-1]}\right]\leq\exp\{1\},$$

since $|\zeta_k|^2\leq\|\mathbf{x}_*-\mathbf{x}_{k-1}\|^2\|\boldsymbol{\delta}_k\|^2\leq D_{\mathcal{X}}^2\left(2\|\theta_1'(\mathbf{x}_k,\boldsymbol{\xi}_{k+1})\|^2+2M^2\right)$.

Implementing the above results, it follows that

$$\text{Prob}\left(S_t > 2\Omega_2 D_{\mathcal{X}}M\sqrt{t}\right)\leq\exp\left\{-\frac{\Omega_2^2}{4}\right\}.$$

Since $S_t = tB_t$, we have

$$\text{Prob}\left(B_t > 2\Omega_2\frac{D_{\mathcal{X}}M}{\sqrt{t}}\right)\leq\exp\left\{-\frac{\Omega_2^2}{4}\right\}$$

as desired.

Combining (107), (108) and (109), we obtain

$$\text{Prob}\left(\text{Err}_\rho(\bar{\mathbf{u}}_t) > (1+\Omega_1)\frac{M^2}{2t}\sum_{k=1}^{t}\eta_k + 2\Omega_2\frac{D_{\mathcal{X}}M}{\sqrt{t}} + C_t\right) \le \exp\{-\Omega_1\} + \exp\left\{-\frac{\Omega_2}{4}\right\},$$

where $\text{Err}_\rho(\bar{\mathbf{u}}_t) \equiv \theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\|_2$. Substituting $\Omega_1 = \Omega, \Omega_2 = 2\sqrt{\Omega}$ and plugging in $\eta_k = \frac{D_{\mathcal{X}}}{M\sqrt{2k}}$, we obtain (91) as desired. □

3.7.2 Proof of Theorem 2

Proof. By the strong-convexity of θ_1 we have $\forall \mathbf{x}$:

$$\begin{aligned}
&\theta_1(\mathbf{x}_k) - \theta_1(\mathbf{x}) \le \langle \theta_1'(\mathbf{x}_k), \mathbf{x}_k - \mathbf{x}\rangle - \frac{\mu}{2}\|\mathbf{x} - \mathbf{x}_k\|^2 \\
&= \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x}\rangle + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k\rangle + \langle \theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_k - \mathbf{x}_{k+1}\rangle - \frac{\mu}{2}\|\mathbf{x} - \mathbf{x}_k\|^2.
\end{aligned}$$

Following the same derivations as in Lemma 9 and Theorem 7 (i), we have

$$\begin{aligned}
&\mathbb{E}\left[\theta(\bar{\mathbf{u}}_t) - \theta(\mathbf{u}_*) + \rho\|A\bar{\mathbf{x}}_t + B\bar{\mathbf{y}}_t - \mathbf{b}\|_2\right] \\
&\le \mathbb{E}\left\{\frac{1}{t}\sum_{k=0}^{t-1}\left[\frac{\eta_{k+1}\|\theta_1'(\mathbf{x}_k, \boldsymbol{\xi}_{k+1})\|^2}{2} + \left(\frac{1}{2\eta_{k+1}} - \frac{\mu}{2}\right)\|\mathbf{x}_k - \mathbf{x}_*\|^2 - \frac{\|\mathbf{x}_{k+1} - \mathbf{x}_*\|^2}{2\eta_{k+1}}\right]\right\} \\
&+ \frac{\beta D_{\mathbf{y}_*,B}^2}{2t} + \mathbb{E}\Big[\max_{\boldsymbol{\lambda}\in\mathcal{B}_0}\Big\{\frac{1}{2\beta t}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_0\|_0^2\Big\}\Big] \\
&\le \frac{M^2}{2t}\sum_{k=1}^{t}\frac{1}{\mu k} + \frac{1}{t}\sum_{k=0}^{t-1}\mathbb{E}\left[\frac{\mu k}{2}\|\mathbf{x}_k - \mathbf{x}_*\|^2 - \frac{\mu(k+1)}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_*\|^2\right] + \frac{\beta D_{\mathbf{y}_*,B}^2}{2t} + \frac{\rho^2}{2\beta t} \\
&\le \frac{M^2\log t}{\mu t} + \frac{\mu D_{\mathcal{X}}^2}{2t} + \frac{\beta D_{\mathbf{y}_*,B}^2}{2t} + \frac{\rho^2}{2\beta t}.
\end{aligned}$$

□

3.7.3 Proof of Theorem 3

Proof. The Lipschitz smoothness of θ_1 implies that $\forall k \ge 0$:

$$\begin{aligned}
\theta_1(\mathbf{x}_{k+1}) &\le \theta_1(\mathbf{x}_k) + \langle \nabla\theta_1(\mathbf{x}_k), \mathbf{x}_{k+1} - \mathbf{x}_k\rangle + \frac{L}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 \\
&\overset{(76)}{=} \theta_1(\mathbf{x}_k) + \langle \nabla\theta_1(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x}_k\rangle - \langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_{k+1} - \mathbf{x}_k\rangle + \frac{L}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2.
\end{aligned}$$

It follows that $\forall \mathbf{x} \in \mathcal{X}$:

$$
\begin{aligned}
&\theta_1(\mathbf{x}_{k+1}) - \theta_1(\mathbf{x}) + \left\langle \mathbf{x}_{k+1} - \mathbf{x}, -A^T \boldsymbol{\lambda}_{k+1} \right\rangle \\
&\leq \theta_1(\mathbf{x}_k) - \theta_1(\mathbf{x}) + \langle \nabla \theta_1(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x}_k \rangle - \langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_{k+1} - \mathbf{x}_k \rangle \\
&\quad + \frac{L}{2} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 + \left\langle \mathbf{x}_{k+1} - \mathbf{x}, -A^T \boldsymbol{\lambda}_{k+1} \right\rangle \\
&= \theta_1(\mathbf{x}_k) - \theta_1(\mathbf{x}) + \langle \nabla \theta_1(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x} - \mathbf{x}_k \rangle - \langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_{k+1} - \mathbf{x}_k \rangle + \frac{L}{2} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 \\
&\quad + \left[\langle \nabla \theta_1(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x} \rangle + \left\langle \mathbf{x}_{k+1} - \mathbf{x}, -A^T \boldsymbol{\lambda}_{k+1} \right\rangle \right] \\
&\leq \langle \nabla \theta_1(\mathbf{x}_k), \mathbf{x}_k - \mathbf{x} \rangle + \langle \nabla \theta_1(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x} - \mathbf{x}_k \rangle - \langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_{k+1} - \mathbf{x}_k \rangle + \frac{L}{2} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 \\
&\quad + \left[\langle \nabla \theta_1(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x} \rangle + \left\langle \mathbf{x}_{k+1} - \mathbf{x}, -A^T \boldsymbol{\lambda}_{k+1} \right\rangle \right] \\
&= \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_{k+1} \rangle + \frac{L}{2} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 + \left[\langle \nabla \theta_1(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x} \rangle + \left\langle \mathbf{x}_{k+1} - \mathbf{x}, -A^T \boldsymbol{\lambda}_{k+1} \right\rangle \right] \\
&= \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_{k+1} \rangle + \frac{L}{2} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 + \left\langle \mathbf{x} - \mathbf{x}_{k+1}, \beta A^T B(\mathbf{y}_k - \mathbf{y}_{k+1}) \right\rangle \\
&\quad + \left\langle \nabla \theta_1(\mathbf{x}_k, \boldsymbol{\xi}_{k+1}) + A^T \left[\beta(A\mathbf{x}_{\mathbf{k+1}} + B\mathbf{y}_k - \mathbf{b}) - \boldsymbol{\lambda}_k \right], \mathbf{x}_{k+1} - \mathbf{x} \right\rangle \\
&\overset{(82)}{\leq} \frac{1}{2\eta_{k+1}} \left(\|\mathbf{x} - \mathbf{x}_k\|^2 - \|\mathbf{x} - \mathbf{x}_{k+1}\|^2 \right) - \frac{1/\eta_{k+1} - L}{2} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 \\
&\quad + \left\langle \mathbf{x} - \mathbf{x}_{k+1}, \beta A^T B(\mathbf{y}_k - \mathbf{y}_{k+1}) \right\rangle + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_{k+1} \rangle .
\end{aligned}
$$

The last inner product can be bounded as below using Young's inequality, given that $\eta_{k+1} \leq \frac{1}{L}$:

$$
\begin{aligned}
&\langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_{k+1} \rangle \\
&= \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k \rangle + \langle \boldsymbol{\delta}_{k+1}, \mathbf{x}_k - \mathbf{x}_{k+1} \rangle \\
&\leq \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2\left(1/\eta_{k+1} - L\right)} \|\boldsymbol{\delta}_{k+1}\|^2 + \frac{1/\eta_{k+1} - L}{2} \|\mathbf{x}_k - \mathbf{x}_{k+1}\|^2 .
\end{aligned}
$$

Combining this with inequalities (142,87) and (88), we can get a similar statement as that of Lemma 9:

$$
\begin{aligned}
&\theta(\mathbf{u}_{k+1}) - \theta(\mathbf{u}) + (\mathbf{w}_{k+1} - \mathbf{w})^T F(\mathbf{w}_{k+1}) \leq \frac{\|\boldsymbol{\delta}_{k+1}\|^2}{2(1/\eta_{k+1} - L)} \\
&+ \frac{1}{2\eta_{k+1}} \left(\|\mathbf{x}_k - \mathbf{x}\|^2 - \|\mathbf{x}_{k+1} - \mathbf{x}\|^2 \right) + \frac{\beta}{2} \left(\|A\mathbf{x} + B\mathbf{y}_k - \mathbf{b}\|^2 - \|A\mathbf{x} + B\mathbf{y}_{k+1} - \mathbf{b}\|^2 \right) \\
&+ \langle \boldsymbol{\delta}_{k+1}, \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2\beta} \left(\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_k\|_2^2 - \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}\|_2^2 \right) .
\end{aligned}
$$

The rest of the proof are essentially the same as Theorem 7 (i), except that we use the new definition of $\bar{\mathbf{u}}_k$ in (93). □

CHAPTER IV

STOCHASTIC KERNEL MACHINES

4.1 Introduction

In this chapter we extend the idea of stochastic programming to kernelized support vector machines (SVM) (Vapnik [1982]), one of the most popular nonlinear discriminative learning methods that can achieve good generalization. SVM can be used for a variety of learning problems, such as classification (Vapnik [1982]), ranking (Joachims [2002]), regression (Smola and Scholkopf [1998]), quantile estimation (Scholkopf et al. [2001]). The focus of this chapter will be on the computational aspect of SVMs, especially the scalability of **nonlinear** SVM classifiers to large-scale problems. The proposed algorithms can also be readily used for ranking, regression and other regularized risk minimization problems.

Training SVM classifiers can be formulated as a minimization

$$\min_{\mathbf{w}\in\mathbb{H}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} L\left(\mathbf{w};(\mathbf{x}_i,y_i)\right) \tag{110}$$

where L is the loss function and $\mathbb{H}$ is a vector space. The parameter λ is used as a trade-off between the squared 2-norm regularizer and the empirical risk. The training set is denoted as $\{(\mathbf{x}_i,y_i)\}_{i=1}^N$ with samples $\mathbf{x}_i\in\mathbb{R}^d$ and corresponding labels $y_i\in\{-1,1\}$. A popular loss function often chosen is the hinge loss: $L\left(\mathbf{w},b;(\mathbf{x}_i,y_i)\right)=\max\left\{0,1-y_i\left(\langle\mathbf{w},\phi(\mathbf{x}_i)\rangle+b\right)\right\}$, which is a convex surrogate of the $0-1$ loss. Note that there are many alternative convex loss functions available for classification tasks, such as the squared hinge loss, log-likelihood loss, Huber loss and its variants. The *squared hinge loss* is adopted in this chapter.

The prediction function can be expressed as: $\text{sign}\left(h(\mathbf{x})\right)=\text{sign}\left(\langle\mathbf{w},\phi(\mathbf{x})\rangle+b\right)$. Here $\phi(\cdot)$ is a mapping from $\mathbb{R}^d$ to a feature space $\mathbb{H}$ which is often chosen as a kernel-induced Hilbert space equipped with an inner product $\langle\cdot,\cdot\rangle$. The kernel function is $k(\mathbf{x}_i,\mathbf{x}_j)=\langle\phi\left(\mathbf{x}_i\right),\phi\left(\mathbf{x}_j\right)\rangle$. The $N\times N$ kernel matrix is denoted as $K=\{K_{ij}\}$, where $K_{ij}=k(\mathbf{x}_i,\mathbf{x}_j)$. If $\phi\left(\mathbf{x}_i\right)=\mathbf{x}_i$, $h(\mathbf{x})$ is a linear hyperplane in $\mathbb{R}^d$ and it is called a linear SVM, otherwise it is a nonlinear SVM.

Slack variables $\boldsymbol{\xi}$ can be introduced to handle cases where the two classes are not separable. The soft-margin SVM with p^{th} polynomial hinge loss (Cortes and Vapnik [1995]) can then be expressed as

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{H}, b\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^N} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i^p \\ \text{s.t.} \quad & y_i\left(\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b\right) \geq 1-\xi_i,\ \xi_i \geq 0,\ \forall i. \end{aligned} \tag{sP}$$

(sP) is called the *primal problem* of SVM. When $p = 1$, it is often called L1-SVM. When $p = 2$ it is called L2-SVM [1] which corresponds to squared hinge loss.

By introducing Lagrangian multipliers $\boldsymbol{\alpha}$ and using Karush-Kuhn-Tucker (KKT) optimality conditions, the *dual problem* of SVM with (squared) hinge loss can be expressed as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbb{R}^N} \quad & -\frac{1}{2}\boldsymbol{\alpha}^T\left(K\odot\mathbf{y}\mathbf{y}^T\right)\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{1} \\ \text{s.t.} \quad & 0\leq\alpha_i\leq U,\ i=1,\ldots,N,\ \mathbf{y}^T\boldsymbol{\alpha}=0, \end{aligned} \tag{sD}$$

where the Hadamard product $K\odot\mathbf{y}\mathbf{y}^T = \{y_iy_jK_{ij}\}$, and $\mathbf{1}$ is a vector of all ones. For L1-SVM, $U = C$, and $U = \infty$ for L2-SVM.

Historically, a majority of previous SVM solvers deal with (sD), due to the fact that the nonlinear mapping $\phi(\cdot)$ in (sP) is hard to handle and its constraints are complex, while (sD) is a standard quadratic programming (QP) problem with box and equality constraints. Nonetheless solving (sD) using general QP solvers has not been widely adopted for large-scale problems, since no matter how easy or difficult the problem is, the kernel matrix K is always dense. For instance, the interior point method needs $O(dN^2)$ memory to store K and $O(N^3)$ time for matrix inversions, and both are prohibitive even for small problems like $N = 10^3$. The low-rank approximation method (Smola and Scholkopf [2000]) has been proposed to tackle this problem.

Another vein of solving SVMs are chunking and decomposition methods (Vapnik [1982], Osuna et al. [1997], Platt [1999]). The main idea is to optimize over a small working set B where $|B| \ll N$ and update this set after each iteration, while the α_is of the rest of the

[1] Please notice the difference between L2-SVM (squared hinge loss) and SVM with 2-norm regularization term (any loss).

samples remain unchanged. How to choose B is crucial for this class of methods. Sequential minimal optimization (SMO) (Platt [1999]) takes $|B| = 2$. The two samples are chosen as the pair that violates the complementarity conditions the most. Although there is no rate of convergence analysis for SMO, yet empirically its worst-case time complexity is at least $O(N^{2.3})$ (Platt [1999]). Shrinking and caching techniques have been proposed for further speed up (Joachims [1999]), and second order information has been used for working set selection (Fan et al. [2005]).

Linear SVM solvers are adopted for large-scale problems, especially text classifications (Chang et al. [2008], Hsieh et al. [2008], Joachims [2005], Teo et al. [2007]). Coordinate descent, cutting-plane and bundle methods are utilized to solve the unconstrained problem 110 directly. Since there are only d variables instead of N in nonlinear SVMs, state-of-the-art linear SVM solvers can achieve an $O(dN)$ time complexity or even better. However, generally speaking, the prediction error of a linear classifier could be larger than a nonlinear one, as shown in our experiments.

Stochastic programming techniques have shown to be very efficient for large-scale learning problems, both theoretically and empirically (Saad [1998], Kivinen et al. [2004], Zhang [2004], Shalev-Shwartz et al. [2007]). For large-scale problems, it is often desired to make a trade-off between computational complexity and the precision of underlying optimization algorithms (Bottou and Bousquet [2008]). Stochastic methods, despite of its slow rate of convergence compared with batch methods, can make each iteration very cheap. Hence if only approximate solutions are desired, as in large-scale learning problems, stochastic methods can be much faster than batch methods. This is an important motivation of our work. Among many previous work, *stochastic approximation* (Kushner and Yin [2003]) is often used for online/stochastic convex optimizations. The celebrated stochastic gradient descent (SGD) $\mathbf{w}_{t+1} = \Pi(\mathbf{w}_t - \eta \nabla f(\mathbf{w}, (\mathbf{x}_t, y_t))|_{\mathbf{w}=\mathbf{w}_t})$ is a most popular example (Bottou and LeCun [2005]), where η is a learning rate, $\nabla f(\mathbf{w}, (\mathbf{x}_t, y_t))$ is a noisy approximation of the true gradient. In many cases it only involves a single training sample. $\Pi()$ is a projection on the feasible set of $\mathbf{w}$. The $O(d)$ memory requirement of SGD makes it perfect for large-scale online learning scenarios. Second order information is also used in more recent

work (Schraudolph et al. [2007]). Note that most of these work solve linear SVMs via the primal problem (110) since it is an unconstrained problem. Thus there is no need to do the extra projection $\Pi()$ which could be even more expensive.

Inspired by the core vector machine (Tsang et al. [2005]) and its relations with sparse greedy methods (Clarkson [2008]), in this chapter we propose a family of stochastic nonlinear SVM solvers which solve L2-SVMs in the dual problems. These new algorithms are based on a simple deterministic constrained optimization method: the *Frank-Wolfe method (FW)* (Frank and Wolfe [1956]), hence they are named *stochastic Frank-Wolfe algorithms (SFW)*. Unlike stochastic approximation algorithms such as SGD and its variants, SFW has the flavor of *sample average approximation* (Shapiro et al. [2009]) which is another vein of stochastic programming.

With a slight modification of the primal problem (sP), an alternative dual problem can be formulated as a simplex constrained QP problem which can be solved efficiently by SFW with time complexity $O\left(d\frac{Q^2}{\epsilon^2}\right)$, where $\epsilon = f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha})$, and Q is a constant explained in Theorem 11. A modified algorithm using SFW with "away steps" (Wolfe [1970]) is used to improve the performance of SFW.

4.2 The Frank-Wolfe Method

The Frank-Wolfe method (FW), also known as the conditional gradient method (Bertsekas [1999]), is a simple and classic first order feasible direction method. It was among the earliest methods that solve convex problems with a continuously differentiable objective function and linear constraints:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}). \tag{111}$$

Although in its original form, FW has only a sublinear convergence (Frank and Wolfe [1956]), a modified version of it can achieve linear convergence (Wolfe [1970]). Besides, the computations in each iteration can be made cheap provided that the constraints are linear.

FW generates a sequence of feasible vectors $\{\mathbf{x}^{(k)}\}$ using line search: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^k \mathbf{d}^{(k)}$, where stepsize $\lambda^{(k)} \in [0, 1]$, $\mathbf{d}^{(k)} = \bar{\mathbf{x}} - \mathbf{x}^{(k)}$ is a feasible descent direction satisfying

$\bar{\mathbf{x}} \in \mathcal{X}$ and $(\mathbf{d}^{(k)})^T \nabla f(\mathbf{x}^{(k)}) < 0$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \left(\bar{\mathbf{x}} - \mathbf{x}^{(k)} \right). \tag{112}$$

To search for a best feasible direction, i.e., the best $\bar{\mathbf{x}}$, FW uses the first order Taylor expansion of $f(\mathbf{x})$ and solves the optimization problem:

$$\bar{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathcal{X}} \left(\mathbf{x} - \mathbf{x}^{(k)} \right)^T \nabla f \left(\mathbf{x}^{(k)} \right). \tag{113}$$

The direction search problem (113) needs to be at least no harder than (111) such that FW can be of practical use. This is the case for continuously differentiable $f(\mathbf{x})$ and linear constraints, since (113) is then a linear program that can be solved via the simplex method.

A special case is when the feasible set is a unit simplex: $\triangle := \left\{ \mathbf{x} \mid \sum_{i=1}^{N} x_i = 1, x_i \geq 0, \forall i \right\}$, and (113) is simplified as:

$$\min_{\mathbf{x} \in \triangle} \sum_{i=1}^{N} \frac{\partial f \left(\mathbf{x}^{(k)} \right)}{\partial x_i} \left(x_i - x_i^{(k)} \right). \tag{114}$$

In this case the solution $\bar{\mathbf{x}}$ of (114) has all coordinates equal to 0 except for a single coordinate indexed by p^*, corresponding to the smallest partial derivative, for which $x_{p^*} = 1$. We denote this solution by $\mathbf{e}(p^*)$.

The sublinear convergence of FW can be stated by the following theorem.

Theorem 10 (Frank and Wolfe [1956])**.** *Let $\mathbf{x}^*$ be optimal for (111) and $\{\mathbf{x}^{(k)}\}$ be a sequence generated by FW, then there exists an index K, constants B and ζ such that*

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) \leq \frac{B}{k + \zeta}, \ \forall k \geq K. \tag{115}$$

4.3 Stochastic Frank-Wolfe for SVM

4.3.1 Alternative Formulation of SVM

It might be possible to solve the dual (sD) using FW with the simplex method. However, an alternative formulation of SVM's primal problem can make the dual problem more suitable for FW. The motivation is that, if we can obtain a dual problem that has only simplex constraints, then instead of solving (113), we can look for the explicit solution of the simpler problem (114).

The alternative formulation, which was proposed and successfully utilized by core vector machines (Tsang et al. [2005]), is a variant of L2-SVM. It can be expressed as:

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{H},b\in\mathbb{R},\rho\in\mathbb{R},\boldsymbol{\xi}\in\mathbb{R}^N} \quad & \frac{1}{2}\left(\|\mathbf{w}\|^2 + b^2\right) + C\sum_{i=1}^{N}\xi_i^2 - \nu\rho \\ \text{s.t.} \quad & y_i\left(\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b\right) \geq \rho - \xi_i, \ \forall i. \end{aligned} \tag{aP}$$

Comparing with (sP), (aP) adds a regularization term b^2 and a margin factor $\nu\rho$, where $\nu > 0$ can be removed when we see the dual problem later. By introducing b^2 we can remove the equality constraint $\mathbf{y}^T\boldsymbol{\alpha} = 0$, while by $\nu\rho$ and the usage of the squared hinge loss, we can obtain the simplex constraint.

The dual problem of (aP) can be expressed as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbb{R}^N} \ & -\boldsymbol{\alpha}^T\left(K \odot \mathbf{y}\mathbf{y}^T + \mathbf{y}\mathbf{y}^T + I/2C\right)\boldsymbol{\alpha} \\ \text{s.t.} \ & \boldsymbol{\alpha}^T\mathbf{1} = \nu, \ \alpha_i \geq 0, \ \forall i = 1, \ldots, N, \end{aligned} \tag{aD}$$

where I is an identity matrix. We denote

$$A := K \odot \mathbf{y}\mathbf{y}^T + \mathbf{y}\mathbf{y}^T + I/2C, \tag{116}$$

with component $A_{ij} = y_i y_j\left(K_{ij} + 1\right) + \delta_{ij}/2C$ where $\delta_{pm} = 1$ if $p = m$ and $\delta_{pm} = 0$ otherwise. The objective function of (aD) is homogeneous in $\boldsymbol{\alpha}$, hence we can simply let $\nu = 1$. In this setting, when taking $\xi = 0$, the constraints of (aP) states that the two classes are separated by the margin $2\rho/\|\mathbf{w}\|$. This margin can also be calculated from the KKT complementary slackness conditions.

Since A is positive definite, (aD) is a concave QP problem with unit simplex constraints. This make it in a good situation that FW method can be readily utilized.

Note that in (aP) we do not impose the nonnegativity of ρ explicitly due to the following fact:

Proposition 2. *Imposing $\rho \geq 0$ in (aP) leads to a dual problem that has the same optimal solution as (aD).*

Proof. With $\rho \geq 0$, the new dual problem is $\max_{\boldsymbol{\alpha}\in\mathbb{R}^N} -\boldsymbol{\alpha}^T A\boldsymbol{\alpha}$ s.t. $\boldsymbol{\alpha}^T\mathbf{1} \geq \nu$, $\alpha_i \geq 0$, $i = 1, \ldots, N$. Suppose that it has an optimal solution $\boldsymbol{\alpha}^*$ such that $\boldsymbol{\alpha}^{*T}\mathbf{1} > \nu$, then

$0 < \nu/(\boldsymbol{\alpha}^{*T}\mathbf{1}) < 1$. Define $\tilde{\boldsymbol{\alpha}} = \nu\boldsymbol{\alpha}^*/(\boldsymbol{\alpha}^{*T}\mathbf{1})$. It follows that $-\tilde{\boldsymbol{\alpha}}^T A\tilde{\boldsymbol{\alpha}} \leq -\boldsymbol{\alpha}^{*T} A\boldsymbol{\alpha}^*$ and $-\tilde{\boldsymbol{\alpha}}^T A\tilde{\boldsymbol{\alpha}} = -\left(\nu/(\boldsymbol{\alpha}^{*T}\mathbf{1})\right)^2 \boldsymbol{\alpha}^{*T} A\boldsymbol{\alpha}^* \geq -\boldsymbol{\alpha}^{*T} A\boldsymbol{\alpha}^*$. Hence $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^*$ and $\boldsymbol{\alpha}^{*T}\mathbf{1} = \nu$. □

It should be mentioned that, compared with conventional formulation of L2-SVM, the regularized bias term in the alternative formulation does not affect its performance, as will be observed in our experiments. This has also been shown empirically by many other previous work.

4.3.2 SFW Algorithm

The stochastic Frank-Wolfe algorithm (SFW) solves the dual problem (aD) stochastically. Within the k^{th} iteration, SFW solves the following approximation problem

$$\begin{aligned} &\max_{\boldsymbol{\alpha}\in\mathbb{R}^k} f(\boldsymbol{\alpha}) = -\boldsymbol{\alpha}^T A\boldsymbol{\alpha} \\ &\text{s.t. } \boldsymbol{\alpha}^T\mathbf{1} = 1,\ \alpha_i \geq 0,\ \forall i = 1,\ldots,k, \end{aligned} \tag{aD-k}$$

for a few FW steps. Then the sample size k is increased and a number of additional steps are performed for the updated approximation problem. k is then increased again, a few FW steps are performed. For each iteration, one does not need to solve the approximation problem with a high precision.

In online learning settings, k can be increased as long as new samples are available. In stochastic batch learning settings, when all samples are in a working set and no new sample is available, SFW will proceed along a direction guided by an existed sample in the working set.

Algorithm 4 shows the proposed Stochastic Frank-Wolfe Algorithm (SFW). The time-complexities of the most time-consuming steps are given.

Algorithm 4 SFW

1: $p^*_{(0)} = \text{rand}()$, $q^{(1)} = A_{p^*_{(0)}p^*_{(0)}}$, $\lambda^*_{(0)} = 1$

2: $\boldsymbol{\alpha}^{(1)} = [0, \ldots, 0, 1, 0, \ldots, 0]^T = \mathbf{e}(p^*_{(0)})$, update $g^{(1)}$

3: **for** $k = 1, \ldots$ **do**

4: Sample $\{\mathbf{x}_k, y_k\}$ provided. Calculate $g^{(k)}_k$. // $O(dk)$

5: **if** $g^{(k)}_k \geq \max_{p \in S^{(k-1)}} g^{(k)}_p$ **then**

6: $S^{(k)} \leftarrow S^{(k-1)} \cup \{k\}$

7: **else**

8: $S^{(k)} \leftarrow S^{(k-1)}$

9: **end if**

10: TOWARD // $O(dk)$ or $O(k)$

11: **end for**

12: $b = \mathbf{y}^T \boldsymbol{\alpha}$

Algorithm 5 shows the FW step of Algorithm 4 (line 10). It is named TOWARD since in the next subsection we will introduce an AWAY step.

Algorithm 5 TOWARD

1: $p^*_{(k)} \leftarrow \text{argmax}_{p \in S^{(k)}} g^{(k)}_p$

2: $\lambda^*_{(k)} \leftarrow \text{Clamp}\left\{1 - \frac{g_{p^*_{(k)}}/2 + A_{p^*_{(k)}p^*_{(k)}}}{q^{(k)} + g_{p^*_{(k)}} + A_{p^*_{(k)}p^*_{(k)}}}, [0, 1]\right\}$

3: $q^{(k+1)} \leftarrow \left(1 - \lambda^*_{(k)}\right)^2 q^{(k)} - \lambda^*_{(k)}\left(1 - \lambda^*_{(k)}\right) g_{p^*_{(k)}} + \left(\lambda^*_{(k)}\right)^2 A_{p^*_{(k)}p^*_{(k)}}$

4: **for** $p = 1, \ldots, |S^{(k)}|$ **do**

5: $g^{(k+1)}_p \leftarrow \left(1 - \lambda^*_{(k)}\right) g^{(k)}_p - 2\lambda^*_{(k)} A_{pp^*_{(k)}}$

6: **end for**

7: $\boldsymbol{\alpha}^{(k+1)} \leftarrow \left(1 - \lambda^*_{(k)}\right) \boldsymbol{\alpha}^{(k)} + \lambda^*_{(k)} \mathbf{e}\left(p^*_{(k)}\right)$

We denote the objective function of (aD-k) as $f(\boldsymbol{\alpha}) := -\boldsymbol{\alpha}^T A \boldsymbol{\alpha}$, and the corresponding

gradient as $\mathbf{g}(\boldsymbol{\alpha}) = -2A\boldsymbol{\alpha}$. For $p = 1, \ldots, k$, the p^{th} component of $\mathbf{g}(\boldsymbol{\alpha})$ is calculated as:

$$\begin{aligned} g_p &= -2y_p \sum_{m=1}^{k} y_m \alpha_m \left(K_{pm} + 1 + y_p y_m \delta_{pm}/2C\right) \\ &= -2y_p \sum_{m=1}^{k} y_m \alpha_m \left(K_{pm} + 1\right) - \alpha_p/C. \end{aligned} \tag{117}$$

The algorithm maintains a working set $S^{(k)}$. For initialization, a random sample indexed by $p^*_{(0)}$ is chosen and we set $S^{(0)} = \{p^*_{(0)}\}$, $\boldsymbol{\alpha}^{(1)} = \mathbf{e}(p^*_{(0)})$. The gradient is then initialized using (117).

In the following k^{th} iteration, if the new training sample $\{\mathbf{x}_k, y_k\}$ can provide a better feasible direction than **any** old samples in $S^{(k)}$, this new sample is selected as $p^*_{(k)}$ and is included in the new working set $S^{(k+1)}$. Otherwise, the best old sample within $S^{(k)}$ will be selected as $p^*_{(k)}$ for updating $\boldsymbol{\alpha}$ and $\mathbf{g}(\boldsymbol{\alpha})$, and $S^{(k+1)}$ will remains the same as $S^{(k)}$. Therefore the number of indices in the working set $|S^{(k)}| \leq k$, and $\boldsymbol{\alpha}^{(k)}$ has at most k nonzero items.

The search direction $\mathbf{d}^{(k)} = \mathbf{e}(p^*_{(k)}) - \boldsymbol{\alpha}^{(k)}$ starts from the current solution $\boldsymbol{\alpha}^{(k)}$ and points to one of the vertices of the unit simplex. Once this optimal vertex is determined, we can use the limited minimization rule to determine the stepsize $\lambda^*_{(k)}$:

$$\begin{aligned} \lambda^*_{(k)} &= \arg\max_{\lambda} f\left(\boldsymbol{\alpha}^{(k)} + \lambda(\mathbf{e}(p^*_{(k)}) - \boldsymbol{\alpha}^{(k)})\right) \\ &= \arg\min_{\lambda} \lambda^2 \left(\mathbf{e}(p^*_{(k)}) - \boldsymbol{\alpha}^{(k)}\right)^T A \left(\mathbf{e}(p^*_{(k)}) - \boldsymbol{\alpha}^{(k)}\right) + \\ & 2\lambda \left(\mathbf{e}(p^*_{(k)}) - \boldsymbol{\alpha}^{(k)}\right)^T A\boldsymbol{\alpha}^{(k)} + \boldsymbol{\alpha}^{(k)T} A\boldsymbol{\alpha}^{(k)} \\ &= 1 + \frac{\mathbf{e}^T(p^*_{(k)}) A\boldsymbol{\alpha}^{(k)} - A_{p^*_{(k)}p^*_{(k)}}}{\boldsymbol{\alpha}^{(k)T} A\boldsymbol{\alpha}^{(k)} - 2\mathbf{e}^T(p^*_{(k)}) A\boldsymbol{\alpha}^{(k)} + A_{p^*_{(k)}p^*_{(k)}}} \\ &= 1 + \frac{-g_{p^*_{(k)}}/2 - A_{p^*_{(k)}p^*_{(k)}}}{q^{(k)} + g_{p^*_{(k)}} + A_{p^*_{(k)}p^*_{(k)}}}, \end{aligned} \tag{118}$$

where we denote $q^{(k)} := \boldsymbol{\alpha}^{(k)T} A\boldsymbol{\alpha}^{(k)}$. Note that $q^{(k+1)}$ can be calculated by updating from

$q^{(k)}$ rather than starting from scratch:

$$\begin{aligned} q^{(k+1)} &= \left[\boldsymbol{\alpha}^{(k)} + \lambda^*_{(k)}(\mathbf{e}(p^*_{(k)}) - \boldsymbol{\alpha}^{(k)})\right]^T A \\ &\quad \left[\boldsymbol{\alpha}^{(k)} + \lambda^*_{(k)}(\mathbf{e}(p^*_{(k)}) - \boldsymbol{\alpha}^{(k)})\right] \\ &= (1-\lambda^*_{(k)})^2 q^{(k)} + 2\lambda^*_{(k)}(1-\lambda^*_{(k)})\mathbf{e}^T(p^*_{(k)})A\boldsymbol{\alpha}^{(k)} + \\ &\quad (\lambda^*_{(k)})^2 A_{p^*_{(k)}p^*_{(k)}}. \end{aligned} \tag{119}$$

In order to make sure that the new feasible solution remains in the feasible set, we need to clamp $\lambda^*_{(k)}$ in the interval $[0,1]$: if $\lambda^*_{(k)} < 0 \to \lambda^*_{(k)} = 0$, if $\lambda^*_{(k)} > 1 \to \lambda^*_{(k)} = 1$.

Calculations of $g_p^{(k+1)}$ can also be done by updating from $g_p^{(k)}$ in the same manner as (119), as shown in line 5 of Algorithm 5. A practical method for further accelerating line 7 is that, instead of scaling vector $\boldsymbol{\alpha}$ for every iteration, which take $O(k)$ time, we can simply maintain a scaler $c = \prod_k (1-\lambda^*_{(k)})$, and only update the $p^*_{(k)}$-th component of $\boldsymbol{\alpha}^{(k+1)}$ by $\lambda^*_{(k)}\mathbf{e}(p^*_{(k)})/(1-\lambda^*_{(k)})$. This can reduce its time complexity to $O(1)$. Scaler c is multiplied back when the true values of $\boldsymbol{\alpha}$ are needed.

Calculation of the bias term $b = \mathbf{y}^T\boldsymbol{\alpha}$ is directly obtained by taking partial derivative of the Lagrangian of (aP) wrt b to 0.

4.3.3 SFW Algorithm with Away Steps

In addition to "toward steps" as in Eq.(112), "away steps"

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)}\left(\mathbf{x}^{(k)} - \bar{\mathbf{x}}\right) \tag{120}$$

can also be considered. We denote search directions for toward steps as $\mathbf{d}_t^{(k)} := \bar{\mathbf{x}} - \mathbf{x}^{(k)}$ and $\mathbf{d}_a^{(k)} := \mathbf{x}^{(k)} - \bar{\mathbf{x}}$ for away steps.

Introducing away steps is a method that can boost FW to linear convergence (Wolfe [1970]). We can illustrate it using a simplex-constrained example shown in Fig.22, where $\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2$ are vertices of the simplex and the initial solution is x_0. If we only use toward steps, the convergence is pretty slow: $\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_3 \to \cdots$. However, with away steps, the solution converges in merely 3 steps: $\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}^*$, where the last one is an away step: $\mathbf{d}_a = \mathbf{x}_2 - \mathbf{e}_0$.

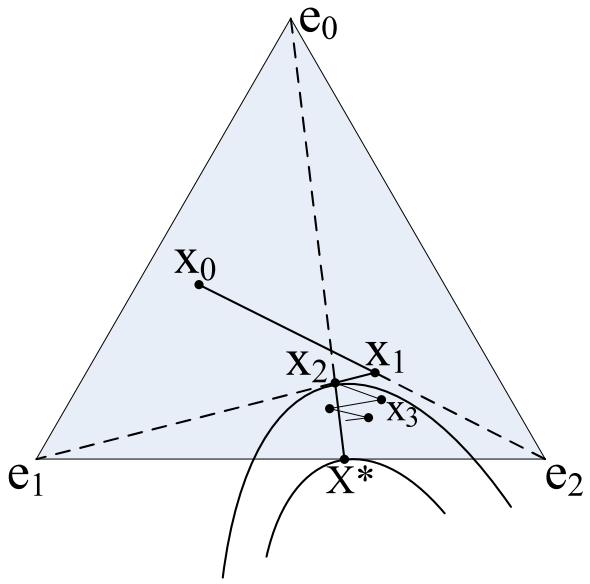

Figure 22: FW with Away Steps.

Applying this method in our dual SVM problem (aD-k), we can use the following to decide whether to make a toward or away step in the k^{th} iteration:

Algorithm 6 DECIDESTEP

1: **if** $\mathbf{d}_t^{(k)T} \nabla f(\boldsymbol{\alpha}^{(k)}) \geq \mathbf{d}_a^{(k)T} \nabla f(\boldsymbol{\alpha}^{(k)})$ **then**
2: TOWARD
3: **else**
4: AWAY
5: **end if**

The AWAY step is given in Algorithm 7. The stepsize is obtained in a similar way as (118). In order to keep the new solution feasible, we need $(1+\lambda)\alpha_{p^*_{(k)}} - \lambda \geq 0$, that is $0 \leq \lambda \leq \alpha_{p^*_{(k)}}/(1-\alpha_{p^*_{(k)}})$.

Algorithm 7 AWAY

1: $p^*_{(k)} \leftarrow \operatorname{argmin}_{p \in S^{(k)}} g_p^{(k)}$

2: $\lambda^*_{(k)} \leftarrow \text{Clamp}\left\{ \frac{g_{p^*_{(k)}}/2 + A_{p^*_{(k)} p^*_{(k)}}}{q^{(k)} + g_{p^*_{(k)}} + A_{p^*_{(k)} p^*_{(k)}}} - 1, \left[0, \frac{\alpha_{p^*_{(k)}}}{1 - \alpha_{p^*_{(k)}}}\right] \right\}$

3: $q^{(k+1)} \leftarrow \left(1 + \lambda^*_{(k)}\right)^2 q^{(k)} + \lambda^*_{(k)} \left(1 + \lambda^*_{(k)}\right) g_{p^*_{(k)}} + \left(\lambda^*_{(k)}\right)^2 A_{p^*_{(k)} p^*_{(k)}}$

4: **for** $p = 1, \ldots, |S^{(k)}|$ **do**

5: $\quad g_p^{(k+1)} \leftarrow \left(1 + \lambda^*_{(k)}\right) g_p^{(k)} + 2\lambda^*_{(k)} A_{p p^*_{(k)}}$

6: **end for**

7: $\boldsymbol{\alpha}^{(k+1)} \leftarrow \left(1 + \lambda^*_{(k)}\right) \boldsymbol{\alpha}^{(k)} - \lambda^*_{(k)} \mathbf{e}\left(p^*_{(k)}\right)$

Replacing TOWARD in Algorithm 4 (line 10) with DECIDESTEP, we obtain a modified SFW algorithm with away steps, named MSFW.

4.4 Convergence Analysis

In this section, we give the rate of convergence of SFW and MSFW, where we use some techniques from (Clarkson [2008]) and (Guelat and Marcotte [1986]). We firstly introduce the *Wolfe dual* and its weak duality (Wolfe [1961]).

Definition 3. *Let f and g_i be concave and continuously differentiable on $\mathbb{R}^N$. The primal problem is*

$$\max \ f(\mathbf{x}) \quad s.t. \ g_i(\mathbf{x}) \geq 0, i = 1, \ldots, m. \tag{121}$$

Its Wolfe dual is

$$\begin{aligned} &\min \ L(\mathbf{x}, u) = f(\mathbf{x}) + \sum_{i=1}^{m} u_i g_i(\mathbf{x}) \\ &s.t. \ \nabla f(\mathbf{x}) + \sum_i u_i \nabla g_i(\mathbf{x}) = 0, \quad u \geq 0 \end{aligned} \tag{122}$$

The following proposition establishes the weak duality of Wolfe dual.

Proposition 3. *Let $q^* = \inf_{u \geq 0} \sup_{\mathbf{x}} L(\mathbf{x}, u)$, $f^* = \sup_{\mathbf{x}}(f(\mathbf{x}))$. Then $q^* \geq f^*$.*

Applying this weak duality to a simplex-constrained concave problem as used in (aD), we have the following result:

Lemma 11. *Let $\boldsymbol{\alpha}^*$ be the optimal solution of (aD). Solving (aD) using SFW algorithm with toward directions $\mathbf{d}_t^{(k)} = \mathbf{e}(p^*_{(k)}) - \boldsymbol{\alpha}^{(k)}$, the following holds for every iteration k:*

$$\mathbf{d}_t^{(k)T} \nabla f(\boldsymbol{\alpha}) \geq f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k)}). \tag{123}$$

Proof. Consider a general unit simplex-constrained concave problem

$$\max_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}),\ s.t.\ \boldsymbol{\alpha} \in \triangle. \tag{124}$$

Introducing Lagrange multipliers y and z_i we obtain its Wolfe dual:

$$\min_{\boldsymbol{\alpha}, y, z_i} f(\boldsymbol{\alpha}) - \boldsymbol{\alpha}^T \nabla f(\boldsymbol{\alpha}) - y$$
$$s.t.\ y + z_i + \nabla f(\boldsymbol{\alpha})_i = 0,\ \forall i.$$

The constraint is equivalent to $-y \geq \max_i \nabla f(\boldsymbol{\alpha})_i$. Taking the smallest feasible $-y = \max_i \nabla f(\boldsymbol{\alpha})_i$, the above dual problem can be rewritten as:

$$\min_{\boldsymbol{\alpha}} \left(f(\boldsymbol{\alpha}) - \boldsymbol{\alpha}^T \nabla f(\boldsymbol{\alpha}) + \max_i \nabla f(\boldsymbol{\alpha})_i \right).$$

Using Proposition 3, it follows that

$$f(\boldsymbol{\alpha}) - \boldsymbol{\alpha}^T \nabla f(\boldsymbol{\alpha}) + \max_i \nabla f(\boldsymbol{\alpha})_i \geq f(\boldsymbol{\alpha}^*). \tag{125}$$

Now replacing the general concave f in (124) with the one in (aD-k), (125) can be rewritten as:

$$f(\boldsymbol{\alpha}^{(k)}) - \boldsymbol{\alpha}^{(k)T} \nabla f(\boldsymbol{\alpha}) + \max_{i \in S^{(k)}} \nabla f(\boldsymbol{\alpha})_i \geq f(\boldsymbol{\alpha}^*_{(k)}) \tag{126}$$

where $\boldsymbol{\alpha}^*_{(k)}$ is the optimal solution of (aD-k).

Since matrix A defined in (116) is symmetric positive definite (s.p.d.), and $\boldsymbol{\alpha}$ for non-working samples are all set to 0, we can decompose the objective function of (aD) as $-\boldsymbol{\alpha}^T A \boldsymbol{\alpha} = -\boldsymbol{\alpha}^T A_k \boldsymbol{\alpha} - \boldsymbol{\alpha}^T (A - A_k) \boldsymbol{\alpha}$, where $-\boldsymbol{\alpha}^T A_k \boldsymbol{\alpha}$ corresponds to the working samples $S^{(k)}$ used for solving (aD-k). Since A_k is s.p.d., using similar techniques as in (Smola and Scholkopf [2000]), we can prove that $A - A_k$ is also s.p.d.. It follows that

$$\begin{aligned} f(\boldsymbol{\alpha}^*_{(k)}) &= -\boldsymbol{\alpha}^T A_k \boldsymbol{\alpha} = -\boldsymbol{\alpha}^T A \boldsymbol{\alpha} + \boldsymbol{\alpha}^T (A - A_k) \boldsymbol{\alpha} \\ &\geq -\boldsymbol{\alpha}^T A \boldsymbol{\alpha} = f(\boldsymbol{\alpha}^*). \end{aligned} \tag{127}$$

Combining (126) and (127), we have

$$\max_{i \in S^{(k)}} \nabla f(\boldsymbol{\alpha})_i - \boldsymbol{\alpha}^{(k)T} \nabla f(\boldsymbol{\alpha}) \geq f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k)})$$

which completes the proof. □

We are now ready to present the rate of convergence for SFW.

Theorem 11. *Algorithm SFW solves (aD) such that there exist a constant $\zeta \geq -1/2$, and for all $k = 1, 2, \ldots$*

$$f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k)}) \leq \frac{Q}{k + \zeta}, \tag{128}$$

where $Q = \sup_k Q_k$ and $Q_k = \mathbf{d}_t^{(k)T} A \mathbf{d}_t^{(k)}$.

Proof. Using the updating rule of SFW for $\boldsymbol{\alpha}$, we have:

$$f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k+1)}) = f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k)}) - \frac{\left[\mathbf{d}_t^{(k)T} A \boldsymbol{\alpha}^{(k)}\right]^2}{Q_k}.$$

Using Lemma 11 we have

$$f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k+1)}) \leq f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k)}) - \frac{\left[f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k)})\right]^2}{Q_k}.$$

Denoting $\beta_k = f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k)})$, it follows that

$$\beta_{k+1} \leq \beta_k - \frac{\beta_k^2}{Q_k} \leq \frac{\beta_k}{1 + \frac{\beta_k}{Q_k}} = \frac{1}{\frac{1}{Q_k} + \frac{1}{\beta_k}} \leq \frac{1}{\frac{1}{Q} + \frac{1}{\beta_k}}.$$

For $k = 1$, denote its stepsize $\lambda_{(1)}^* = \eta$. Then

$$\eta = \lambda_{(1)}^* = \frac{\mathbf{d}_t^{(1)T} \nabla f(\boldsymbol{\alpha})}{2Q_1} \leq 1.$$

Using Lemma 11, $\beta_1 \leq \mathbf{d}_t^{(1)T} \nabla f(\boldsymbol{\alpha}) = 2\eta Q_1 \leq 2\eta Q$. By induction we have:

$$\beta_2 \leq \frac{1}{\frac{1}{Q} + \frac{1}{2\eta Q}} = \frac{Q}{1 + 1/2\eta}, \ldots, \beta_k \leq \frac{Q}{k + (\frac{1}{2\eta} - 1)}.$$

Since $0 \leq \eta \leq 1$, $\zeta = \frac{1}{2\eta} - 1 \geq -1/2$ which completes the proof. □

Theorem 11 means that, to achieve ϵ-accuracy on the dual problem, i.e. $\epsilon = f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha})$, Algorithm 4 needs $t = O(\frac{Q}{\epsilon})$ iterations. It follows that the worst-case time complexity of Algorithm 4 is $\sum_{k=1}^{t} O(dk) = O(dt^2) = O\left(d\frac{Q^2}{\epsilon^2}\right)$.

The convergence analysis for MSFW is more involved. We firstly prove that for k large enough, all FW steps will be away steps.

Lemma 12. *In MSFW, there exist $M > 0$ such that for all $k \geq M$, $\mathbf{d}_t^{(k)T} \nabla f(\boldsymbol{\alpha}^{(k)}) \leq \mathbf{d}_a^{(k)T} \nabla f(\boldsymbol{\alpha}^{(k)})$.*

Proof. See the first part of Theorem 5's proof in (Guelat and Marcotte [1986]). □

The rate of convergence is give below.

Theorem 12. *Algorithm MSFW solves (aD) such that there exists $0 < c < 1$, and for all $k \geq M$,*

$$\frac{f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k+1)})}{f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^{(k)})} \leq c, \tag{129}$$

where M is defined in Lemma 12.

Proof. (scratch) The objective function $f(\boldsymbol{\alpha})$ is Lipschitz continuous and strongly convex, i.e. there exists γ_1 and γ_2 such that

$$\begin{aligned} &\gamma_1 \|\boldsymbol{\alpha}^{(k+1)} - \boldsymbol{\alpha}^{(k)}\|^2 \\ &\leq \left(\boldsymbol{\alpha}^{(k+1)} - \boldsymbol{\alpha}^{(k)}\right)^T \left(\nabla f(\boldsymbol{\alpha}^{(k+1)}) - \nabla f(\boldsymbol{\alpha}^{(k)})\right) \\ &\leq \gamma_2 \|\boldsymbol{\alpha}^{(k+1)} - \boldsymbol{\alpha}^{(k)}\|^2. \end{aligned} \tag{130}$$

The following proof essentially follows the proof of Theorem 2 in (Guelat and Marcotte [1986]). □

4.5 Experimental Results

4.5.1 Datasets

In this section, several real-world datasets from various application domains will be used to evaluate the efficiency of the proposed stochastic algorithms. The first four datasets are at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. The last two are at `http://www.cse.ust.hk/~ivor/cvm.html`. The sizes of these datasets range from small

scale (thousands samples) to mid-large scale (half a million samples). In some datasets, the number of training samples for each class are fairly unbalanced. Among these datasets, "w3a" is text data which is of relatively high dimension but very sparse. Table 3 gives the details of them.

4.5.2 SVM Solvers for Comparison

The proposed algorithms are SFW: stochastic Frank-Wolfe and MSFW: modified stochastic Frank-Wolfe with away steps. We compare them with several existing methods described as below.

- SMO-L1: sequential minimal optimization for L1-SVM (Platt [1999]). This is one of the most popular solver for nonlinear SVM classifications with hinge loss.
- SMO-L2: sequential minimal optimization for L2-SVM. The squared hinge loss is adopted. The implementation of its training algorithm is essentially the same as SMO-L1. The only differences are: the box constrains of SMO-L1 is replaced with an nonnegativity constraint; an additional $1/2C$ term is added in each kernel calculation.
- SMO-shrink: SMO with shrinking technique (Joachims [1999]). The shrinking technique is based on the heuristics that some samples that have reached their bounds tend to remain unchanged in following iterations, thus can be temporarily removed from the working set. In our implementations, the working set is shrunk after every 1000 iterations.
- SMO-wss2: SMO with shrinking technique and second order information for working set selection (Fan et al. [2005]). It is the state-of-the-art nonlinear SVM solver used in LibSVM.
- SGD-linear: stochastic gradient descent for online/stochastic linear SVM with hinge loss (Zhang [2004]). The learning rate is chosen as $1/k$ for the k^{th} iteration.
- SGD-kernel: kernelized stochastic gradient descent for online/stochastic nonlinear SVM with hinge loss, also known as NORMA (Kivinen et al. [2004]). The learning rate is chosen as $1/k$ for the k^{th} iteration. The margin parameter ρ is set to 1,

Dataset Name	# of Classes	# of Training Samples (N)	# of Testing Samples	# of Dim. (d)	Density	Comments
svmguide1	2	3,089	4,000	4	100%	real values; class 1: 64.7%
w3a	2	4,912	44,837	300	3.88%	integer values; class -1: 97.1%
a9a	2	32,561	16,281	123	11.28%	integer values; class -1: 75.9%
ijcnn1	2	35,000	91,701	22	59.09%	integer and real values; class -1: 90.2%
usps01	2	266,079	75,383	676	14.95%	real values; class 1: 54.3%
covertype	2	522,910	58,102	54	22.00%	integer values; class -1: 51.2%

Table 3: Real-world datasets from various application domains.

Dataset	σ	C for L1-SVM	C for L2-SVM
svmguide1	20	1	0.4
w3a	4	18	5
a9a	10	1.5	0.5
ijcnn1	0.61	7	5
usps01	10	10	50
covertype	100	10000	1000

Table 4: Parameters used for L1-SVM and L2-SVM.

which corresponds to hinge loss.

- Pegasos: primal estimated sub-gradient solver for online/stochastic linear SVM with hinge loss (Shalev-Shwartz et al. [2007]).

4.5.3 Parameters and Implementation Issues

For nonlinear methods, we choose Gaussian radial basis function kernels

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)\right)$$

. The kernel bandwidth σ and the parameter C in (sD) and (aD) are chosen via grid searches, and the results with best testing accuracies are reported. Note that the C parameters for L1-SVM and L2-SVM are generally different. Table 4 shows the different parameter combinations used for our datasets.

For stopping criteria in batch learning tasks, all the above SMO-type algorithms will stop when the "gap" $\min_{i\in I_{\text{up}}} y_i g_i - \max_{i\in I_{\text{down}}} y_i g_i$ is less than 0.001, where $I_{\text{up}} = \{i|\alpha_i < C \text{ if } y_i = 1, \text{ or } \alpha_i > 0 \text{ if } y_i = -1\}$ and $I_{\text{down}} = \{i|\alpha_i > 0 \text{ if } y_i = 1, \text{ or } \alpha_i < C \text{ if } y_i = -1\}$. This tolerance is the same as the default stopping tolerance used in LibSVM. For all the stochastic algorithms, they will stop after running for 1 or 2 epochs of iterations, where an epoch stands for N iterations.

We do not include the caching technique in SMO or our SFW implementations for the purpose of fair comparisons, since its performance varies for different datasets.

All the methods are implemented in C++, and all the data are in double-precision. The experiments are carried out on a workstation with Xeon 3.00GHz CPU, 8 GB RAM and 64-bit Linux system. In the following results, all the times shown are training time measured in CPU time, excluding the time spent for data loading and prediction.

4.5.4 Comparisons on Convergence

In this section, we will compare performances of SFW and MSFW to other nonlinear SVM solvers, namely SMO-L1, SMO-L2, SMO-shrink, SMO-wss2 and SGD-kernel. The primal/dual values are expensive to calculate, so instead of evaluating the convergence on objective

values, we show the convergence history in testing accuracy, which is indeed what we care about in practice.

For each dataset and solver, we run several trials with different number of iterations. After each trial we use the testing sets to calculate testing accuracies. Fig.23∼28 show the convergence histories of testing accuracies, against training time.

In our implementations of stochastic algorithms, in order to mimic online learning scenarios, each dataset is randomly permuted before feeding to the solvers. Hence these plots can also be used to evaluate the performance of SFW and MSFW in online learning tasks.

Note that SGD-kernel is not shown in Fig. 24 and 26, since our experiments show that the method's testing accuracy is always the portion of the major class for datasets "w3a" and "ijcnn1", in which the two classes are high unbalanced.

It can be observed that stochastic Frank-Wolfe methods can quickly reach acceptable accuracies in the very early iterations. Although this is a general benefit of stochastic algorithms, SFW and MSFW still converge much faster than the stochastic SGD-kernel.

Although MSFW is proved to have a faster convergence rate than SFW, this theoretical advantage is not very prominent in our experiments. This deserves further investigation.

4.5.5 Comparisons on Batch Learning Tasks

In this section, we will compare the performances of nonlinear solvers for batch learning. In batch learning tasks, normally a pre-defined stopping criterion should be set to terminate the optimization process.

As we will shown in Table 5, running SFW or MSFW for 1 or 2 epochs is enough to obtain very good testing accuracies. As a comparison, for SMO-type algorithms, we use the stopping criterion as discussed in section 4.5.3. This criterion is widely adopted by many software packages, e.g. LibSVM and SVMLight.

The testing accuracies in the left half of Table 5 show that all the proposed stochastic Frank-Wolfe algorithms achieve comparable or even higher accuracies than the rest of the L1/L2-SVM solvers. The right half shows that the proposed algorithms are consistently faster than SMO-type methods. Table 6 shows the number of iterations and number of

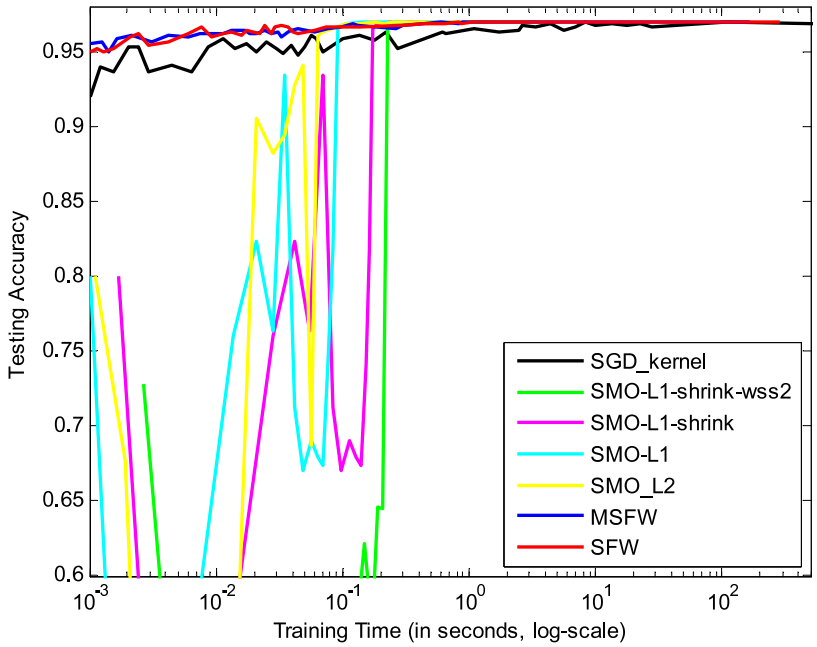

Figure 23: Dataset: svmguide1

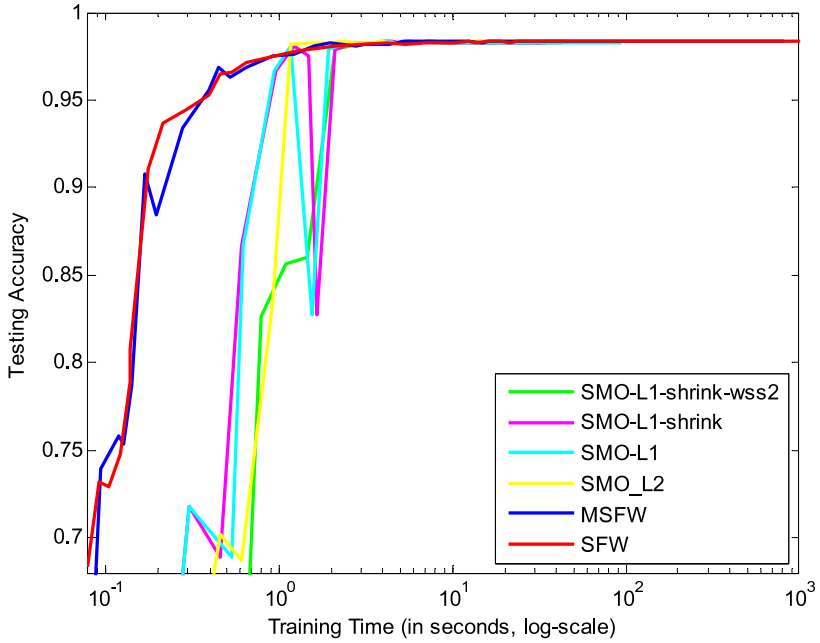

Figure 24: Dataset: w3a

support vectors. We can see that the Frank-Wolfe methods have generally larger amount of support vectors than L1-SVM solvers. This is due to the difference between hinge loss

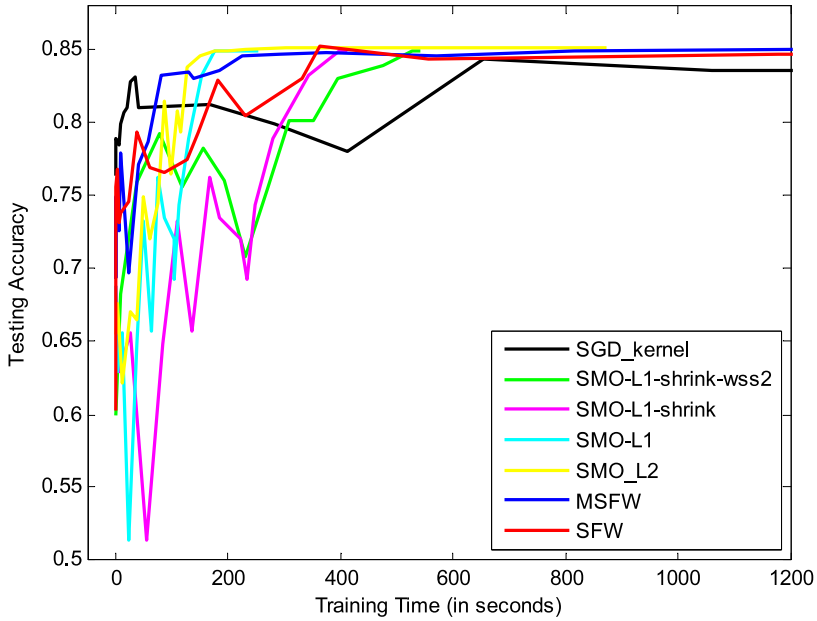

Figure 25: Dataset: a9a

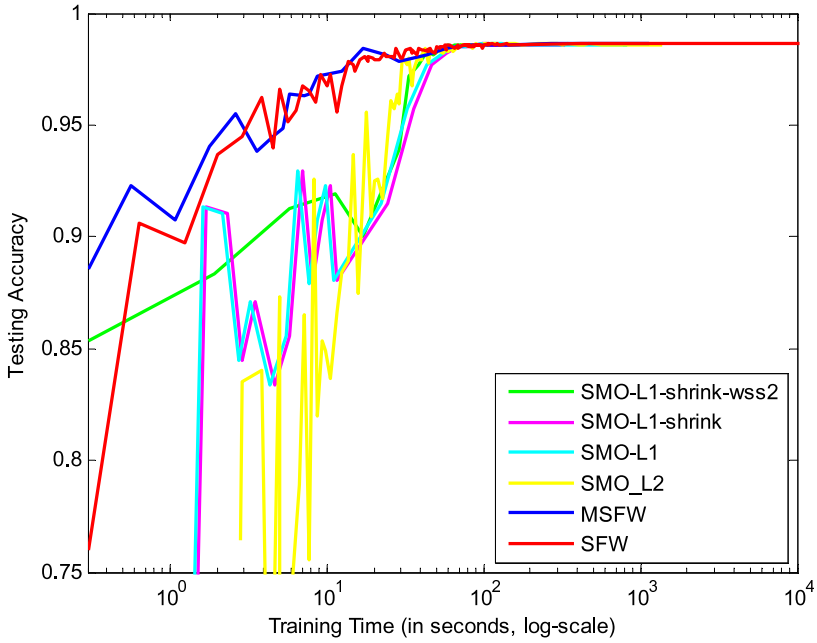

Figure 26: Dataset: ijcnn1

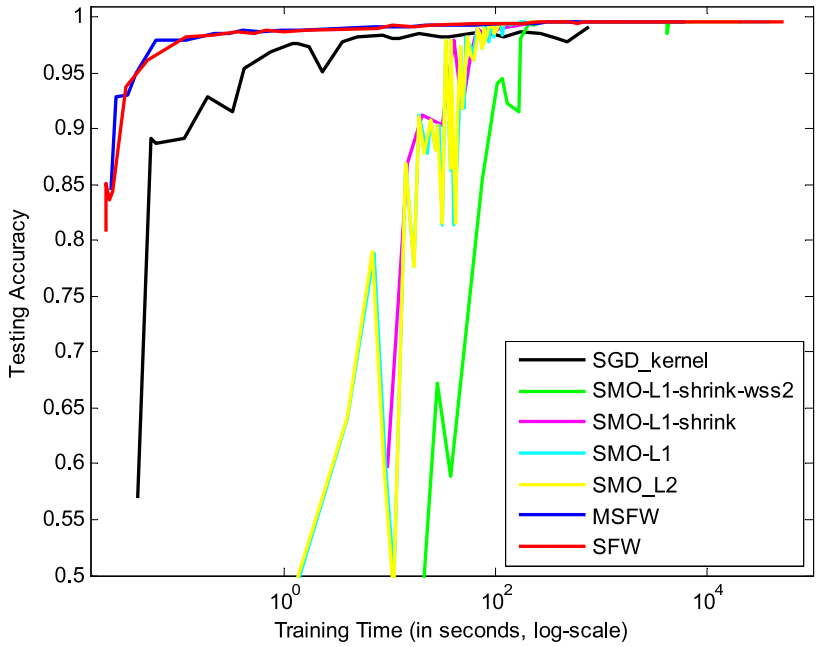

Figure 27: Dataset: usps01

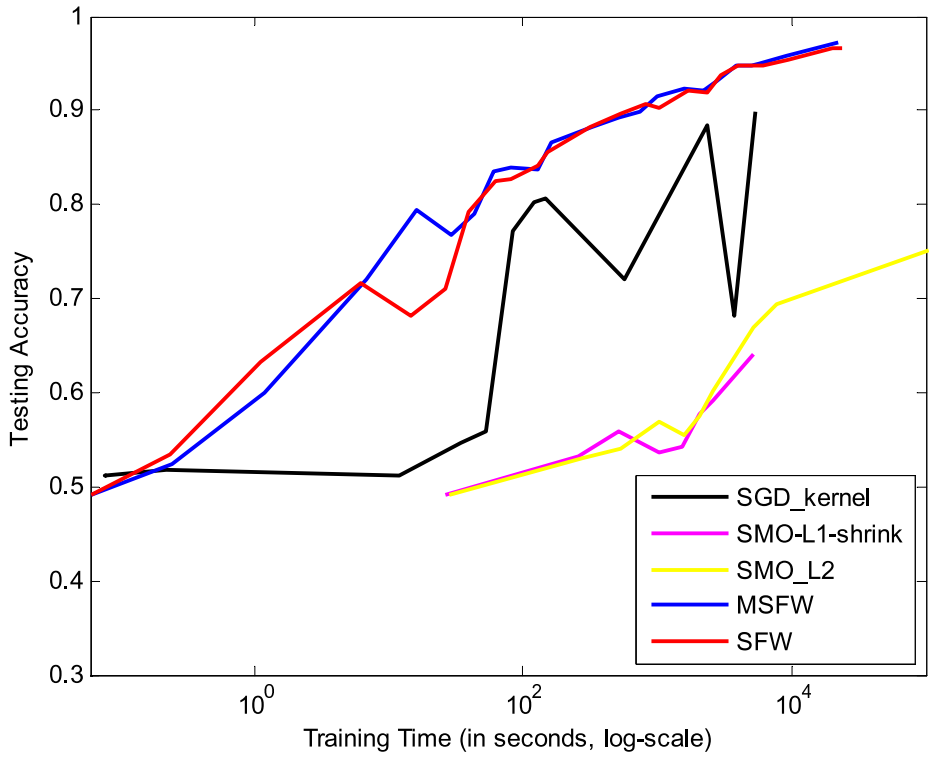

Figure 28: Dataset: covertype

Dataset	Best Testing Accuracy (%)								CPU Time for Training (in seconds)							
	SFW 1 epo.	SFW 2 epo.	MSFW 1 epo.	MSFW 2 epo.	SMO-L2	SMO-L1	SMO-shrink	SMO-wss2	SFW 1 epo.	SFW 2 epo.	MSFW 1 epo.	MSFW 2 epo.	SMO-L2	SMO-L1	SMO-shrink	SMO-wss2
svmguide1	96.85	96.98	**97.00**	97.00	97.00	97.00	97.00	97.00	**0.30**	0.86	0.30	0.89	2.12	2.16	1.28	1.53
w3a	98.13	98.29	98.06	**98.33**	98.32	98.30	98.30	98.30	2.94	9.74	**2.79**	10.19	32.61	72.47	31.93	34.38
a9a	**85.24**	84.69	84.87	85.13	85.21	84.92	84.92	84.92	362.35	1178.7	375.75	1257.0	872.45	**222.55**	404.96	533.72
ijcnn1	98.07	98.58	98.17	98.55	**98.61**	98.61	98.61	98.61	24.08	75.73	**23.67**	78.51	637.32	808.32	134.12	76.83
usps01	**99.54**	99.53	99.54	99.53	99.54	99.54	99.54	99.54	3262.8	7228.7	**2534.2**	6242.1	19208	23226	3470.7	4421.0
covertype	96.68	98.23	97.23	98.23	98.23	**98.24**	98.24	98.24	23549	1.33e5	**22167**	1.13e5	2.46e5	2.13e5	2.06e5	2.33e5

Table 5: Experimental results for nonlinear SVM solvers. Best results are bolded and underscored.

Dataset	Number of Epochs/Iterations								Number of Support Vectors							
	SFW 1 epo.	SFW 2 epo.	MSFW 1 epo.	MSFW 2 epo.	SMO-L2	SMO-L1	SMO-shrink	SMO-wss2	SFW 1 epo.	SFW 2 epo.	MSFW 1 epo.	MSFW 2 epo.	SMO-L2	SMO-L1	SMO-shrink	SMO-wss2
svmguide1	1	2	1	2	2929	3010	3058	1250	693	971	672	942	1487	515	518	510
w3a	1	2	1	2	4178	9285	9360	4034	570	797	529	779	1273	562	572	564
a9a	1	2	1	2	34408	8829	8801	8110	12804	16743	13131	17267	20436	11980	11984	11981
ijcnn1	1	2	1	2	56139	69725	71003	10880	2772	3779	2706	3724	5286	2897	2899	2897
usps01	1	2	1	2	19394	23398	23398	11692	2509	2736	1382	1692	1959	1893	1893	1885
covertype	1	2	1	2	1.00e6	9.97e5	9.25e5	8.64e5	38004	1.10e5	29060	1.10e5	1.10e5	1.07e5	1.07e5	1.07e5

Table 6: Experimental results for nonlinear SVM solvers, continued.

and squared hinge loss.

4.5.6 Comparisons of Linear and Nonlinear SVMs

In this section, we use Figure 29∼33 to demonstrate the performance of SFW algorithms against two linear SVM solvers, namely SGD-linear and Pegasos. The result is very interesting. In most datasets, the linear solvers have a much lower testing accuracy than nonlinear ones. While in some datasets, e.g. "w3a", linear SVM solvers can be much faster than nonlinear solvers, with almost comparable testing accuracy on convergence. The reason might be the sparseness of this webpage data. In such datasets, a linear hyperplane is already enough to separate the two classes.

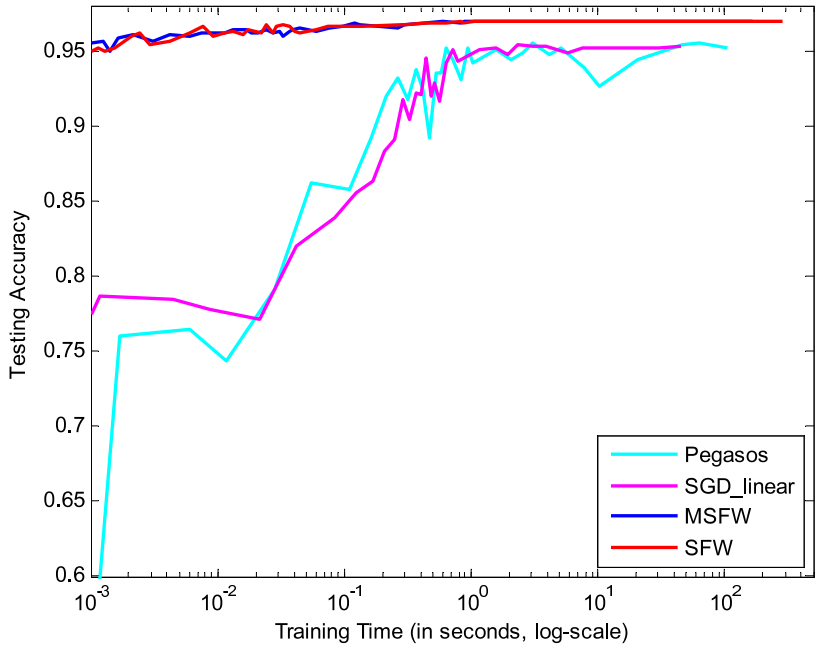

Figure 29: Dataset: svmguide1

4.6 Conclusions of this Chapter

We propose two stochastic Frank-Wolfe algorithms for nonlinear SVMs. These algorithms are very simple and efficient for both batch and online tasks. They achieve comparable or even better accuracies than state-of-the-art batch and online algorithms, and are significantly faster. SFW has a provable time complexity $O\left(d\frac{Q^2}{\epsilon^2}\right)$.

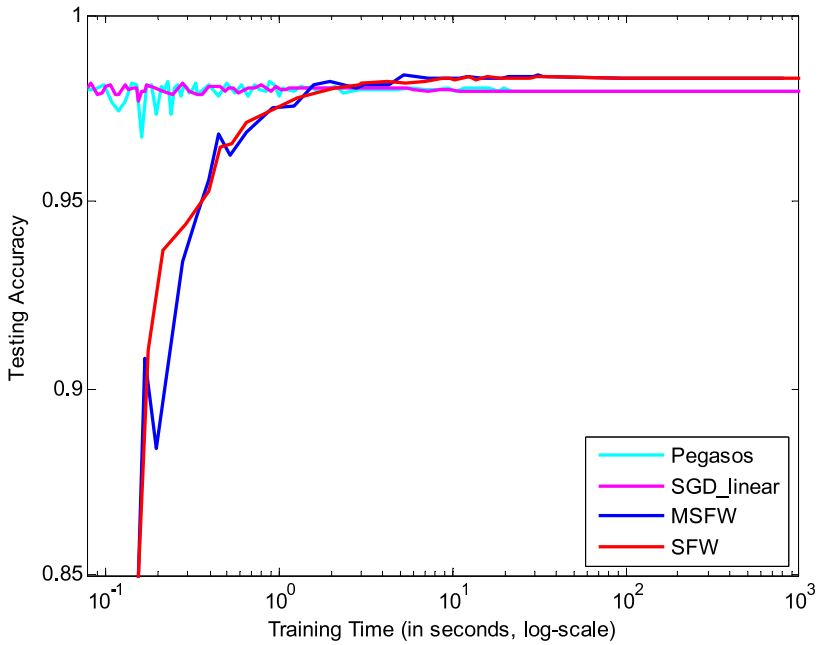

Figure 30: Dataset: w3a

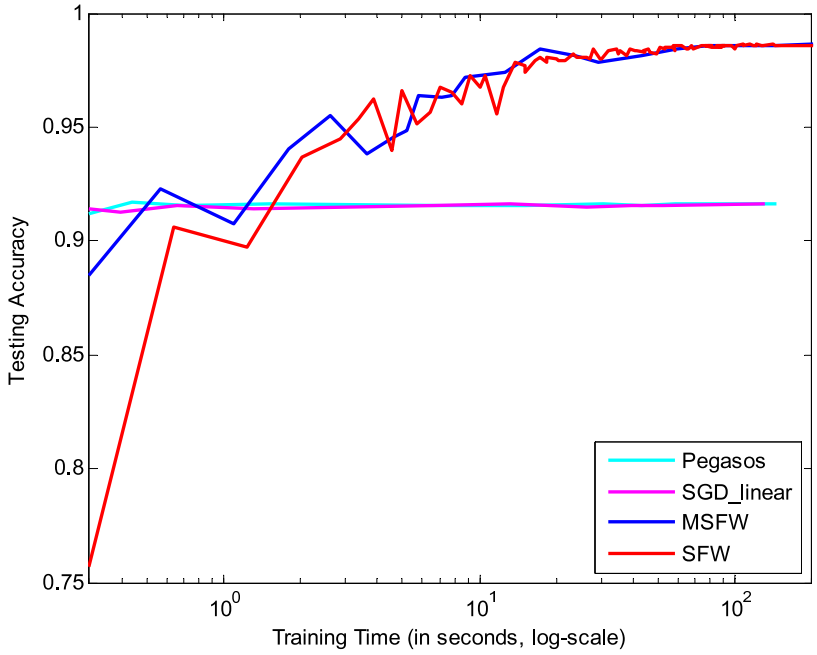

Figure 31: Dataset: ijcnn1

On-going work includes adopting shrinking and caching techniques. We will extend SFW to regression, ranking and semi-supervised learning problems. Sparse matrix storage

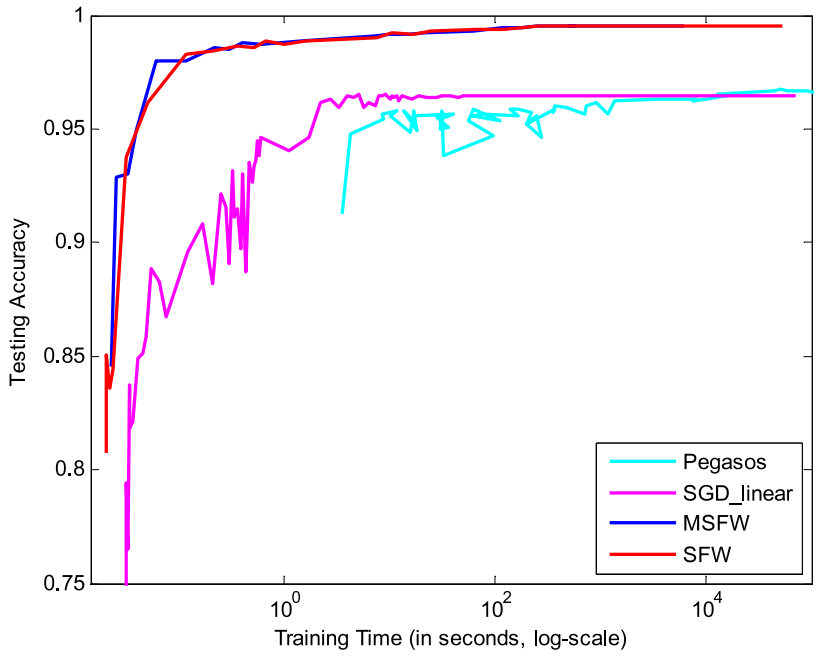

Figure 32: Dataset: usps01

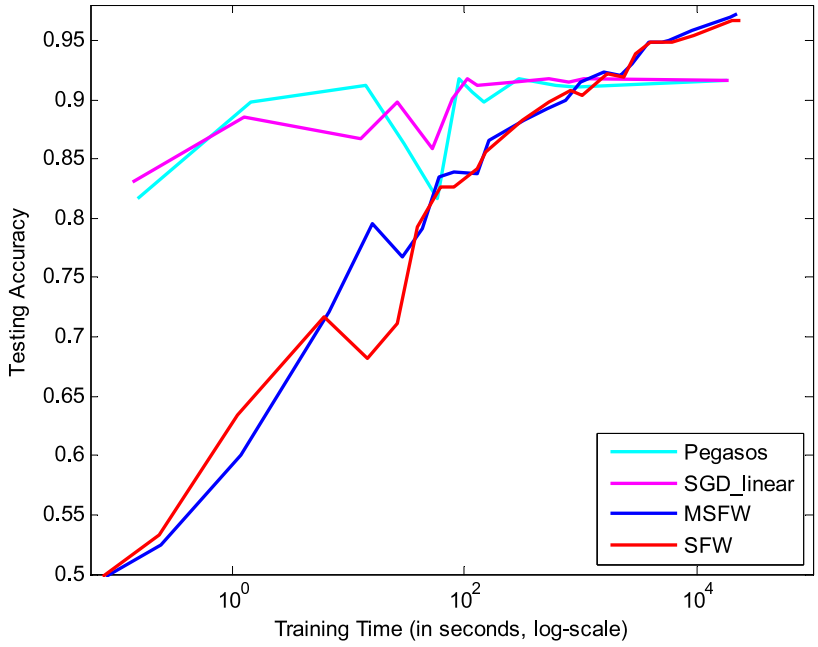

Figure 33: Dataset: covertype

and computation will be implemented for further speed up. The gap between the theoretical linear convergence of MSFW and its practical performance will also be investigated.

CHAPTER V

DISTRIBUTED LEARNING VIA CONSENSUS ADMM

5.1 Introduction

In this chapter we investigate the problem of data-distributed learning. This is a important problem that arises in many real-world machine learning applications. For example, in many large-scale machine learning systems, data samples are distributed over hundreds or thousands of general purpose servers. Locally accessing data is typically faster than the remote access due to the latency of network communication and limited bandwidth. The same problem can happen in wireless sensor networks where the data is collected locally by each sensor node and the resource constraints preclude any learning algorithm that demands high volumes of inter-sensor communications. In both these realistic scenarios, there is no pragmatic or desirable way to move data to a central node or move large amount of data between nodes. Despite long-standing efforts to federate data in various ways, in reality for large-scale problems, data will always be distributed for various reasons.

We formulate the distributed learning problem as a consensus constrained optimization problem and solve it using the general methodology of Alternating Direction Method of Multipliers (ADMM) (Glowinski and Marroco [1975], Gabay and Mercier [1976]). As surveyed in the monograph (Boyd et al. [2010]), ADMM is a flexible algorithmic framework for solving constrained problems. Its unique characteristic of "separability" can be utilized to explore various structures of the learning problems. For our distributed consensus learning problem, the main structure of concern is the underlying communication topology, which can be easily modeled as equality constraints in ADMM. Topology is one of the most critical issues in implementing consensus learning for two reasons: First, different topologies might lead to different iteration complexities for the algorithms. Second, the distribution and number of edges in the communication graph will result in different communication overloads. A practical system should always make a proper balance between these factors.

One of the central themes in distributed learning is the question "What is the *best* communication topology?" To reach a definitive answer to this question, one still needs to overcome major hurdles because the convergence behavior of ADMM in this context not only depends on the communication topology, but also on the penalty parameter β used in the augmented Lagrangian. The main focus of this chapter is to characterize the interplay between these factors, and to this end we present a new convergence analysis for ADMM with Lipschitz smooth and strongly convex functions (Section 5.4). Based on the derived convergence rates, we design an adaptive scheme to choose β (Section 5.5). In Section 5.6 we use several sets of numerical examples to show: a) to what extent does β affect the convergence rates; b) given the "optimal" β, which topology achieves faster convergence rates; c) the effectiveness of the proposed adaptive β strategy; and d) a practical selection for β for simple ADMM cases.

5.1.1 Related Work

There are generally two classes of methods for the distributed learning in the literature. The first class includes the gradient-based primal methods: e.g. the distributed subgradient descent methods (Nedic and Ozdaglar [2009], Dekel et al. [2011]) and the distributed dual averaging methods (Duchi et al. [2010], Agarwal and Duchi [2011], Duchi et al. [2012]). The second class are primal-dual methods based on the augmented Lagrangian method (Zhu et al. [2009]) or ADMM (Boyd et al. [2010], Mateos et al. [2010], Mota et al. [2012]). In gradient-based methods, the (sub)gradients are transmitted and aggregated in the hope that all workers will asymptotically obtain information from all data samples. While for the second class, the consensus requirements are *explicitly* encoded as constraints, and all data samples are kept local. The starting point for our work is the D-ADMM algorithm (Mota et al. [2012]) which belongs to the second class. However in this chapter we focus on the convergence behavior of the algorithm and we want to investigate how it will be affected by the various factors of our problem.

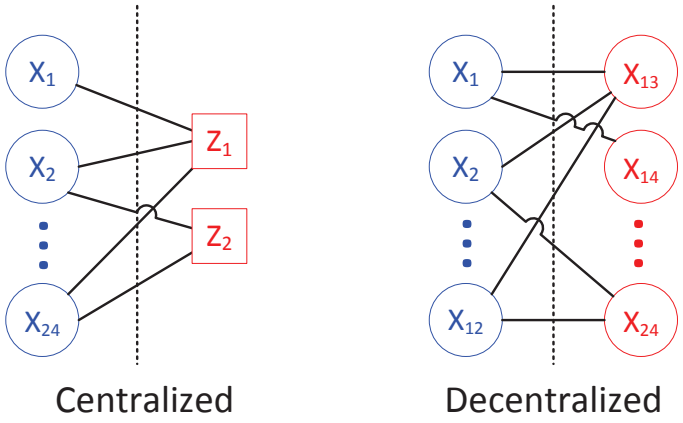

Figure 34: Two ways to formulate bipartite graphs. Left: centralized learning with two global (central) variables. Right: decentralized learning.

5.2 Problem Settings and Notations

We are interested in the following distributed consensus learning problem:

$$\begin{aligned} &\min f(\mathbf{x}) \equiv \sum_{i=1}^{N} f_i(\mathbf{x}_i), \\ &\text{s.t. } \mathbf{x}_1 = \mathbf{x}_2 = \ldots = \mathbf{x}_N, \end{aligned} \tag{131}$$

where $\mathbf{x}_i \in \mathbf{R}^D$ and each worker i is associated with an individual function $f_i(\mathbf{x}_i)$ and its corresponding subset of data. The N distributed workers are connected via a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N indexed vertices and $\mathcal{E}$ is the set of edges of the network. Each vertex v_i is associated with a local variable $\mathbf{x}_i$. Information can be transferred between v_i and v_j in either direction as long as they are connected by edge e_{ij}. Note that despite the connectivity via e_{ij}, v_i and v_j have the freedom to choose whether they want to exchange information or not. In other words, $\mathcal{G}$ only reflects the connectivity, but not communications.

We propose to solve problem (131) by ADMM in parallel. To take advantage of ADMM's capacities in dealing with separable functions, we have at least the following two structural options, as illustrated in Fig.34, where we use a case with 24 workers as an example.

1. **Centralized Learning.** We use axillary global (central) variables $\mathbf{z} \equiv \{\mathbf{z}_j\}$ such that every $\mathbf{x}_i$ are connected to some $\mathbf{z}_j$. In this way we can reproduce equivalent

connectivities represented by the original graph $\mathcal{G}$. When $|\mathbf{z}| = 1$, this is called master-slave consensus optimization, where the global variable $\mathbf{z}$ is hosted by the master node, and all $\mathbf{x}_i$ variables are updated at slaves nodes. When $|\mathbf{z}| > 1$, the paradigm is called general form consensus optimization (Boyd et al. [2010]).

2. **Decentralized Learning.** Global variables are not necessary in this paradigm, hence there is no master node. The N local functions f_i are simply divided into groups, where communication only happens between different groups, but not within each group. For simplicity, we divide them into 2 groups. Following the work of (Mota et al. [2012]) we design a *bipartite graph* for communication.

In this chapter we focus on the second paradigm since the centralized learning can be regarded as a special case of the decentralized learning where the master nodes do not have their own data samples.

Both the above two distributed learning paradigms can be conveniently formulated as the following problem that can be solved by ADMM:

$$\begin{aligned} &\min_{\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y}} \theta_1(\mathbf{x}) + \theta_2(\mathbf{y}), \\ &\text{s.t. } A\mathbf{x} + B\mathbf{y} = \mathbf{b}, \end{aligned} \tag{132}$$

where θ_1 and θ_2 are convex functions, $\mathcal{X}$ and $\mathcal{Y}$ are closed convex sets. In this chapter, instead of using the classic ADMM (Boyd et al. [2010]), we follow the scheme of generalized ADMM (Alg.8) as discussed in (He and Yuan [2012b]). The only difference is the additional term for the proximity function $\frac{1}{2}\|\mathbf{x} - \mathbf{x}^k\|_G^2$, where the $G-$norm is defined as $\|\mathbf{x}\|_G = \sqrt{\mathbf{x}^T G\mathbf{x}}$. Variations of ADMM can be derived from different G, e.g. the linearized ADMM (Goldfarb et al. [2010], Zhang et al. [2011]). We use $\|\cdot\|$ to denote the l_2 norm. The *augmented Lagrangian* in Alg.8 is defined as:

$$\mathcal{L}_\beta(\mathbf{x}, \mathbf{y}, \lambda) \equiv \theta_1(\mathbf{x}) + \theta_2(\mathbf{y}) - \langle \boldsymbol{\lambda}, A\mathbf{x} + B\mathbf{y} - \mathbf{b}\rangle + \frac{\beta}{2}\|A\mathbf{x} + B\mathbf{y} - \mathbf{b}\|^2, \tag{133}$$

where β is a pre-defined penalty parameter that is crucial in achieving faster rates of convergence. We make the following assumptions for the rest of this chapter.

Algorithm 8 Generalized ADMM ($G \succeq 0$)

[0.] Initialize $\mathbf{y}^0$ and $\boldsymbol{\lambda}^0$.
for $k = 0, 1, 2, \ldots$ **do**
  [1.] $\mathbf{x}^{k+1} \leftarrow \arg\min_{\mathbf{x}\in\mathcal{X}} \mathcal{L}_\beta(\mathbf{x}, \mathbf{y}^k, \boldsymbol{\lambda}^k) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}^k\|_G^2$.
  [2.] $\mathbf{y}^{k+1} \leftarrow \arg\min_{\mathbf{y}\in\mathcal{Y}} \mathcal{L}_\beta(\mathbf{x}^{k+1}, \mathbf{y}, \boldsymbol{\lambda}^k)$.
  [3.] $\boldsymbol{\lambda}^{k+1} \leftarrow \boldsymbol{\lambda}^k - \beta\left(A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}\right)$.
end for

Assumption 5. *Functions θ_1 and θ_2 are L_1 and L_2 Lipschitz smooth, and are μ_1 and μ_2 strongly convex.*

5.3 Distributed Consensus Learning

As discussed in Section 5.2, we are interested in the decentralized learning paradigm where the N workers constitute a bipartite graph $\mathcal{B} \equiv \{\mathcal{V}_L, \mathcal{V}_R, \mathcal{C}\}$ with left part $\mathcal{V}_L$ and right part $\mathcal{V}_R$. The communication edge set $\mathcal{C} \subseteq \mathcal{E}$ represents the communication scheme: if there is an edge c_{pn}, then worker v_p and v_n will exchange information in each iteration of ADMM. Note that even if v_p and v_n is connected by the network edge $e_{pn} \in \mathcal{E}$, no communication will be carried out if they are not connected by c_{pn}.

The distributed consensus learning can thus be formulated as an optimization problem with $|\mathcal{C}|$ equality constraints $\{\mathbf{x}_p = \mathbf{y}_n : \ \forall c_{np} \in \mathcal{C}\}$. Writing these constraints in ADMM's matrix form $A\mathbf{x} + B\mathbf{y} = \mathbf{0}$, we can see that $A \in \mathbb{R}^{D|\mathcal{C}|\times D|\mathcal{V}_L|}$ is a matrix of $|\mathcal{C}|$ block-rows, with each block row containing only one identity matrix I and 0 for others. Matrix B is defined similarly, with each block-row containing only one $-I$. The positions of I and $-I$ in each block-row of A and B indicates the consensus between two specific workers. An example is illustrated in Fig.35. Since there are $|\mathcal{C}|$ consensus constraints, we introduce $|\mathcal{C}|$ Lagrangian multipliers $\boldsymbol{\lambda}_{pn}$ for each edge c_{pn}. The ADMM based distributed consensus learning is given in Alg.9, where the augmented Lagrangians are

$$\begin{aligned}
\mathcal{L}_\beta(\mathbf{x}_i, \mathbf{y}^k, \boldsymbol{\lambda}^k) &= f_i(\mathbf{x}_i) - \sum_{n=1}^{\mathcal{N}_i} \langle \boldsymbol{\lambda}_{in}^k, \mathbf{x}_i \rangle + \frac{\beta}{2} \sum_{n=1}^{\mathcal{N}_i} \|\mathbf{x}_i - \mathbf{y}_n^k\|^2 \\
\mathcal{L}_\beta(\mathbf{x}^{k+1}, \mathbf{y}_i, \boldsymbol{\lambda}^k) &= f_i(\mathbf{y}_i) + \sum_{p=1}^{\mathcal{P}_i} \langle \boldsymbol{\lambda}_{pi}^k, \mathbf{y}_i \rangle + \frac{\beta}{2} \sum_{p=1}^{\mathcal{P}_i} \|\mathbf{x}_p^{k+1} - \mathbf{y}_i\|^2.
\end{aligned} \tag{134}$$

Here $\mathcal{N}_i$ represents the number of right workers (in $\mathcal{V}_R$) connected to the left worker i, and

$\mathcal{P}_i$ represents the number of left workers (in $\mathcal{V}_L$) connected to the right worker i.

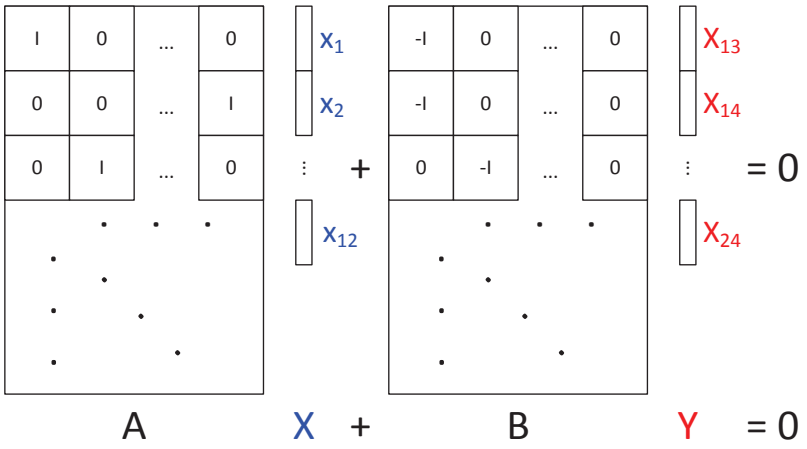

Figure 35: Consensus constraints expressed in matrix form.

In Alg.9, all $\mathbf{x}_i$ are updated in parallel by the left workers, followed by the parallel updates of $\mathbf{y}_i$ by the right workers. In practice, all the updates of $\boldsymbol{\lambda}$ are computed in parallel by the right workers, since they have access to the latest copies of $\mathbf{y}^{k+1}$ and $\mathbf{x}^{k+1}$ in each iteration k, while the left workers only have $\mathbf{x}^{k+1}$ and the old copy of $\mathbf{y}^k$.

Algorithm 9 Distributed Consensus Learning

[0.] Initialize $\mathbf{y}^0$ and $\boldsymbol{\lambda}^0$.
for $k = 0, 1, 2, \ldots$ **do**
  [1.] $\forall i$ (parallel) $\mathbf{x}_i^{k+1} \leftarrow \arg\min_{\mathbf{x}_i} \mathcal{L}_\beta(\mathbf{x}_i, \mathbf{y}^k, \boldsymbol{\lambda}^k)$.
  [2.] $\forall i$ (parallel) $\mathbf{y}_i^{k+1} \leftarrow \arg\min_{\mathbf{y}_i} \mathcal{L}_\beta(\mathbf{x}^{k+1}, \mathbf{y}_i, \boldsymbol{\lambda}^k)$.
  [3.] $\forall p, n$ (parallel) $\boldsymbol{\lambda}_{pn}^{k+1} \leftarrow \boldsymbol{\lambda}_{pn}^k - \beta\left(\mathbf{x}_p^{k+1} - \mathbf{y}_n^{k+1}\right)$.
end for

5.3.1 Three Dimensions of the Problem Space

Taking a closer look at Alg.9 we can find that there are actually three factors for the implementation of this algorithm. Firstly, we can choose any communication topology that is encoded in matrices A and B. Secondly, the penalty parameter β can be any positive number. Thirdly, it is free to change the updating order for $\mathbf{x}$ and $\mathbf{y}$ (the update of $\boldsymbol{\lambda}$ should also be modified accordingly). In order to investigate the interactions among these factors, we use both theoretical analysis (Section 5.4, 5.5) and numerical examples (Section 5.6) to

study the convergence of Alg.9.

5.4 Iteration Complexities of ADMM

The global convergence of ADMM was established in the literature (Gabay [1983], Glowinski and Tallec [1989], Eckstein and Bertsekas [1992]). The $O(1/k)$ convergence rate was established by (He and Yuan [2012a,b]) where the authors only assume that θ_1 and θ_2 are convex. When these functions are both Lipschitz smooth and strongly convex, linear convergence rates are reported very recently. In (Hong and Luo [2012]), the authors derived R-linear rates for the sum of primal and dual gaps for a setting that is more general than (132). However, the constants in the bound is not directly applicable to our setting. In (Deng and Yin [2012]), the authors present linear rates only for the case when $G = 0$, and as a consequence no rate is given for $\mathbf{x}$. In the following we present explicit formulas of linear rates for all the primal variables $\mathbf{x}$, $\mathbf{y}$ and dual variable $\boldsymbol{\lambda}$.

Lemma 13. *Let $l(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ be a convex differentiable function with gradient $\mathbf{g}$. Let scalar $s \geq 0$. For any vector $\mathbf{u}$ and $\mathbf{v}$, denote their Bregman divergence as $D(\mathbf{u}, \mathbf{v})$. If $\forall \mathbf{u} \in \mathcal{X}$, $\mathbf{x}^* \equiv \arg\min_{\mathbf{x}\in\mathcal{X}} l(\mathbf{x}) + sD(\mathbf{x}, \mathbf{u})$, then with $\Theta \equiv \langle \mathbf{g}(\mathbf{x}^*), \mathbf{x}^* - \mathbf{x} \rangle$, we have*

$$\Theta \leq s \left[D(\mathbf{x}, \mathbf{u}) - D(\mathbf{x}^*, \mathbf{u}) - D(\mathbf{x}, \mathbf{x}^*) \right]. \tag{135}$$

Proof. Invoking the optimality condition we have

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

which is equivalent to

$$\begin{aligned} \langle \mathbf{g}(\mathbf{x}^*), \mathbf{x}^* - \mathbf{x} \rangle &\leq s \langle \nabla D(\mathbf{x}^*, \mathbf{u}), \mathbf{x} - \mathbf{x}^* \rangle \\ &= s \langle \nabla \omega(\mathbf{x}^*) - \nabla \omega(\mathbf{u}), \mathbf{x} - \mathbf{x}^* \rangle \\ &= s \left[D(\mathbf{x}, \mathbf{u}) - D(\mathbf{x}, \mathbf{x}^*) - D(\mathbf{x}^*, \mathbf{u}) \right]. \end{aligned}$$

□

The following key lemma states that $\|\mathbf{w}^k - \mathbf{w}^*\|_M$ is monotonically non-increasing, and the reduction of $\mathbf{w}^k - \mathbf{w}^*$ is faster than $\mathbf{w}^k - \mathbf{w}^{k+1}$. Variations of this lemma have been

presented several times in the literature under different settings and assumptions (He et al. [2000], Boyd et al. [2010], He and Yuan [2012b], Deng and Yin [2012]). Our result is more general in the sense that this lemma is applicable to convex feasible sets $\mathcal{X}$ and $\mathcal{Y}$, not just $\mathbb{R}^x$ and $\mathbb{R}^y$. The proof is pretty simple and only relies on the optimality conditions.

Lemma 14. *Under Assumption 5 we have*

$$\begin{aligned}&\|\mathbf{w}^k - \mathbf{w}^*\|_M^2 - \|\mathbf{w}^{k+1} - \mathbf{w}^*\|_M^2 \geq \|\mathbf{w}^k - \mathbf{w}^{k+1}\|_M^2 \\ &+ 2\mu_1\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + 2\mu_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2,\end{aligned} \tag{136}$$

where $\mathbf{w}^k \equiv (\mathbf{x}^k, \mathbf{y}^k, \boldsymbol{\lambda}^k)^T$, $\mathbf{w}^* \equiv (\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^*)^T$ *is the optimal solution of (132), and*

$$M \equiv Diag\left(G,\ \beta B^T B,\ \frac{I}{\beta}\right). \tag{137}$$

Proof. By the strong convexity of θ_1 and θ_2 we have $\forall \mathbf{x} \in \mathcal{X}$ and $\forall \mathbf{y} \in \mathcal{Y}$:

$$\theta_1(\mathbf{x}^{k+1}) - \theta_1(\mathbf{x}) \leq \left\langle \theta_1'(\mathbf{x}^{k+1}), \mathbf{x}^{k+1} - \mathbf{x} \right\rangle - \frac{\mu_1}{2}\|\mathbf{x}^{k+1} - \mathbf{x}\|^2. \tag{138}$$

$$\theta_2(\mathbf{y}^{k+1}) - \theta_2(\mathbf{y}) \leq \left\langle \theta_2'(\mathbf{y}^{k+1}), \mathbf{y}^{k+1} - \mathbf{y} \right\rangle - \frac{\mu_2}{2}\|\mathbf{y}^{k+1} - \mathbf{y}\|^2. \tag{139}$$

Invoking the optimality condition of Line 2 of Alg. 8 we have $\forall \mathbf{y} \in \mathcal{Y}$:

$$\langle \theta_2'(\mathbf{y}^{k+1}) + B^T\left[\beta(A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}) - \boldsymbol{\lambda}^k\right], \mathbf{y}^{k+1} - \mathbf{y}\rangle \leq 0. \tag{140}$$

Using Lemma 13 by taking the Bregman divergence $D(\cdot,\cdot)$ as $\|\|_G^2$ ($G \succeq 0$) we have $\forall \mathbf{x} \in \mathcal{X}$:

$$\begin{aligned}&\theta_1(\mathbf{x}^{k+1}) - \theta_1(\mathbf{x}) + \left\langle \mathbf{x}^{k+1} - \mathbf{x}, -A^T\boldsymbol{\lambda}^{k+1} \right\rangle \\ &\overset{(138)}{\leq} \left\langle \theta_1'(\mathbf{x}^{k+1}) - A^T\boldsymbol{\lambda}^{k+1}, \mathbf{x}^{k+1} - \mathbf{x} \right\rangle - \frac{\mu_1}{2}\|\mathbf{x}^{k+1} - \mathbf{x}\|^2 \\ &= \left\langle \theta_1'(\mathbf{x}^{k+1}) + A^T\left[\beta(A\mathbf{x}^{k+1} + B\mathbf{y}^k - \mathbf{b}) - \boldsymbol{\lambda}^k\right], \mathbf{x}^{k+1} - \mathbf{x} \right\rangle \\ &- \frac{\mu_1}{2}\|\mathbf{x}^{k+1} - \mathbf{x}\|^2 + \left\langle \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k), \mathbf{x}^{k+1} - \mathbf{x} \right\rangle \\ &\overset{(135)}{\leq} \frac{1}{2}\left(\|\mathbf{x} - \mathbf{x}^k\|_G^2 - \|\mathbf{x} - \mathbf{x}^{k+1}\|_G^2\right) - \frac{1}{2}\|\mathbf{x}^k - \mathbf{x}^{k+1}\|_G^2 \\ &- \frac{\mu_1}{2}\|\mathbf{x}^{k+1} - \mathbf{x}\|^2 + \left\langle \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k), \mathbf{x}^{k+1} - \mathbf{x} \right\rangle\end{aligned} \tag{141}$$

The last term can be further bounded as

$$
\begin{aligned}
&\left\langle \beta A^T B(\mathbf{y}^{k+1}-\mathbf{y}^k), \mathbf{x}^{k+1}-\mathbf{x}\right\rangle \\
&= \frac{\beta}{2}\left(\|A\mathbf{x}+B\mathbf{y}^k-\mathbf{b}\|^2 - \|A\mathbf{x}+B\mathbf{y}^{k+1}-\mathbf{b}\|^2\right) \\
&+ \frac{\beta}{2}\left(\|A\mathbf{x}^{k+1}+B\mathbf{y}^{k+1}-\mathbf{b}\|^2 - \|A\mathbf{x}^{k+1}+B\mathbf{y}^{k}-\mathbf{b}\|^2\right) \\
&= \frac{\beta}{2}\left(\|A\mathbf{x}+B\mathbf{y}^k-\mathbf{b}\|^2 - \|A\mathbf{x}+B\mathbf{y}^{k+1}-\mathbf{b}\|^2\right) \\
&- \frac{\beta}{2}\|\mathbf{y}^k-\mathbf{y}^{k+1}\|^2_{B^TB} - (\mathbf{y}^k-\mathbf{y}^{k+1})^T B^T(\boldsymbol{\lambda}^k-\boldsymbol{\lambda}^{k+1}) \\
&\le \frac{\beta}{2}\left(\|A\mathbf{x}+B\mathbf{y}^k-\mathbf{b}\|^2 - \|A\mathbf{x}+B\mathbf{y}^{k+1}-\mathbf{b}\|^2\right) \\
&- \frac{\beta}{2}\|\mathbf{y}^k-\mathbf{y}^{k+1}\|^2_{B^TB},
\end{aligned}
\tag{142}
$$

where in the last step we used Lemma 3.1 of (He and Yuan [2012b]). Combining (139) and (140) we have

$$
\theta_2(\mathbf{y}^{k+1}) - \theta_2(\mathbf{y}) + \left\langle \mathbf{y}^{k+1}-\mathbf{y}, -B^T\boldsymbol{\lambda}^{k+1}\right\rangle \le -\frac{\mu_2}{2}\|\mathbf{y}^{k+1}-\mathbf{y}\|^2. \tag{143}
$$

We also have the following equality from the updating rule of $\boldsymbol{\lambda}$ in Line 3:

$$
\begin{aligned}
&\left\langle \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}, A\mathbf{x}^{k+1}+B\mathbf{y}^{k+1}-\mathbf{b}\right\rangle \\
&= \frac{1}{\beta}\left\langle \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}, \boldsymbol{\lambda}^k-\boldsymbol{\lambda}^{k+1}\right\rangle \\
&= \frac{1}{2\beta}\left(\|\boldsymbol{\lambda}-\boldsymbol{\lambda}^k\|^2 - \|\boldsymbol{\lambda}-\boldsymbol{\lambda}^{k+1}\|^2\right) - \frac{1}{2\beta}\|\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^k\|^2.
\end{aligned}
\tag{144}
$$

Summing (141), (142), (143) and (144), taking $\mathbf{x}=\mathbf{x}^*, \mathbf{y}=\mathbf{y}^*, \boldsymbol{\lambda}=\boldsymbol{\lambda}^*$ and using the fact

that $A\mathbf{x}^* + B\mathbf{y}^* - \mathbf{b} = 0$ we get

$$\begin{aligned}
&\frac{1}{2}\left(\|\mathbf{x}^k - \mathbf{x}^*\|_G^2 - \|\mathbf{x}^{k+1} - \mathbf{x}^*\|_G^2\right) \\
&+ \frac{\beta}{2}\left(\|\mathbf{y}^k - \mathbf{y}^*\|_{B^TB}^2 - \|\mathbf{y}^{k+1} - \mathbf{y}^*\|_{B^TB}^2\right) \\
&+ \frac{1}{2\beta}\left(\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*\|^2 - \|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2\right) \\
&\geq \frac{\mu_1}{2}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + \frac{\mu_2}{2}\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 \\
&+ \frac{1}{2}\|\mathbf{x}^k - \mathbf{x}^{k+1}\|_G^2 + \frac{\beta}{2}\|\mathbf{y}^k - \mathbf{y}^{k+1}\|_{B^TB}^2 + \frac{1}{2\beta}\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\|^2 \\
&+ \left[\theta_1(\mathbf{x}^{k+1}) - \theta_1(\mathbf{x}^*) + \theta_2(\mathbf{y}^{k+1}) - \theta_2(\mathbf{y}^*)\right] \\
&+ (\mathbf{x}^{k+1} - \mathbf{x}^*)^T(-A^T\boldsymbol{\lambda}^{k+1}) + (\mathbf{y}^{k+1} - \mathbf{y}^*)^T(-B^T\boldsymbol{\lambda}^{k+1}) \\
&+ (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*)^T(A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}) \\
&\geq \mu_1\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + \mu_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 + \frac{1}{2}\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_M^2,
\end{aligned} \tag{145}$$

where the last inequality is due to the strong convexity of θ_1 and θ_2. □

Remark 2. *For the general convex cases, i.e. $\mu_1 = \mu_2 = 0$, the $O(1/k)$ convergence rate of ADMM can be easily derived from Lemma 14 (He and Yuan [2012b]).*

5.4.1 Linear Convergence Rates

For strongly convex ($\mu_1, \mu_2 > 0$) and Lipschitz smooth functions, linear convergence rates can also be obtained from Lemma 14. Note that all the results in this section rely on the assumption that $\mathcal{X} = \mathbb{R}^x$ and $\mathcal{Y} = \mathbb{R}^y$. In the following results we use $\Lambda_{\max}(M)$ and $\Lambda_{\min}(M)$ to denote the maximum and minimum eigenvalues of a matrix M.

We are interested in the following two cases that will be presented separately: $G = 0$ for the classic ADMM and $G \succ 0$ for the generalized ADMM.

Theorem 13. *When $G = 0$, $\mathcal{X} = \mathbb{R}^x$ and $\mathcal{Y} = \mathbb{R}^y$, $\mathbf{x}$, $\mathbf{y}$ and $\boldsymbol{\lambda}$ converge linearly:*

$$\|\mathbf{w}^{k+1} - \widetilde{\mathbf{w}}^*\|_P^2 \leq \left(\frac{1}{1+\tau}\right)^k \|\mathbf{w}^0 - \mathbf{w}^*\|_M^2, \tag{146}$$

where

$$\begin{aligned}
&P \equiv \left(2\mu_1 + \frac{\mu_1^2}{2\beta\Lambda_{\max}(AA^T)}, 2\mu_2 + \beta\Lambda_{\min}(B^TB), \frac{1}{\beta}\right) I, \\
&\widetilde{\mathbf{w}}^* \equiv (\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^k)^T, \text{and}
\end{aligned}$$

$$\tau \equiv \frac{2\mu_2}{\frac{L_2^2}{\beta c} + \beta \Lambda_{\max}(B^T B)}. \tag{147}$$

Here $c > 0$ is the largest positive constant that satisfies

$$\|B^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*)\|^2 \geq c\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2 \quad \forall k. \tag{148}$$

Theorem 14. *When $G \succ 0$, $\mathcal{X} = \mathbb{R}^x$ and $\mathcal{Y} = \mathbb{R}^y$, $\mathbf{x}$, $\mathbf{y}$ and $\boldsymbol{\lambda}$ converge linearly:*

$$\|\mathbf{w}^{k+1} - \mathbf{w}^*\|_M^2 \leq \left(\frac{1}{1+\tau_G}\right)^k \|\mathbf{w}^0 - \mathbf{w}^*\|_M^2, \tag{149}$$

where

$$\tau_G \equiv \min\left\{\frac{2\mu_1}{\Lambda_{\max}(G)}, \tau\right\},$$

M is defined in (137) and τ is defined in (147).

5.5 Strategy for Choosing β Adaptively

Despite of many efforts towards finding a good penalty parameter β (He et al. [2000], Wang and Liao [2001], Cands et al. [2011]), it still remains a serious issue in implementing any instance of ADMM. This parameter controls the balance between the reductions of the dual residual $\mathbf{s}^{k+1} \equiv \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k)$ and the primal residual $\mathbf{r}^{k+1} \equiv A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}$ as defined in (Boyd et al. [2010]). A large β enforces more the primal feasibility $A\mathbf{x}_k - B\mathbf{y}_k = \mathbf{b}$, but results in a larger violation in the dual feasibility. A small β tends to reduce the difference between $\mathbf{y}^{k+1}$ and $\mathbf{y}^k$, leading to a faster satisfaction of the dual feasibility, at the expense of a larger violation of the primal feasibility.

Moreover, a bad choice of β might lead to very slow convergence rates for *both* the primal and dual feasibilities. A numerical example for consensus least squares is shown in Fig.36, where the bipartite graph consists of only two workers and the consensus constraint is simply $\mathbf{x} = \mathbf{y}$. Increasing β from the optimal value 0.47 to 3 not only results in a significantly higher dual residual than the primal residual, but also slows down both residuals from 10^{-6} to 10^{-3} (primal) and 10^{-2} (dual), all measured at iteration 20. Decreasing β to 0.1 makes the primal residual higher than the dual residual, but both are around 10^{-4} at iteration 20, which are still much higher than those using the optimal β.

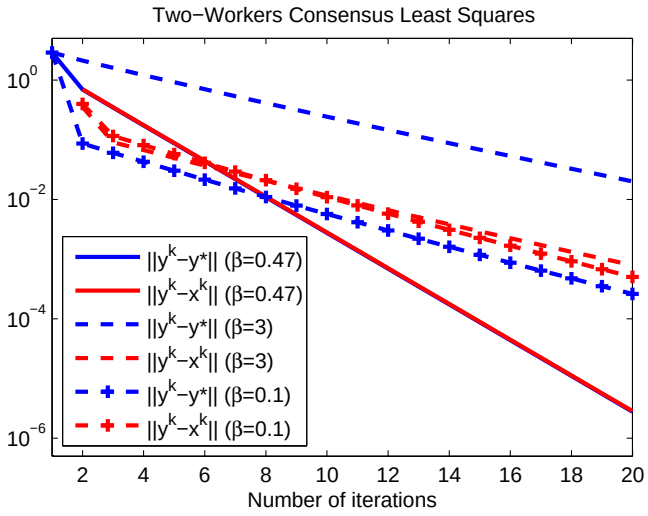

Figure 36: Values of β significantly affect convergence rates for both primal and dual residuals.

Since the optimal parameter β is essentially data-dependent, a natural idea is to search it adaptively during the iterations of ADMM. However we still need to answer two questions: 1. What is a good initial value β^0 that we shall start with? 2. What updating rule shall we adopt?

Towards the first question, we can use our convergence results that are presented in Theorem 13 and 14. For simplicity, we assume that in Theorem 14 ($G \succ 0$), we always choose a G such that $\frac{2\mu_1}{\Lambda_{\max}(G)} > \tau$. Then in both cases the linear convergence rate is upper bounded by $\left(\frac{1}{1+\tau}\right)^k$ where $\tau \equiv \frac{2\mu_2}{\frac{L_2^2}{\beta c}+\beta\Lambda_{\max}(B^TB)}$. Here $c > 0$ is the largest positive constant that satisfies $\|B^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*)\|^2 \geq c\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2$. Since a large τ results in a faster rate, we can let $\frac{L_2^2}{\beta c} = \beta\Lambda_{\max}(B^TB)$ and take the "optimal" $\beta^* = \frac{L_2}{c\Lambda_{\max}(B^TB)}$. Although BB^T is positive semidefinite, yet B is not always of full row-rank. Hence in the worst case BB^T could be singular and $c = \Lambda_{\min}(BB^T)$ can as small as 0, resulting in a $\beta^* = \infty$. However, in practice a very large β is rarely a good choice, implying that $c = \Lambda_{\min}(BB^T)$ might be too pessimistic. It is very hard to estimate c, since we do not know $\boldsymbol{\lambda}^*$, nor the relation between B and $\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*$. Our proposed strategy is to find an underestimated β by taking the most optimistic $\hat{c} = \Lambda_{\max}(BB^T) > c$ and the initial guess

$$\beta^0 = L_2/(\Lambda_{\max}(B^TB) * \Lambda_{\max}(BB^T)). \tag{150}$$

We can see that this underestimated β^0 is always smaller than β^*.

Towards the updating rule, we proposed a multiplicative method (Alg.10) that is inspired by (He et al. [2000], Wang and Liao [2001]). In these two papers, the authors proposed to choose β adaptively by $\beta^{k+1} \leftarrow \beta^k * m$ if $q^k = \frac{\|A\mathbf{x}^{k+1}+B\mathbf{y}^{k+1}-\mathbf{b}\|}{\|A^TB(\mathbf{y}^{k+1}-\mathbf{y}^k)\|}$ is larger than some threshold q^{th}, where $m > 1$ is a fixed and predefined constant. Typical choices might be $q^{\text{th}} = 10$ and $m = 2$ (Boyd et al. [2010]). In comparison, we propose to update β^k by multiplying an adaptive number $\sqrt{q^k} \equiv \sqrt{\frac{\|A\mathbf{x}^{k+1}+B\mathbf{y}^{k+1}-\mathbf{b}\|}{\|B(\mathbf{y}^{k+1}-\mathbf{y}^k)\|}}$. This simple method is motivated by the idea of balancing the convergence rates of the primal residual $\mathbf{r}^{k+1} \equiv A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}$ and the dual residual $\mathbf{s}^{k+1} \equiv \beta A^TB(\mathbf{y}^{k+1} - \mathbf{y}^k)$. Intuitively, the more q^k is deviated from 1, the further β^k is from β^*, hence deserving a larger scaling. Concrete examples that support this intuition are given in Sec. 5.6.

Algorithm 10 Adaptive β for ADMM

INPUT: $q^{\text{th}} > 1$
Initialize $\beta^0 = L_2/(\Lambda_{\max}(B^TB) * \Lambda_{\max}(BB^T))$.
for $k = 0, 1, 2, \ldots$ **do**
  $q^k = \frac{\|A\mathbf{x}^{k+1}+B\mathbf{y}^{k+1}-\mathbf{b}\|}{\|A^TB(\mathbf{y}^{k+1}-\mathbf{y}^k)\|}$
  if $q^k \geq q^{\text{th}}$ or $q^k \leq \frac{1}{q^{\text{th}}}$ **then**
    $\beta^{k+1} \leftarrow \beta^k * \sqrt{q^k}$
  end if
end for

Additionally, for our distributed consensus learning (Alg.9), it is extremely easy to obtain $\Lambda_{\max}(B^TB)$ and $\Lambda_{\max}(BB^T)$. They are simply the maximum degree of the right nodes of the bipartite graph, as summarized in the following result.

Proposition 4. *Let matrix $B \in \mathbb{R}^{D|\mathcal{C}|\times D|\mathcal{V}_R|}$ be of $|\mathcal{C}|$ block-rows and $|\mathcal{V}_R|$ block-columns, with each row block having only one $-I$, and $\mathbf{0}$ for others (Figure 35). Then $\Lambda_{\max}(B^TB) = \Lambda_{\max}(BB^T) = \max\{\text{Degree}(v \in \mathcal{V}_R)\}$.*

5.6 Numerical Results

In this section, several sets of numerical examples will be used to: a) empirically demonstrate how ADMM's three degrees of freedom affect our proposed consensus learning algorithm; b) illustrate how well the proposed adaptive β updating strategy works. In addition, we

proposed a practical β that works quite well for simple ADMM instances where $A = I$ and $B = -I$.

5.6.1 Experimental Settings

In all examples presented in this section, we generate a dataset for the following distributed regression task:

$$\begin{aligned} \min_{\mathbf{x}} f(\mathbf{x}) &\equiv \sum_{i=1}^{N} (S_i^T \mathbf{x}_i - \mathbf{l}_i)^2, \\ \text{s.t. } \mathbf{x}_1 &= \mathbf{x}_2 = \ldots = \mathbf{x}_N, \end{aligned} \tag{151}$$

We assume that the total $48,000$ data samples are evenly distributed among $N = 24$ workers. Each worker i has $2,000$ samples of $D = 50$ dimensions. Components of the data matrix S_i of each worker are generated from the normal distribution $\mathcal{N}(0, 1)$. The real regression coefficients $\mathbf{x}_1 = \mathbf{x}_2 = \ldots = \mathbf{x}_{\text{true}} \in \mathbb{R}^D$ have 10% zeros, and each non-zero dimension is draw from the normal distribution $\mathcal{N}(0, 1)$. The dependent variables (labels) are perturbed by Gaussian white noise $\mathcal{N}(0, 10^{-4})$.

For comparison purposes, we consider the following communication topologies:

- *Complete bipartite graph.* The 24 workers are divided into two groups: 12 are on the left $\mathcal{V}_L$ and 12 on the right $\mathcal{V}_R$. Each worker communicates with all the other 12 workers on the other partition. It is $(12, 12)$-biregular.
- *Master-salve.* The 24 workers are divided into two groups of 1 and 23 workers each. The master communicates with all the 23 slaves on the other partition. It is $(23, 1)$- or $(1, 23)$-biregular.
- $(3, 3)$-*Biregular graph.* The bipartition of workers is the same as the complete bipartite. Each worker has the same degree 3.
- *Bucky spanning tree.* The 24 workers form a spanning tree, where is taken from a buckyball, as shown in Fig.37, where the red ones are on the left, and the yellow ones are on the right.
- *Ring.* A ring is also a $(2, 2)$-biregular graph.
- *Ring+1edge.* An additional edge of the longest chord is added to the ring, making it not biregular.

- *Chain.* A chain is the spanning tree with the largest diameter.

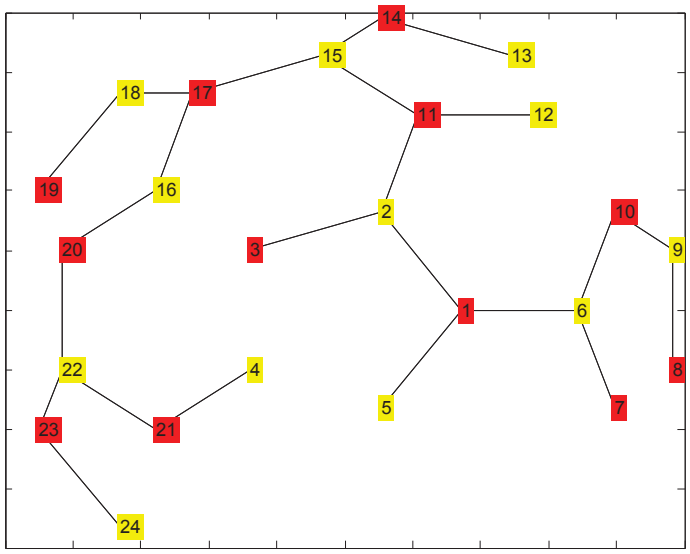

Figure 37: Buckyball spanning tree.

5.6.2 Varying β

We have already presented a very simple example in Section 5.5 showing that a bad choice of β can significantly slow down the convergence of ADMM. Now we use the complete bipartite communication topology to show that β is still a crucial parameter for the distributed consensus learning with more than 2 workers.

The primal residual $\|\mathbf{r}^k\|$ and dual residual $\|\mathbf{s}^k\|$ are plotted in Fig.38 and 39 as functions of both the number of iterations and β. We have several observations. First, both residuals converge linearly for any β values we tried from $10^{-2} \sim 10^2$, although some β converge faster than the others. This is expected, since our linear convergence rates in Theorem 13 and 14 are simple functions of β, no matter how large or small it is. Second, for small β, the primal residual $\|\mathbf{r}^k\|$ is larger than the dual $\|\mathbf{s}^k\|$, and for large β the reverse holds. Third, the "optimal" $\beta^* = 0.4467$ is neither too large nor too small. It is the parameter that achieves the lowest values for *both* $\|\mathbf{r}^k\|$ and $\|\mathbf{s}^k\|$, and these two lowest values are very close to each other. This observation provides some evidences for the effectiveness of our proposed strategy of adaptive β (Alg.9).

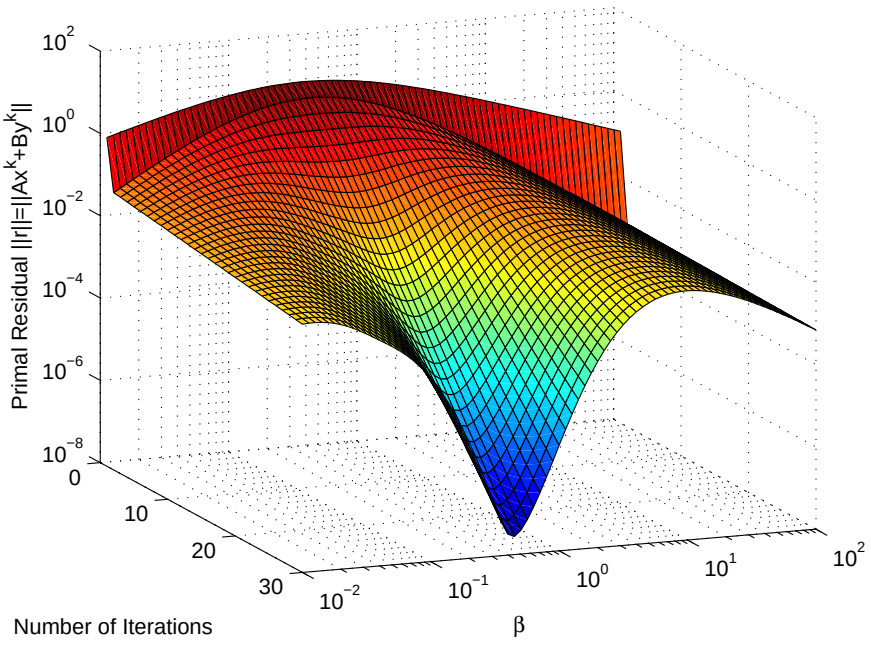

Figure 38: Primal residual as a function of β and number of iterations. Topology: complete bipartite graph.

5.6.3 Comparing Communication Topologies Using Optimal βs

As we discussed in Section 5.3.1, the three degrees of freedom of ADMM all contribute to the convergence speed of the algorithm. Their interplay is so complex that it is not easy to draw a clear conclusion of which communication topology is the "best". Here we simplify this problem by fixing the other two degrees and only explore the effects of communication topologies. For each topology, we seek the "optimal" β from a set of $1,000$ candidates ranging from $\beta^0/10$ to $100\beta^0$, where the formula for the underestimated β^0 is given in (150) and Proposition 4 can be used to calculate the maximum eigenvalues.

The fastest possible primal and dual convergences for each topology are plotted in Fig.40. Again we can observe that all residuals converge linearly, and the values of $\|\mathbf{r}^k\|$ and $\|\mathbf{s}^k\|$ are very close at the same iteration given the optimal β of each topology. It is also very clear that the complete bipartite and master-slave topologies converge at almost the same rate, and they are both faster than the others. This is an interesting observation, since the complete bipartite graph has 144 edges, which is higher than the master-slave's 23,

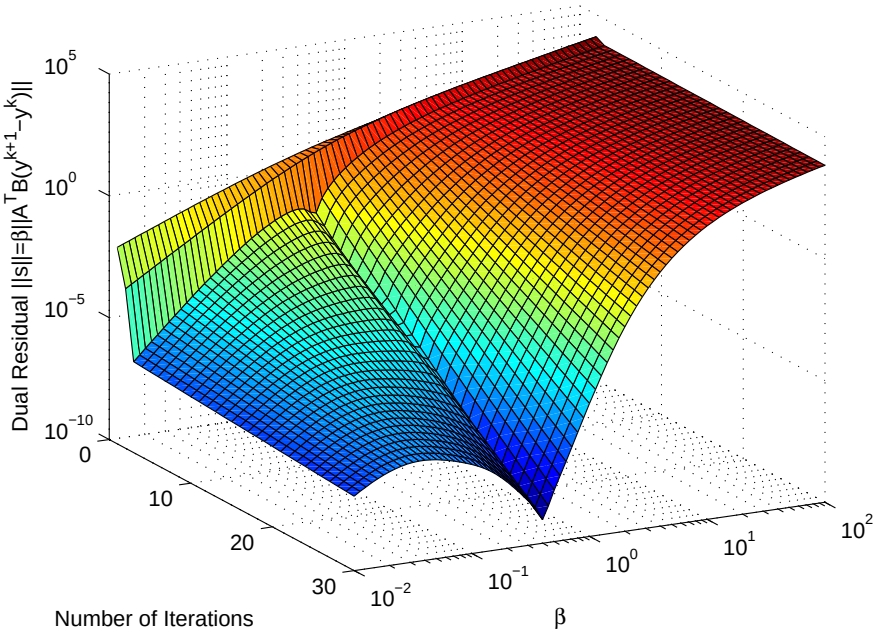

Figure 39: Dual residual as a function of β and number of iterations. Topology: complete bipartite graph.

however the master-slave have a higher bandwidth requirement for the master node than the complete bipartite where the bandwidth requirement is balanced for all workers. The $(3, 3)$-biregular graph is much faster than the bucky spanning tree, although they have the same maximum degree 3 for each bipartition. This might due to the fact that the spanning tree taken from the buckyball graph has a minimum degree 1 for some workers. The spanning tree is even slower than the $(2, 2)$-biregular ring, implying that a biregular graph might be preferred for the faster convergence rates of consensus learning. This preference can be also observed from the comparison between the ring and the ring+one edge, where more edges do not necessarily lead to faster rates. The chain topology is the slowest one, which is expected, since it has the smallest number of edges and the smallest minimum (1) and maximum (2) degrees.

5.6.4 Adaptive β using Alg.10

The above observations verify that an effective implementation of our consensus learning (Alg.9) heavily relies on a good β. Hence in the follows we use the distributed consensus

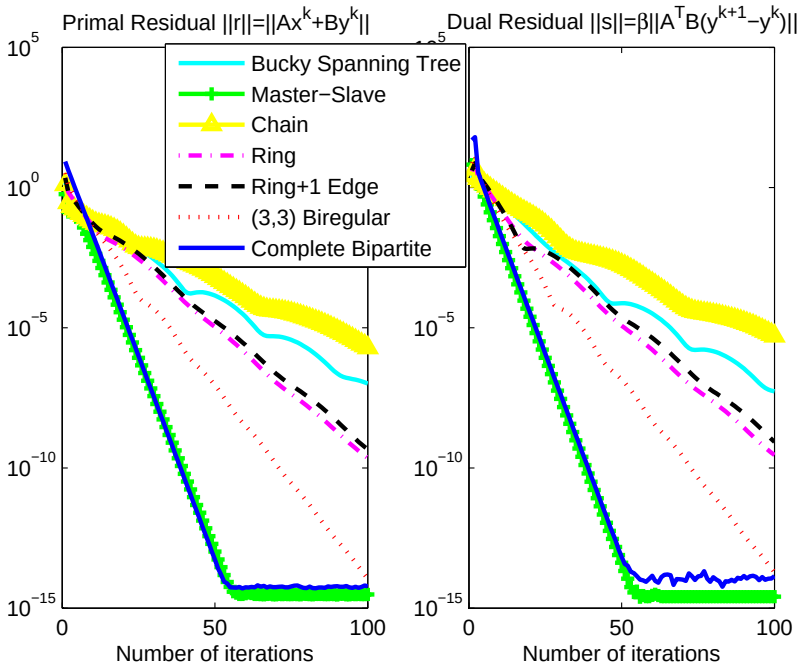

Figure 40: Primal and dual residuals using the optimal βs.

learning task as a testbed for our proposed adaptive β strategy (Alg.10). Note that this method is very general and can be used as a plug-in for other ADMM instances.

All the experimental settings are the same as Subsection 5.6.3, except that we replace the fixed "optimal" β with the adaptive strategy. As a comparison, we implemented He's adaptive β proposed in (He et al. [2000], Wang and Liao [2001]) using the parameters suggested in (Boyd et al. [2010]), and take the initial $\beta^0 = 1$ for all topologies. We plot the convergence history of the primal and dual residuals in Fig.41 and 42.

Comparing Fig.41 with Fig.40 one can observe that the proposed strategy for β works very well. The convergence rates are very close to those with "optimal" βs. Residuals for the master-slave topology are not monotonically decreasing, but the overall rates are still comparable with the optimal case, if not any faster. He's method (Fig.42) works reasonably well for some topologies, but is still much slower than our proposed method, except for the master-slave. One reason might be that the uninformative initial guess $\beta^0 = 1$ is improper, and it should be both data- and topology-dependent as we suggested in Alg.10.

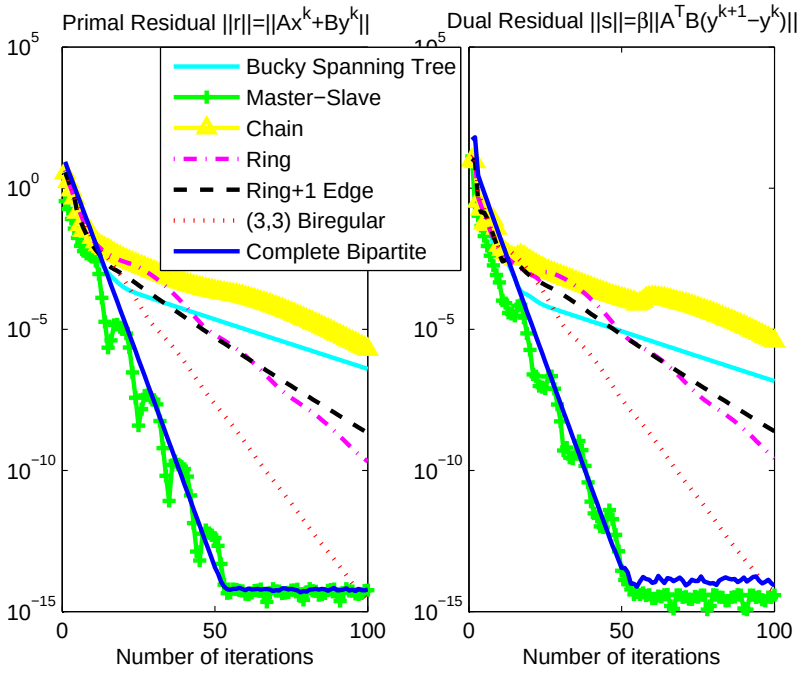

Figure 41: Primal and dual residuals using proposed Alg.10.

5.6.5 Changing the Updating Order

The third degree of freedom for ADMM is the order with which $\mathbf{x}$ and $\mathbf{y}$ are updated. Although we have no pointers coming directly from our theoretical results, empirically it is the weakest factor comparing with the communication topology and the value of β. We test it using the same settings as in Subsection 5.6.3. We observe that for all topologies except the master-slave, after changing the updating order, the changes of convergence rates are tiny, and the optimal β^* are essentially the same as before. For the master-slave topology, similar convergence rates can also be obtained, although we have to reduce the optimal β^* from 4.71 to 4.33.

5.6.6 Practical β for the Simple Case: $\mathbf{x} = \mathbf{y}$

In this last set of experiments, we present a practical β for the case where the constraint of ADMM is simply $\mathbf{x} = \mathbf{y}$, i.e. $A = I, B = -I$ and $\mathbf{b} = 0$. We found that taking the fixed penalty parameter

$$\beta = \sqrt{\mu_1 L_2} \tag{152}$$

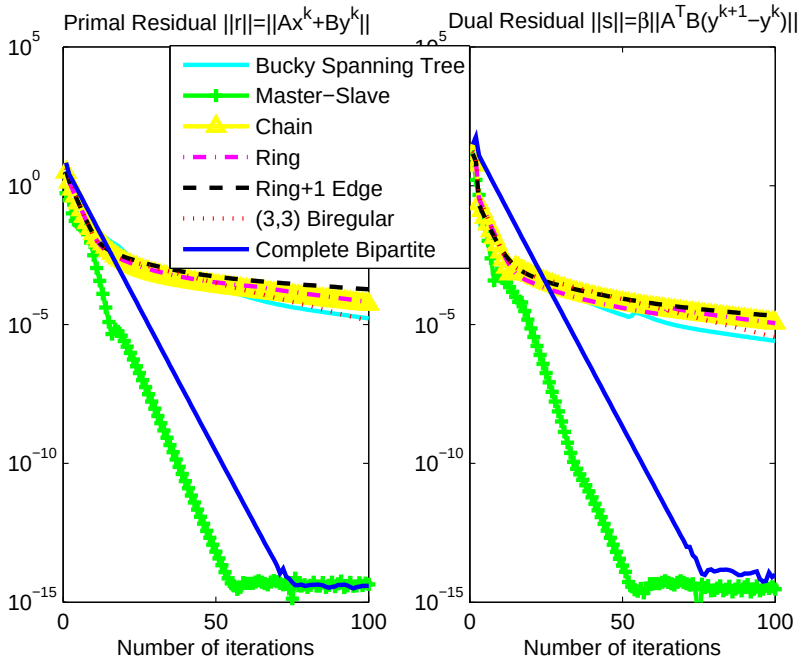

Figure 42: Primal and dual residuals using the method of (He et al. [2000], Wang and Liao [2001], Boyd et al. [2010]).

works quite well in practice although currently we do not have any theoretical evidence to support its effectiveness. To satisfy the smoothness and strongly-convex assumptions, we use the ridge regression $\min_{\mathbf{x}} \sum_{i=1}^{N}(\mathbf{x}^T\mathbf{s}_i - l_i)^2 + \frac{\alpha}{2}\|\mathbf{x}\|^2$ as our objective function. Putting it in ADMM's canonical form (132) we have $\theta_1(\mathbf{x}) = \sum_{i=1}^{N}(\mathbf{x}^T\mathbf{s}_i - l_i)^2$ and $\theta_2(\mathbf{y}) = \frac{\alpha}{2}\|\mathbf{y}\|^2$. We test (152) using $2,000$ samples of dimension 50. In this simulated dataset, $\mu_1 = 1,436.5$. Results for $\alpha = 1$ and $\alpha = 100$ are plotted in Fig.43 and 44. When $\alpha = L_2 = 1$, $\sqrt{\mu_1 L_2} = 37.90$, and the optimal β^* shown in Fig.43 is 39.64. When $\alpha = L_2 = 100$, $\sqrt{\mu_1 L_2} = 379.02$, and the optimal β^* shown in Fig.44 is 384.42.

5.7 Conclusions of this Chapter

In this chapter, we presented an ADMM-based consensus learning method for training distributed data samples in parallel. We used bipartite communication topologies to take advantage of ADMM's capacities in dealing with separable functions. We identify the three degrees of freedom in implementing this method: communication topology, penalty parameter β and the order for updating variables. In order to investigate the joint effects

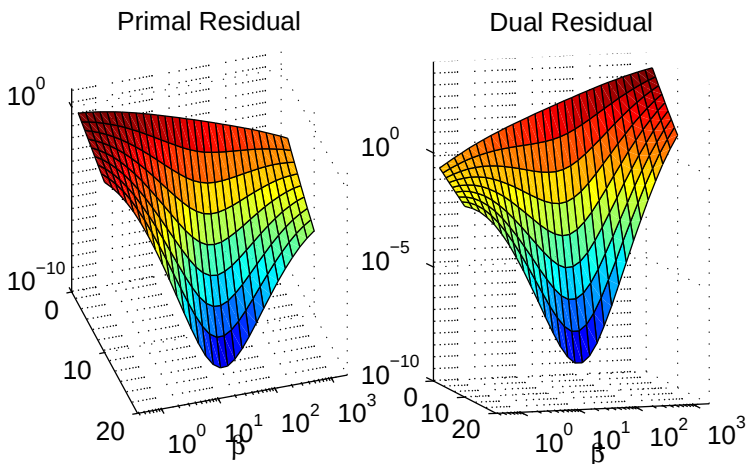

Figure 43: Ridge regression $\alpha = 1$.

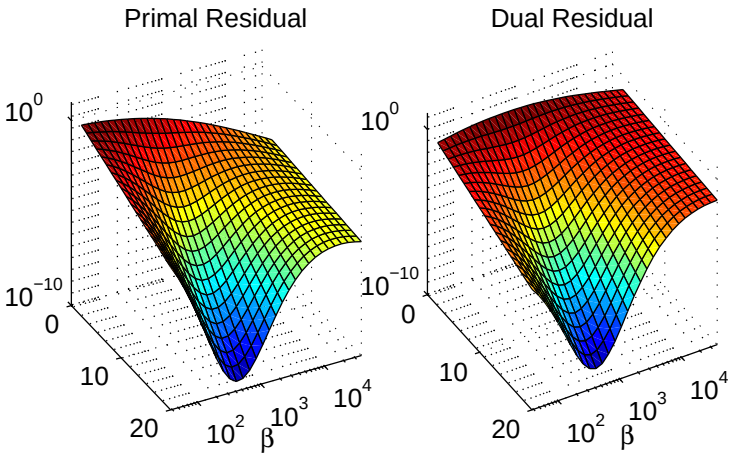

Figure 44: Ridge regression $\alpha = 100$.

of these factors, we provided an analysis of ADMM's convergence behavior. The analysis demonstrates that all the primal and dual variables enjoy a linear rate of convergence. Due to the difficulty in obtaining a very sharp rate from which the optimal β^* can be derived, we proposed a strategy for choosing β adaptively, with an underestimated initial guess β_0 that is derived from our bound. Numerical experiments show that β^* is achieved at a point where the norms of primal and dual residuals are close and decrease at the fastest rate. With β^*, the complete bipartite and the master-slave graphs converge fastest, followed by bi-regular graphs. The proposed strategy of adaptive β is very efficient.

There are several interesting directions that remain to be explored. A tighter and more

instructive bound is deserved. It is possible to extend our method to asynchronous variants. It is also promising to investigate the possibilities with assumptions weaker than Assumption 5. A potential application is the distributed consensus Lasso.

5.8 Appendix

5.8.1 Proof for Theorem 13

Proof. Invoking the KKT optimality conditions for (132),

$$\theta_1'(\mathbf{x}^*) - A^T\boldsymbol{\lambda}^* = 0,\ \theta_2'(\mathbf{y}^*) - B^T\boldsymbol{\lambda}^* = 0. \tag{153}$$

Invoking the optimality conditions for Line 1 and 2 of Alg.8,

$$\theta_1'(\mathbf{x}^{k+1}) - A^T\boldsymbol{\lambda}^k + \beta A^T(A\mathbf{x}^{k+1} + B\mathbf{y}^k - b) = 0 \tag{154}$$

and

$$\theta_2'(\mathbf{y}^{k+1}) - B^T\boldsymbol{\lambda}^k + \beta B^T(A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - b) = 0. \tag{155}$$

By the Lipshitz smoothness of θ_2 and (153,155) we have

$$\|\theta_2'(\mathbf{y}^{k+1}) - \theta_2'(\mathbf{y}^*)\| = \|B^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*)\| \leq L_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|, \tag{156}$$

hence by the definition of c (148) we have:

$$\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2 \leq \frac{L_2^2}{c}\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2. \tag{157}$$

By (153) and (154) we have

$$\|\theta_1'(\mathbf{x}^{k+1}) - \theta_1'(\mathbf{x}^*)\| = \|A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*) + \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k))\| \tag{158}$$

Combing (158) and the fact of strong-convexity

$$\|\theta_1'(\mathbf{x}^{k+1}) - \theta_1'(\mathbf{x}^*)\| \geq \mu_1\|\mathbf{x}^{k+1} - \mathbf{x}^*\|$$

we have

$$\begin{aligned}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 &\leq \frac{1}{\mu_1^2}\|A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*) + \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k))\|^2 \\ &= \frac{1}{\mu_1^2}\|(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*) + \beta B(\mathbf{y}^{k+1} - \mathbf{y}^k)\|_{AA^T}^2 \\ &\leq \frac{2\Lambda_{\max}(AA^T)}{\mu_1^2}\left[\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2 + \beta^2\|\mathbf{y}^{k+1} - \mathbf{y}^k\|_{B^TB}^2\right]\end{aligned} \tag{159}$$

Invoking Lemma 14 with $G = 0$ we have

$$\begin{aligned}
&\frac{\mu_1^2}{2\beta\Lambda_{\max}(AA^T)}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \\
&\leq \frac{1}{\beta}\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2 + \beta\|\mathbf{y}^{k+1} - \mathbf{y}^k\|^2_{B^TB} \\
&\overset{(136)}{\leq} \beta\|\mathbf{y}^k - \mathbf{y}^*\|^2_{B^TB} + \frac{1}{\beta}\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*\|^2 \\
&\quad - \beta\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2_{B^TB} - 2\mu_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 \\
&\quad - \frac{1}{\beta}\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\|^2 - 2\mu_1\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2
\end{aligned} \tag{160}$$

Rearranging the items of the above inequality we have

$$\begin{aligned}
&\mu_1\left(2 + \frac{\mu_1}{2\beta\Lambda_{\max}(AA^T)}\right)\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + \beta\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2_{B^TB} + 2\mu_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 \\
&+ \frac{1}{\beta}\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\|^2 \leq \beta\|\mathbf{y}^k - \mathbf{y}^*\|^2_{B^TB} + \frac{1}{\beta}\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*\|^2 = \|\mathbf{w}^k - \mathbf{w}^*\|^2_M.
\end{aligned} \tag{161}$$

Next we bound the right hand side of the above inequality. Denote $\tau > 0$ such that

$$\tau\|\mathbf{w}^k - \mathbf{w}^*\|^2_M \overset{(157)}{\leq} \tau\left[\frac{L_2^2}{\beta c} + \beta\Lambda_{\max}(B^TB)\right]\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 = 2\mu_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2, \tag{162}$$

and the formula of τ (147) follows. Combing Lemma 14 and (162) we have

$$\|\mathbf{w}^k - \mathbf{w}^*\|^2_M - \|\mathbf{w}^{k+1} - \mathbf{w}^*\|^2_M \geq \tau\|\mathbf{w}^{k+1} - \mathbf{w}^*\|^2_M, \tag{163}$$

and together with (161) the linear rate follows. □

5.8.2 Proof for Theorem 14

Proof. This result simply follows Lemma 14 and (162). □

CHAPTER VI

DISCUSSION AND FUTURE WORK

In this dissertation we investigate the design and analysis of scalable algorithms for machine learning. From the computational point of view, stochastic and distributed algorithms have been the main focuses of this work.

The main idea behind all of our algorithm design is to achieve low error rates for both the estimation and optimization errors. This idea stems from a deeper understanding on the main sources of the generalization error of a learning algorithm. We identify the lower bounds for each of these components and introduce the concept of statistically and computationally optimal learning algorithms.

The error decomposition is illustrated by some simple examples. We observe that when using the same basis optimization method e.g. gradient descent, the stochastic algorithm is almost always faster than its batch counterpart, given that the sample size is large enough and the samples are scanned for only one pass. When multiple epochs are allowed in the training process, the stochastic algorithm still outperforms when the noise level is low or moderate, even if the i.i.d. assumption does not hold anymore.

Since the slow $O(1/\sqrt{t})$ worst-case convergence rate for minimizing nonsmooth functions is of the same order as that of the estimation error (for general convex functions), we propose a stochastic smoothing method to alleviate the effect of this dominating factor. Our convergence analysis show that the proposed accelerated nonsmooth stochastic gradient descent algorithm achieves optimal rates under both convex and strongly convex assumptions, and they are also computationally optimal due to the one pass computational cost of stochastic algorithms. We also propose a "Batch-to-Online" conversion for online learning, and show that optimal regrets can be obtained. We will extend our method to constrained minimizations, as well as cases when the approximated function $\hat{f}()$ is not easily obtained by maximizing $\mathbf{u}$. Nesterov's excessive gap technique has the "true" optimal $1/t^2$

bound, and we will investigate the possibility of integrating it in our algorithm.

To take advantages of both the separable structures in machine learning problems and the scalability of the stochastic algorithms, we propose a stochastic alternating direction method of multipliers. It is applicable to nonsmooth loss functions, which is more general than the classic ADMM. We also demonstrate the rates of convergence for our algorithm under various structural assumptions of the stochastic function: $O(1/\sqrt{t})$ for convex functions and $O(\log t/t)$ for strongly convex functions. Compared to previous literature, we establish the convergence rate of ADMM, for the first time, in terms of both the objective value and the feasibility violation. A novel application named Graph-Guided SVM is proposed to demonstrate the usefulness of our algorithm. Stochastic ADMM is general enough to be extend to many other applications if the graphical-lasso prior can be introduced and the inter-feature relations can be explored.

The stochastic Frank-Wolfe algorithms proposed for nonlinear kernel machines can be regarded as a stochastic greedy algorithm. Extensive experiments show that they are much faster than the (batch) sequential minimal optimization and the online kernel SVM. Comparing with fast linear SVM solvers, kernel SFW still outperforms for only a small number of iterations. Future work is to further reduce the number of support vectors such that the per-iteration cost can be reduced.

Our convergence analysis indicates that the consensus ADMM based data-distributed learning method exhibits fast linear convergence rates for strongly convex problems. To make it optimal we identify the three degrees of freedom in implementing this method: communication topology, penalty parameter β and the order for updating variables. We proposed a strategy for choosing the penalty parameter β adaptively, with an underestimated initial guess β_0 that is derived from our bound. Numerical experiments show that the optimal β^* is achieved at a point where the norms of primal and dual residuals are close and decrease at the fastest rate. With β^*, the complete bipartite and the master-slave graphs converge fastest, followed by bi-regular graphs. Extending our method to asynchronous updates is very important for real-world applications. How to relax the smoothness assumption is another important direction to explore. A potential application is the distributed

consensus Lasso.

A few topics are listed below for future investigation.

1. This dissertation is mainly about the exploration of problem/function structures. Another rich source of ***structural information comes from the dataset*** itself. For example, in first-order methods, the Lipschitz constant of gradients or the modulus of strong convexity are very important in choosing stepsizes. How to estimate these constants onlinely and efficiently is an open question. Our recent work on noise-adaptive stepsizes is my first attempt along this direction (Ouyang and Gray [2012a]).

2. The outputs of stochastic algorithms are typically not as stable as deterministic ones, despite of their low cost for computation. This is not surprising, since the convergence guarantees of stochastic algorithms are always in expectations. ***Stabilizing stochastic algorithms*** by reducing the variance of results is preferred in many applications where robustness is a crucial consideration. Using more data samples is the most trivial way in achieving this goal. However, the question whether the stochastic results can be stabilized *algorithmically* still remains open. Empirically, an ergodic averaging on SGD can significant reduce the variance (Ouyang and Gray [2012b]), hence more theoretical analysis is deserved.

3. Most results in my thesis depend on the convex formulation of a problem. However, important ***nonconvex problems*** are almost everywhere in machine learning especially when latent models are involved. Examples are matrix factorization, neural networks, deep learning and variational inference for Bayesian statistics. Migrating stochastic/online algorithms and the corresponding analysis to these problems are extremely important and promising. As a starting point, simple nonconvex formulations such as alternating minimization for matrix factorizations will be analyzed as an canonical example, since our empirical observations show that the local optimality issues are only minor or moderate in these problems.

4. Replacing random sampling in Monte-Carlo methods by ***importance-based sampling*** is another idea to explore data information. An analogy is the difference between stochastic algorithms and greedy algorithms where the most important path is taken in each iteration. A naive greedy search for the best data sample take $O(N)$ time for N samples, while a smarter search algorithm might speed it up to $o(\log N)$ if approximate results are acceptable. The concept of *importance* here is defined broadly. In additional to the statistical importance of data samples, the capacities and reliabilities of distributed workers and communication networks can be also modeled under this concept.

5. The field of ***robust optimization*** (Nemirovski [2012]) pioneered by Dr. Nemirovski (collaborator and member of my thesis committee) has been successfully applied to many areas beyond the operations research, such as finance, manufacturing engineering, chemical engineering and medicine. Little attention has been attracted from the machine learning community until very recently (Caramanis et al. [2012]). In additional to the stochastic programming setting, robust optimization is the other promising computational framework that can handle the uncertainties of machine learning, where data samples are almost always corrupted by noises of unknown distributions. One of its most appealing benefits is its deterministic setting, i.e. *no i.i.d. assumptions are made upon the data distribution*, and instead an *uncertain-but-bounded* data model is made. Under this framework, one can obtain results with probabilistic guarantees like "the solution has the max probability δ of making a loss lager than ϵ", as opposed to the a good-in-expectation result.

Bibliography

Alekh Agarwal and John Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems 24*, pages 873–881. 2011.

Alekh Agarwal, Peter L. Bartlett, P. Ravikumar, and Martin J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *Information Theory, IEEE Trans.*, 2012.

Francis Bach and Eric Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *NIPS*, 2011.

O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

Peter L. Bartlett. Fast rates for estimation error and oracle inequalities for model selection. *Econometric Theory*, 24(2):545–552, 2008.

Peter L. Bartlett and Shahar Mendelson. Empirical minimization. *Probability Theory and Related Fields*, 135(3):311–334, 2006.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):193–202, 2009.

Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

Leon Bottou. Stochastic gradient descent 2.0. URL `http://leon.bottou.org/projects/sgd`.

Leon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Proceedings of NIPS*, 2008.

Leon Bottou and Yann LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 2005.

Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

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

Emmanuel J. Cands, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, June 2011.

C. Caramanis, S. Mannor, and H. Xu. Robust optimization in machine learning. In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for Machine Learning*. The MIT Press, 2012.

K. W. Chang, C. J. Hsieh, and Lin C. J. Coordinate descent method for large-scale l2-loss linear support vector machines. *Journal of Machine Learning Research*, 9:1369–1398, 2008.

George H-G. Chen and R. T. Rockafellar. Convergence rates in forward-backward splitting. *SIAM Journal on Optimization*, 7(2):421–444, 1997.

Gong Chen and Marc Teboulle. Convergence analysis of a proximal-like minimization algorithm using bregman functions. *SIAM J. on Optimization*, 3(3), 1993.

K. L. Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008.

P. L. Combettes and J. Pesquet. Proximal splitting methods in signal processing. In H. H. et. al. Bauschke, editor, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, chapter 10, pages 185–212. Springer, New York, 2011.

P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Model. Simul.*, 4(4):1168–1200, 2005.

Corinna Cortes and Vladimir N. Vapnik. Support vector networks. *Machine Learning*, 20 (3):273–297, 1995.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413Ű1457, 2004.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *arXiv*, 2010. URL `http://arxiv.org/abs/1012.1367`.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction. In *Proceedings of the 28th International Conference on Machine Learning*, pages 713–720, June 2011.

W. Deng and W. Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. Technical Report TR12-14, Rice University CAAM Technical Report, 2012.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, (10):2899–2934, 2009.

John Duchi, Alekh Agarwal, and Martin Wainwright. Distributed dual averaging in networks. In *Advances in Neural Information Processing Systems 23*, pages 550–558. 2010.

John Duchi, Peter L. Bartlett, and Martin J. Wainwright. Randomized smoothing for stochastic optimization. *arXiv*, 2011. URL `http://arxiv.org/abs/1103.4296`.

John Duchi, Alekh Agarwal, and Martin J. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Trans on Automatic Control*, 57(3):592–606, 2012.

J. Eckstein and D. P. Bertsekas. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3): 293–318, 1992.

R. E. Fan, P. H. Chen, and Lin C. J. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

J. Friedman and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2007.

D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*. North-Holland: Amsterdam, 1983.

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

R. Glowinski and A. Marroco. Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualite, d'une classe de problems de dirichlet non lineares. *Revue Francaise d'Automatique, Informatique, et Recherche Operationelle*, 9(2), 1975.

R. Glowinski and P. L. Tallec. *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*. Studies in Applied and Numerical Mathematics. SIAM, 1989.

D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions, 2010. URL `http://arxiv.org/abs/0912.4571`.

T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM J. Imaging Sci.*, 2(2):323–343, 2009.

Jacques Guelat and Patrice Marcotte. Some comments on wolfe's 'away step'. *Mathematical Programming*, 35:110–119, 1986.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. In *COLT*, 2011.

B. He and X. Yuan. On the $o(1/n)$ convergence rate of the douglas-rachford alternating direction method. *SIAM J. Numer. Anal.*, 50(2):700–709, 2012a.

B. He and X. Yuan. On non-ergodic convergence rate of douglas-rachford alternating direction method of multipliers. 2012b.

B. S. He, H. Yang, and S. L. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and Applications*, 106(2):337–356, 2000.

Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. http://arxiv.org/abs/1208.3922, 2012.

C. J. Hsieh, K. W. Chang, and C. J. Lin. A dual coordinate descent method for large-scale linear svm. In *Proc. 25th Intl. Conf. on Machine Learning (ICML)*, 2008.

Chonghai Hu, James T. Kwok, and Weike Pan. Accelerated gradient methods for stochastic optimization and online learning. In *NIPS 22*, 2009.

Peter J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, 1964.

Thorsten Joachims. Making large-scale svm learning practical. In *Advances in Kernel Methods: Support Vector Learning*. The MIT Press, 1999.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proc. ACM Conf. on Knowledge Discovery and Data Mining (KDD)*, 2002.

Thorsten Joachims. Training linear svms in linear time. In *Proc. ACM Conf. on Knowledge Discovery and Data Mining (KDD)*, 2005.

S. Kim, K. Sohn, and E. P. Xing. A multivariate regression approach to association analysis of a quantitative trait network. *Bioinformatics*, 25(12):204–212, 2009.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Trans. on Signal Processing*, 52(8):2165–2176, 2004.

S. Kogan, D. Levin, B. R. Routledge, J. S. Sagi, and N. A. Smith. Predicting risk from financial reports with regression. In *NAACL-HLT 2009, Boulder, CO*, 2009.

Harold J. Kushner and G. George Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2nd edition, 2003.

G. Lan and S. Ghadimi. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, i: a generic algorithmic framework. *SIAM J. on Optimization*, 2011.

Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 2010. doi: DOI10.1007/s10107-010-0434-y.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, pages 777–801, 2009.

P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM J. on Numerical Analysis*, 16(6):964–979, 1979.

G. Mateos, J. A. Bazerque, and G. B. Giannakis. Distributed sparse linear regression. *IEEE Trans on Signal Processing*, 58(10):5262 –5276, oct. 2010.

Renato D. C. Monteiro and B. F. Svaiter. Iteration-complexity of block-decomposition algorithms and the alternating minimization augmented lagrangian method. Technical report, Georgia Institute of Technology, 2010.

Joao F. C. Mota, Joao M. F. Xavier, Pedro M. Q. Aguiar, and Markus Puschel. D-admm: A communication-efficient distributed algorithm for separable optimization. arXiv:1202.2805, 2012.

Indraneel Mukherjee, Cynthia Rudin, and Robert E. Schapire. The rate of convergence of adaboost. In *Proceedings of the 24th Annual Conference on Learning Theory*, JMLR: Workshop and Conference Proceedings 19 (2011), pages 537–557, 2011.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Trans on Automatic Control*, 54(1):48–61, 2009.

A. Nemirovski. *Lectures on Robust Convex Optimization*. ISYE, Georgia Institute of Technology, 2012. URL `http://www2.isye.gatech.edu/~nemirovs/RO_LN.pdf`.

A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley and Sons, 1983.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. on Optimization*, 19(4):1574–1609, 2009.

Yurii Nesterov. A method for solving a convex programming problem with rate of convergence $o(1/k^2)$. *Soviet Math. Doklady*, 269(3):543–547, 1983.

Yurii Nesterov. *Introductory Lectures on Convex Optimization, A Basic Course*. Kluwer Academic Publishers, 2004.

Yurii Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM J. Optim.*, 16(1):235–249, 2005a.

Yurii Nesterov. Smooth minimization of non-smooth functions. *Math. Program., Ser. A*, 103:127–152, 2005b.

Yurii Nesterov. Gradient methods for minimizing composite objective function. Technical Report CORE DISCUSSION PAPER 2007/76, 2007a.

Yurii Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110(2):245–259, 2007b.

E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, MIT Artificial Intelligence Laboratory, 1997.

Hua Ouyang and Alexander Gray. Nasa: Achieving lower regrets and faster rates via adaptive stepsizes. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining (KDD 2012)*, August 2012a.

Hua Ouyang and Alexander Gray. Stochastic smoothing for nonsmooth minimizations: Accelerating sgd by exploiting structure. In John Langford and Joelle Pineau, editors, *Proceedings of the the 29th International Conference on Machine Learning (ICML 2012)*, July 2012b.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*. The MIT Press, 1999.

Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. on Control and Optimization*, 30(4):838–855, 1992.

Joel B. Predd, Sanjeev R. Kulkarni, and H. Vincent Poor. Distributed learning in wireless sensor networks. In Ananthram Swami, Qing Zhao, Yao-Win Hong, and Lang Tong, editors, *Wireless Sensor Networks: Signal Processing and Communications Perspectives*. Wiley, 2007.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

David Saad. *On-Line Learning in Neural Networks*. Cambridge University Press, 1998.

B. Scholkopf, J. C. Platt, J. Shawe-Taylor, and A. J. Smola. Estimating the support of a high-dimensional distribution. *Neural Computation*, (13):1443–1471, 2001.

Nicol N. Schraudolph, Jin Yu, and Simon Gunter. A stochastic quasi-newton method for online convex optimization. In *Proceedings of AISTATS*, 2007.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*, 2007.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic convex optimization. In *COLT*, 2009.

Ohad Shamir. Making gradient descent optimal for strongly convex stochastic optimization. In *OPT 2011*, 2011. URL `http://arxiv.org/abs/1109.5647`.

Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, Philadelphia, 2009.

A. J. Smola and B. Scholkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2 Technical Report Series, 1998.

A. J. Smola and B. Scholkopf. Sparse greedy matrix approximation for machine learning. In *Proc. 17th Intl. Conf. on Machine Learning (ICML)*, 2000.

Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors. *Optimization for Machine Learning*. Neural Information Processing series. The MIT Press, 2011.

Taiji Suzuki. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *Proceedings of ICML*, 2013.

C. H. Teo, Q. Le, A. J. Smola, and S. V. N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proc. ACM Conf. on Knowledge Discovery and Data Mining (KDD)*, 2007.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B*, 67(1):91–108, 2004.

R. J. Tibshirani and J. Taylor. The solution path of the generalized lasso. *Annals of Statistics*, 39(3):1335–1371, 2011.

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

I. W. Tsang, J. T. Kwok, and P. M. Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *SIAM J. Optim.*, 2008.

Aad van der Vaart and Jon Wellner. *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer, 1996.

V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York Incorporated, 2000.

Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.

H. Wang and A. Banerjee. Online alternating direction method. In *Proceedings of ICML*, 2012.

S. L. Wang and L. Z. Liao. Decomposition method with a variable parameter for a class of monotone variational inequality problems. *Journal of Optimization Theory and Applications*, 109(2):415–429, 2001.

P. Wolfe. A duality theorem for non-linear programming. *Quarterly of Applied Mathematics*, 19:239–244, 1961.

P. Wolfe. Convergence theory in nonlinear programming. In *Integer and Nonlinear Programming*. North-Holland Publishing Company, 1970.

S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *JMLR*, 11:2543–2596, 2010.

Wei Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv*, 2011. URL `http://arxiv.org/abs/1107.2490`.

J. Yang and X. Yuan. Linearized augmented lagrangian and alternating direction methods for nuclear norm minimization. *Mathematics of Computation*, 2012. doi: http://dx.doi.org/10.1090/S0025-5718-2012-02598-1.

J. Yang and Y. Zhang. Alternating direction algorithms for ℓ_1-problems in compressive sensing. *SIAM J. on Scientific Computing*, 33(1):250–278, 2011.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proc. 21st Intl. Conf. on Machine Learning (ICML)*, 2004.

X. Zhang, M. Burger, and S. Osher. A unified primal-dual algorithm framework based on bregman iteration. *J. of Scientific Computing*, 46(1):20–46, 2011.

Hao Zhu, G. B. Giannakis, and A. Cano. Distributed in-network channel decoding. *IEEE Trans on Signal Processing*, 57(10):3970 –3983, 2009.

VITA

Hua Ouyang is a Ph.D. candidate in the School of Computational Science and Engineering, College of Computing, Georgia Tech. He received his M.Phil. from the Chinese University of Hong Kong in 2007, and B.Eng. from Huazhong University of Science and Technology, China, in 2003. He worked as an intern at IBM T. J. Watson Research Center, Hawthorne, NY in 2010. He was the recipient of the 2010 best student paper award from the Statistical Computing Section, American Statistical Association. His primary research interests include large scale machine learning, optimization, computational geometry, information retrieval and computer vision.