Fast, Practical and Scalable First-Order Methods for Modern Machine Learning Problems

ZHOU, Kaiwen

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
December 2022

Thesis Assessment Committee

Professor WONG Man Hon (Chair)
Professor CHENG James (Thesis Supervisor)
Professor SO Anthony Man Cho (Committee Member)
Professor KE Kelly Yiping (External Examiner)

Abstract of thesis entitled:
Fast, Practical and Scalable First-Order Methods for Modern Machine Learning Problems
Submitted by ZHOU, Kaiwen
for the degree of Doctor of Philosophy in Computer Science and Engineering
at The Chinese University of Hong Kong in December 2022

In recent years, due to the increasing dimensionality of optimization problems in the machine learning community, first-order methods—basically gradient descent and its variants—become a popular choice of optimizer. The reason is that these methods are known to be dimension-free, i.e., their iteration complexity is independent of dimensionality. Classic complexity theory focuses on establishing worst-case convergence guarantees for these first-order methods. This analysis is traditionally conducted based on abstract mathematical intuition of the experts. A recent trend in the optimization community focuses on using computer-aided tools to assist the worst-case analysis or to discover new methods. The idea is to formulate the worst-case analysis as another optimization problem, which is usually a semidefinite program (SDP). Solving the SDP gives us numerics that provide hints for conducting the worst-case analysis. This methodology is promising for its capability of discovering tight proofs that are hard or even impossible for human experts to find with bare hands. With the help of this methodology, in this thesis, we describe several of our algorithmic discoveries that have the properties of being (i) *fast*, sometimes the theoretically fastest known scheme in its setting, (ii) *practical*, i.e., being implementable without impractical algorithmic components, being memory-efficient and has a simple intuitive scheme, and (iii) *scalable*, works in a restrictive asynchronous lock-free setting that fully utilizes the parallel computing architectures, which benefits large-scale machine learning tasks.

Optimizing neural networks, on the other hand, does not directly benefit from this methodology due to its overly complicated objective structure. We show that we can gain some insight from our algorithmic discoveries, and then empirically extend them to the deep learning setting, which results in new schemes that are efficient and competitive for deep learning tasks. For the theoretic side of neural network optimization, we also make some initial attempt on deriving theoretically grounded neural network optimizers that are guaranteed to converge to an approximated stationary point.

摘要：

隨著近幾年機器學習領域中優化問題維度的不斷增大，以梯度下降及其變種爲代表的一階優化演算法成爲了機器學習領域中最熱門的優化器。這其中的原因在於這些方法的迭代複雜度與維數無關。在傳統的複雜度理論中，對於一階優化演算法的理論分析專註於建立最壞情況下的迭代複雜度。相關的理論推導也往往高度依賴於優化專家的數學直覺。近幾年，在運籌優化領域裏興起了一類使用計算機的數值工具來輔助複雜度分析甚至是探索新演算法的研究思路。這類思路可以概括爲將優化演算法的理論分析描述爲另一個優化問題（往往是半正定編程問題），然後使用數值優化的工具來求解這個優化問題，其返回的數值解可以爲理論分析提供指引。這一套方法論的獨到之處在於它能夠發現一些相當複雜的，人類難以徒手推導得到的證明。本論文受到這套方法論的指引，提出了幾個全新的演算法。這些演算法擁有以下幾種優勢中的一點或多點：（一）快速，其中幾個演算法是在它們對應的理論場景中目前收斂速度最快的演算法；（二）實際，即演算法中不包含無法實施的部分，且存儲需求小，演算法框架清晰簡潔；（三）高可擴展性，即演算法可以在條件苛刻的異步無鎖分佈式並行計算框架下實施，可以提升超大規模機器學習任務的訓練效率。

然而，對於神經網路的優化，由於其目標函數的結構過於複雜，目前還無法直接受益於此方法論。本論文中對於神經網路優化，我們利用之前所提的演算法帶來的指引，將這些演算法技術擴展至深度學習訓練當中，最終提出了幾個具有競爭力的高效神經網路優化器。在神經網路的理論探索方面，本論文也做出了一些初步的探索，目標是構建具有理論上收斂保證的神經網路優化器。

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Prof. James Cheng, for providing the support and guidance during my PhD study, and for his patience and knowledge in helping me revising my papers. He has taught me how to become an excellent researcher. I would also like to specially thank Prof. Anthony Man-Cho So for having multiple fruitful discussion with me, and for his insightful comments and suggestion on revising my papers. His outstanding mathematical intuition inspired me in all the time of my research and writing of this thesis.

I also learned a lot from my collaborators during the PhD study. I would like to thank (in alphabetical order) Yongqiang Chen, Xinyan Dai, Qinghua Ding, Ruize Gao, Chenhan Jin, Tian Lai, Kaili Ma, Jiongxiao Wang, Binghui Xie, Han Yang and Yonggang Zhang for their extremely useful discussion and important contribution in our coauthored papers. In addition, it is my great pleasure to have other teammates in the Husky data lab to be my close friends.

Last but not the least, I give my special thanks to my parents, Guizi Zhou and Meiqin Zhou, for their continuous support and encouragement during my PhD study. No words could express my gratitude. I can only try my best to be a good son, and love you back in the same way you love me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Optimization problems naturally arise in various quantitative scientific disciplines, such as computer science, operation research, physics and economics. It is usually the case that these problems have distinct structures (i.e., different objective function properties, constraints and optimality measures). This fact makes optimization a research area where there is not a single best performing method in every situation. Usually, practitioner chooses an optimizer based on the task at hand. Thus, designing algorithms tailored for various optimization tasks has been the focus of optimization research for years.

In the last decade, due to the surge of machine learning applications, research on designing efficient machine learning optimizers has gained even more popularity. Machine learning models, especially those composed of deep neural networks, typically have millions or even billions of parameters, which makes the parameter-training task an extremely high-dimensional optimization problem. Due to this fact, research in this area mainly focuses on developing first-order methods, i.e., gradient descent (GD) and its variants, as they are known to be dimension-free, i.e., their iteration complexity is independent of dimensionality.

The theme of this thesis is to design new first-order algorithms for various machine learning tasks such as the training of logistic regression, ridge regression, LASSO, SVM and deep neural networks. Since the development will mainly focus on the theoretic properties of optimization algorithms, in the following sections, we provide some necessary background, do a brief literature review and define the general notations of this thesis.

1.1 Machine Learning Problems

The basic formulation of an unconstrained optimization problem is: Given a function $f : \mathbb{R}^d \to \mathbb{R}$, we solve for[1] $x^\star = \arg\min_{x \in \mathbb{R}^d} f(x)$, where $\mathbb{R}^d$ is the d-dimensional Euclidean space. Without any regularity imposed on f, it is impossible to obtain a (theoretically) meaningful solution $x^\star$ [107]. Before diving into various regularity conditions of f, let us first look at some simple machine learning problems. Below we present the training objectives of several classic binary classification tasks. This training process is called the empirical risk minimization [15]. We denote $\{a_i \in \mathbb{R}^d, b_i \in \{-1, +1\}\}_{i=1}^n$ as the data vectors and labels in the training dataset (totally n data samples), and $\lambda > 0$ is the amount of regularization.

- Binary ℓ_2-logistic regression:

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp\left(-b_i \langle a_i, x \rangle\right)\right) + \frac{\lambda}{2} \|x\|^2,$$

  where $\|\cdot\|$ is the ℓ_2-norm (or Euclidean norm) and $\langle \cdot, \cdot \rangle$ is the inner product.

- Ridge regression:

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} \left(\langle a_i, x \rangle - b_i\right)^2 + \frac{\lambda}{2} \|x\|^2.$$

- LASSO (Least Absolute Shrinkage and Selection Operator [147]):

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} \left(\langle a_i, x \rangle - b_i\right)^2 + \lambda \|x\|_1,$$

  where $\|\cdot\|_1$ is the ℓ_1-norm.

- SVM (Support Vector Machine [23]):

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \max\left\{0, 1 - b_i \langle a_i, x \rangle\right\} + \frac{\lambda}{2} \|x\|^2.$$

[1]We only consider unconstrained problems in this thesis, since it is the most common type of problems in machine learning community. We do not use boldface letters to represent vector, which is a convention in optimization community.

We would usually hope for a single optimizer that works reasonably well for a class of similar tasks such as the above ones, although faster algorithms may exist if we only focus on one specific objective. The trade-off is that: The more problem structure we utilize, the faster the derived algorithm could possibly be, but the less likely it is for the algorithm to generalize to other problems. Two common properties of the above objective functions are that they all have a finite-sum structure:[2]

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{Finite-Sum}$$

and that they are all convex (f and each f_i):

Definition 1 (Convexity)**.** *A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be convex if for all $0 \le t \le 1$ and all $x, y \in \mathbb{R}^d$, it holds that*

$$f(tx + (1-t)y) \le tf(x) + (1-t)f(y).$$

Without the finite-sum structure, general convex optimization has been studied for decades, and it is widely regarded that convex problems are well-solved. However, due to the advances in machine learning, finite-sum convex optimization has gained a lot of attention recently. Since empirical risk minimization plays a key role in almost all the machine learning applications, including the training of deep neural networks [50], devising efficient optimizers for finite-sum objectives becomes a very popular research topic. Clearly, neural networks are non-convex in general. We will discuss them shortly. Another common objective property of binary ℓ_2-logistic regression, ridge regression and SVM is that they are all λ-strongly convex (f and each f_i):

Definition 2 (μ-strong convexity)**.** *A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be μ-strongly convex if for all $x, y \in \mathbb{R}^d$, it holds that*

$$f(x) \ge f(y) + \langle \mathcal{G}, x - y\rangle + \frac{\mu}{2}\left\|x - y\right\|^2,$$

where $\mathcal{G} \in \partial f(y)$, the set of sub-gradient [107] of f at y .

LASSO is not strongly convex in general due to its ℓ_1-regularizer. As the name suggests, strong convexity often leads to faster convergence of the optimizers both empirically and theoretically. Another key factor affecting the convergence rate is the differentiability. L-smoothness is the typical regularity condition for continuously differentiable functions, defined as follows:

[2]Here we absorb the regularizer $\|x\|^2$ or $\|x\|_1$ into the summation as they are independent of i.

Definition 3 (L-smoothness)**.** *A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be L-smooth if its gradient is L-Lipschitz continuous, i.e., $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$ for all $x, y \in \mathbb{R}^d$. Equivalently, f is L-smooth if for all $x, y \in \mathbb{R}^d$, it satisfies that*

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

The objective functions (f and each f_i) of ℓ_2-logistic regression and ridge regression are L-smooth, and they also have simple closed-form expressions for their smoothness constants L. The objectives of LASSO and SVM are not differentiable due to the ℓ_1-regularizer and the max operator, respectively. For non-differentiable objectives, we usually characterize them using the Lipschitz continuity condition:

Definition 4 (L_0-Lipschitz continuity)**.** *A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be L_0-Lipschitz continuous if for all $x, y \in \mathbb{R}^d$, $|f(x) - f(y)| \leq L_0 \|x - y\|$.*

Thus far, we have introduced the regularity conditions that are satisfied by binary ℓ_2-logistic regression, ridge regression, LASSO and SVM. However, an important class of machine learning problems is not discussed, that is neural network optimization. Due to the flexibility in its design, neural network is non-smooth and non-convex in general. It seems that we could only use the Lipschitz continuity to characterize them. In fact, it is still an open question on how to identify useful properties of neural networks that can be used to analyze or construct algorithms [97, 13]. We discuss the existing strategies and some theoretic development in neural network optimization in details in Chapters 5 and 6. We define the following notations which will be used throughout the thesis.

Notations. We let $[n]$ denote the set $\{1, 2, \ldots, n\}$, $\mathbb{E}$ denote the total expectation and $\mathbb{E}_{i_k}$ denote the expectation with respect to a random sample i_k. We use $[x]_v$ to denote the coordinate v of the vector $x \in \mathbb{R}^d$, and for a subset $T \subseteq [d]$, $[x]_T$ denotes the collection of $[x]_v$ for $v \in T$. We let $\lceil \cdot \rceil$ denote the ceiling function, I_d denote the identity matrix and $(\alpha)_+ \triangleq \max\{\alpha, 0\}$. We define $\kappa \triangleq \frac{L}{\mu}$, which is always called the condition ratio.

1.2 Literature on Finite-Sum Optimization

There is a lot of recent progress on optimizing objective functions with a finite-sum structure. Here we briefly summarize the existing literature on finite-sum optimization where the objective function is assumed to be L-smooth and μ-strongly convex (satisfied by ℓ_2-logistic regression and ridge regression), which is a typical theoretic setting considered in these works. We mainly discuss works that focus on improving the *worst-case* convergence rates under this setting.

The finite-sum problem with $n = 1$ is the classic smooth strongly convex setting. Standard analysis shows that for this problem, GD with $\frac{2}{L+\mu}$ stepsize converges linearly at a $(\frac{\kappa-1}{\kappa+1})^2$ rate (see the textbook [109]). The heavy-ball method [123] fails to converge globally on this problem [86]. The celebrated Nesterov Accelerated Gradient method (NAG) is proven to achieve a faster $1 - 1/\sqrt{\kappa}$ rate [109]. This rate remains the fastest one until recently, Scoy et al. [134] proposed the Triple Momentum method (TM) that converges at a $(1-1/\sqrt{\kappa})^2$ rate. Numerical results in Lessard and Seiler [85] suggest that this rate is not improvable. In terms of reducing $\|x - x^\star\|^2$ to ϵ, TM is stated to have an $O\big((\sqrt{\kappa}/2)(\log\frac{1}{\epsilon} + \log\sqrt{\kappa})\big)$ iteration complexity (cf. Table 2, [134]) compared with the $O(\sqrt{\kappa}\log\frac{1}{\epsilon})$ complexity of NAG.

In the general convex setting, recent works [68, 8, 69] propose new schemes such as OGM, G-OGM that have lower complexity than the original NAG. Several of these new schemes were discovered based on the recent works that use semidefinite programming to study worst-case performances of first-order methods, which will be discussed in details in the following sections. Starting from the performance estimation framework introduced in Drori and Teboulle [37], many different approaches and extensions have been proposed [86, 141, 145, 144, 143].

For the $n \geq 1$ case, stochastic gradient descent (SGD) [126], which uses component gradients $\nabla f_i(x)$ to estimate the full gradient $\nabla f(x)$, achieves a lower iteration cost than GD. However, SGD only converges at a sub-linear rate. To fix this issue, various variance reduction techniques have been proposed recently, such as SAG [130, 133], SVRG [64, 152], SAGA [29], SDCA [136] and SARAH [111]. Inspired by the Nesterov's acceleration technique, accelerated stochastic variance-reduced methods have been proposed in pursuit of the lower bound $\Omega(n+\sqrt{n\kappa}\log\frac{1}{\epsilon})$ [151], such as Acc-Prox-SVRG [113], APCG [93], ASDCA [137], APPA [44], Catalyst [91], SPDC [156], RPDG [79], Point-SAGA [27] and Katyusha [4]. Among these methods, Katyusha and Point-SAGA, representing the first two directly accelerated incremental methods, achieve the fastest rates. Point-SAGA leverages a more powerful incremental proximal operator oracle. Katyusha introduces the idea of negative momentum, which serves as a variance reducer that further reduces the variance of the SVRG estimator. This construction motivates several new accelerated methods [158, 3, 81, 77, 159, 160].

1.3 Worst-Case Analysis

To examine the efficiency of an optimizer for a class of optimization problem, a direct thought is that we could conduct multiple numerical experiments and observe its performance. However, since it is usually hard or even impossible to exhaustively evaluate the performance on every instance of the optimization problem, numerical experiments may not be able to provide a strong performance guarantee. This is the situation where we resort to worst-case convergence analysis. Take GD [19] as an example. GD takes an initial guess $x_0 \in \mathbb{R}^d$ and iteratively updates: $x_{k+1} = x_k - \eta \nabla f(x_k)$, for $k = 0, 1, \dots, K-1$, where $\eta > 0$ is the step size. Suppose that we use GD to minimize an L-smooth and convex objective function f. The worst-case analysis is conducted by formulating the properties in Definitions 1 and 3 over the sequence of GD $\{x_0, x_1, \dots, x_K\}$ and $x^\star$, and then carefully manipulating the inequalities to produce an upper bound of the form [12]:

$$f(x_K) - f(x^\star) \leq \frac{L \left\| x_0 - x^\star \right\|^2}{2K}. \tag{1.1}$$

This upper bound tells us that no matter which objective instance f we are optimizing, as long as it is L-smooth and convex, after K iterations of GD, the function value gap $f(x_K) - f(x^\star)$ is guaranteed to be bounded by $\frac{L\|x_0 - x^\star\|^2}{2K}$, and this upper bound goes to 0 and $K \to \infty$. This type of inequality is usually called the non-asymptotic worst-case convergence guarantee, as it characters the convergence of the function value gap $f(x_K) - f(x^\star)$ at any iteration K.

All the theoretic discussion in this thesis aims to establish convergence guarantees of this form. Traditionally, deriving such guarantee relies on brainstorming and abstract mathematical intuition of the optimization experts. It is then natural to ask:

Is there a more principled way to conduct worst-case analysis?

1.4 Computer-Aided Worst-Case Analysis

A recent trend in optimization community focuses on using computer-aided approaches to assist the worst-case analysis, which is pioneered by the work [37]. The development in Chapters 2, 3, 4 are all based on or inspired by such methodology. The main idea is to formulate the worst-case analysis as another optimization problem and use existing solvers to obtain a numerical solution. Such numerics can inspire a symbolic worst-case analysis. To give an example, here we formulate the worst-case analysis of GD as an optimization problem (which is called the performance estimation problem (PEP) [37]):

$$\begin{aligned}
\max_{f:\mathbb{R}^d\to\mathbb{R}} \quad & f(x_K) - f(x^\star) \\
\text{subject to} \quad & \text{function } f \text{ is } L\text{-smooth and convex}, \\
& x_{k+1} = x_k - \eta \nabla f(x_k), k = 0, \ldots, K-1, \\
& \|x_0 - x^\star\| \le R_0, \\
& x_0, \ldots, x_K, x^\star \in \mathbb{R}^d.
\end{aligned} \tag{P1}$$

The intuition of (P1) is to find the "worst" function that maximizes the function value gap at K-th iterate of GD. At first glance, the constraint "function f being L-smooth and convex" looks impossible to solve. A direct thought is to relax it. Denoting the set $I = \{x_0, \ldots, x_K, x^\star\}$, we can relax (P1) as

$$\begin{aligned}
\max_{\substack{\nabla f(x_0),\ldots,\nabla f(x_K)\in\mathbb{R}^d \\ f(x_0),\ldots,f(x_K),f(x^\star)\in\mathbb{R}}} \quad & f(x_K) - f(x^\star) \\
\text{subject to} \quad & \forall x_i, x_j \in I, \\
& f(x_i) \le f(x_j) + \langle \nabla f(x_j), x_i - x_j \rangle + \frac{L}{2}\|x_i - x_j\|^2, \\
& f(x_i) \ge f(x_j) + \langle \nabla f(x_j), x_i - x_j \rangle, \\
& x_{k+1} = x_k - \eta \nabla f(x_k), k = 0, \ldots, K-1, \\
& \|x_0 - x^\star\| \le R_0, \\
& x_0, \ldots, x_K, x^\star \in \mathbb{R}^d.
\end{aligned} \tag{P2}$$

Clearly, this simple discretization forms a necessary condition to "function f being L-smooth and convex". Thus, for the optimal values, we have (P1) $\le$ (P2). One would naturally ask whether this discretization is a sufficient condition. In other words, it is to ask if we have several vectors and scalars $\nabla f(x_0), \ldots, \nabla f(x_K) \in \mathbb{R}^d$, $f(x_0), \ldots, f(x_K), f(x^\star) \in \mathbb{R}$ satisfying the inequalities in (P2), does there exist an L-smooth and convex function f which interpolates these function values and gradients at $x_0, \ldots, x_K, x^\star$? The answer is, perhaps surprisingly, no. See Figure 1 in [145] for a counterexample. In the same work, Taylor et al. [145] discovered that the necessary and sufficient condition for the existence of an L-smooth convex f interpolating the set of triples $\{(x_i, \nabla f(x_i), f(x_i)) \mid x_i \in I\}$ is

$$f(x_i) - f(x_j) - \langle \nabla f(x_j), x_i - x_j \rangle \ge \frac{1}{2L}\|\nabla f(x_i) - \nabla f(x_j)\|^2, \forall x_i, x_j \in I. \tag{1.2}$$

We call it the interpolation condition throughout the thesis. Formulating the PEP with this condition, we obtain

$$\begin{aligned}
\max_{\substack{\nabla f(x_0),\ldots,\nabla f(x_K)\in\mathbb{R}^d\\ f(x_0),\ldots,f(x_K),f(x^\star)\in\mathbb{R}}} \quad & f(x_K)-f(x^\star)\\
\text{subject to} \quad & \forall x_i,x_j\in I,\\
& f(x_i)-f(x_j)-\langle\nabla f(x_j),x_i-x_j\rangle \geq \frac{1}{2L}\left\|\nabla f(x_i)-\nabla f(x_j)\right\|^2,\\
& x_{k+1}=x_k-\eta\nabla f(x_k), k=0,\ldots,K-1,\\
& \|x_0-x^\star\|\leq R_0,\\
& x_0,\ldots,x_K,x^\star\in\mathbb{R}^d.
\end{aligned} \tag{P3}$$

Thus, for the optimal values, it satisfies that (P1) = (P3).

After suitable reformulation, (P3) is an semidefinite programming (SDP) problem, so we have zero duality gap [14]. Given constants L, K and R_0, and fixing $\eta=\frac{1}{L}$, we can use an existing SDP solver such as MOSEK to solve (P3), which returns the exact worst-case performance of GD. A very nice property is that the primal solution of (P3) specifies the f on which GD performs poorly (i.e., the lower bound), and the dual solution of (P3) is the multipliers of the constraints, from which we can derive an upper bound $f(x_K)-f(x^\star)\leq\frac{LR_0^2}{4K+2}$ [37]. This upper bound is tighter than (1.1) from the previous work [12] by a factor of 2. In fact, since we have zero duality gap, this upper bound is the tightest possible one for GD, that is, there exists an f, on which GD converges with this upper bound holding as equality. The analytic expression of the "worst" function f can also be derived from the numerical solution of (P3) [37], demonstrating the significance of this PEP methodology.

An intriguing feature of this methodology is that the discovered worst-case analysis is usually pretty complicated, which seems impossible to find with bare hands. For example, the above tight analysis of GD requires the following potential function (a common analytic component in conducting worst-case analysis [10]):

$$\phi(x_k)=\frac{(4K+2)k}{2K-k}\big(f(x_k)-f(x^\star)\big)+\frac{L\,(4K+2)\,(K-k)}{(2K+1-k)(2K-k)}\left\|x_k-x^\star\right\|^2.$$

The coefficients are too complicated for human experts to discover barehanded.

1.5 Thesis Organization

Table 1.1: Summary of the theoretic setting in each chapter.

Problem	**Oracle**	**Assumptions**	**Chapter**
$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(x)$	Incremental gradient oracle $\nabla f_i(x)$ Incremental proximal point oracle [27]	Smoothness Strong convexity	Chapter 2
	Incremental gradient oracle $\nabla f_i(x)$	Smoothness Convexity	Chapter 3
	Incremental gradient oracle $\nabla f_i(x)$ with sparsity [98]	Smoothness Strong convexity	Chapter 4
$\min_{x\in X} f(x) + h(x)$	Stochastic approximation oracle [103] Proximal oracle $\mathrm{prox}_h(x, \mathcal{G})$ [46]	Smoothness Convexity Bounded variance	Chapter 5
Find approx. stationary point	Stochastic approximation oracle [103]	Lipschitz continuity Bounded variance	Chapter 6

This thesis is organized as follows (detailed theoretic settings are given in Table 1.1):

- Chapter 2 presents 4 theoretically faster algorithms for strongly convex finite-sum problems equipped with various oracles.[3] The derivation is inspired by the PEP methodology mentioned in the introduction. The proposed methods achieve the fastest known convergence rates in their corresponding cases.[4]

- Chapter 3 focuses on finding near-stationary points $\|\nabla f(x)\| \leq \epsilon$ of smooth convex finite-sum problems, which is motivated by several real world applications and facts. We derived 3 new schemes that are more practical than their existing counterparts based on novel usages of the PEP methodology.

[3]Oracle basically means the information available to any algorithm that solves the problem. Oracle complexity (or simply complexity) refers to the required number of oracle calls to find an ϵ-accurate solution.

[4]At the time [161] was published. A recent work [142] further extends the techniques in [161] and achieves the fastest possible rate in the deterministic strongly convex setting.

- Chapter 4 proposes the first asynchronous lock-free method with optimal gradient complexity. This new method is scalable as it fully utilizes the parallel computing architectures such as multi-core computer or distributed system, and is thus highly efficient for large-scale machine learning tasks. The derivation of this method is inspired by the development in Chapter 3.

- Chapter 5 aims to improve neural network optimization, and proposes a new robust momentum technique based on insights from stochastic convex analysis. Its efficacy is empirically verified by extensive deep learning experiments.

- Chapter 6 concerns the theoretic foundation of neural network optimization, in which we propose the first practical stochastic gradient method that is guaranteed to find an approximately stationary points of neural networks with finite-time and dimension-independent complexity.

- Chapter 7 concludes this thesis and discusses some future directions.

We defer all the proofs to the appendices to improve the readability.

Connections Between Different Chapters. As can be seen from the above, Chapters 2, 3 and 4 are connected through the PEP methodology, as they either get inspiration from the numerical solution of a PEP or provide new ideas for the PEP formulation. Moreover, Chapters 2, 3 and 4 all focus on finite-sum convex optimization. Chapters 5 and 6 both study neural network optimization, while Chapter 5 focuses on providing a better empirical neural network optimizer and Chapter 6 aims to build theoretically grounded neural network optimizers.

1.6 Publications Related to This Thesis

The results in this thesis are based on the following papers:

- Chapter 2 is based on the publication [161]:

  K. Zhou, A. M.-C. So, and J. Cheng. Boosting First-Order Methods by Shifting Objective: New Schemes with Faster Worst-Case Rates. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 15405-15416, 2020.

- Chapter 3 is based on the publication [163]:

  K. Zhou, L. Tian, A. M.-C. So, and J. Cheng. Practical Schemes for Finding Near-Stationary Points of Convex Finite-Sums. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3684-3708, 2022.

- Chapter 4 is based on the workshop paper [162]:

  K. Zhou, A. M.-C. So, and J. Cheng. Accelerating Perturbed Stochastic Iterates in Asynchronous Lock-Free Optimization. In *NeurIPS Workshop on Optimization for Machine Learning (NeurIPS OPT)*, 2022.

- Chapter 5 is based on the publication [160]:

  K. Zhou, Y. Jin, Q. Ding, and J. Cheng. Amortized Nesterov's Momentum: A Robust Momentum and Its Application to Deep Learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211-220, 2020.

- Chapter 6 is based on my contribution in the work [146]:

  L. Tian, **K. Zhou**, and A. M.-C. So. On the Finite-Time Complexity and Practical Computation of Approximate Stationarity Concepts of Lipschitz Functions. In *International Conference on Machine Learning (ICML)*, pages 21360-21379, 2022.

Chapter 2

Even Faster First-Order Methods for Strongly Convex Problems

In this chapter, we focus on the following smooth strongly convex finite-sum problem:

$$x^\star = \arg\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}, \tag{2.1}$$

where each f_i is L-smooth and μ-strongly convex.[1]

2.1 The Shifting Methodology

We tackle problem (2.1) from a new angle: Instead of designing methods to solve the original objective function f, we propose methods that are designed to solve a shifted objective h:

$$\begin{aligned}
&\min_{x \in \mathbb{R}^d} h(x) = \frac{1}{n} \sum_{i=1}^{n} h_i(x), \\
&\text{where } h_i(x) = f_i(x) - f_i(x^\star) - \langle \nabla f_i(x^\star), x - x^\star \rangle - \frac{\mu}{2} \left\| x - x^\star \right\|^2.
\end{aligned}$$

It can be easily verified that each $h_i(x)$ is $(L-\mu)$-smooth and convex, $\nabla h_i(x) = \nabla f_i(x) - \nabla f_i(x^\star) - \mu(x - x^\star)$, $\nabla h(x) = \nabla f(x) - \mu(x - x^\star)$, $h_i(x^\star) = h(x^\star) = 0$ and

[1] If each $f_i(\cdot)$ is L-smooth, the averaged function $f(\cdot)$ is itself L-smooth — but typically with a smaller L. We keep L as the smoothness constant for consistency.

$\nabla h_i(x^\star) = \nabla h(x^\star) = \mathbf{0}$, which means that the shifted problem and problem (2.1) share the same optimal solution $x^\star$. Let us write a well-known property of h:

$$\forall x, y \in \mathbb{R}^d, h(x) - h(y) - \langle \nabla h(y), x - y \rangle \geq \frac{1}{2(L-\mu)} \left\| \nabla h(x) - \nabla h(y) \right\|^2, \tag{2.2}$$

which encodes both the smoothness and strong convexity of f. The discrete version of this inequality is equivalent to the *smooth strongly convex interpolation condition* discovered in Taylor et al. [145]. Similar to the discussion in the introduction, this type of inequality forms a necessary and sufficient condition for the existence of a smooth strongly convex f interpolating a given set of triples $\{(x_i, \nabla f_i, f_i)\}$, while the usual collection of L-smoothness and strong convexity inequalities is only a necessary condition.[2] From the perspective of the PEP formulation [37], it implies that tighter results can be derived by exploiting condition (2.2) than using smoothness and strong convexity "separately", which is common in existing worst-case analysis. We show that our methodology effectively exploits this condition and consequently, we derive several methods that achieve faster worst-case convergence rates than their existing counterparts.

In summary, our methodology and proposed methods have the following distinctive features:

- We show that our methodology works for problems equipped with various first-order oracles: deterministic gradient oracle, incremental gradient oracle and incremental proximal point oracle.

- We leverage a cleaner version of the interpolation condition discovered in Taylor et al. [145], which leads to simpler and tighter analysis to the proposed methods than their existing counterparts.

- For our proposed stochastic methods, we deal with shifted variance bounds / shifted stochastic gradient norm bounds, which are different from all previous works.

- All the proposed methods achieve faster worst-case convergence rates than their counterparts (roughly by a factor of 2) that were designed to solve the original objective f.

[2]It implies that those inequalities may allow a non-smooth f interpolating the set, and thus a worst-case rate built upon those inequalities may not be achieved by any smooth f (i.e., the rate is loose). See Taylor et al. [145] for details.

Our development is motivated by the PEP methodology [37] and a recently proposed robust momentum method [24], which converges under a Lyapunov function (also known as the potential function [10]) that contains a term

$$h(x) - \frac{1}{2(L-\mu)} \left\| \nabla h(x) \right\|^2 .$$

This chapter conducts a comprehensive study of the special structure of this term.

This chapter is organized as follows: In Section 2.2, we present high-level ideas and lemmas that are the core building blocks of our methodology. In Section 2.3, we propose an accelerated method for the $n = 1$ case. In Section 2.4, we propose accelerated stochastic variance-reduced methods for the $n \geq 1$ case with incremental gradient oracle. In Section 2.5, we propose an accelerated method for the $n \geq 1$ case with incremental proximal point oracle. In Section 2.6, we provide experiments.

Oracle Definitions. Given a point $x \in \mathbb{R}^d$, an index $i \in [n]$ and $\alpha > 0$, we define various oracles as follows:

- *Deterministic oracle* returns $\big(f(x), \nabla f(x)\big)$.
- *Incremental first-order oracle* returns $\big(f_i(x), \nabla f_i(x)\big)$.
- *Incremental proximal point oracle* returns $\big(f_i(x), \nabla f_i(x), \mathrm{prox}_i^{\alpha}(x)\big)$, where the proximal operator is defined as

$$\mathrm{prox}_i^{\alpha}(z) = \arg\min_x \Big\{ f_i(x) + \frac{\alpha}{2} \left\| x - z \right\|^2 \Big\}.$$

We denote $\epsilon > 0$ as the required accuracy for solving problem (2.1) (i.e., to achieve $\left\| x - x^\star \right\|^2 \leq \epsilon$), which is assumed to be small.

2.2 Tackling the Shifted Objective

As mentioned in the beginning of this chapter, our methodology is to minimize the shifted objective[3] h with the aim of exploiting the interpolation condition. However, a critical issue is that we cannot even compute its gradient $\nabla h(x)$ (or $\nabla h_i(x)$), which requires the knowledge of $x^\star$. We figured out that in some simple cases, a change

[3]In the Lyapunov analysis framework, this is equivalent to picking a family of Lyapunov function that only involves the shifted objective h (instead of f). See Bansal and Gupta [10] for a nice review of Lyapunov-function-based proofs.

of "perspective" is enough to access this gradient information. Take GD $x_{k+1} = x_k - \eta \nabla f(x_k)$ as an example. Based on the definition $\nabla h(x_k) = \nabla f(x_k) - \mu(x_k - x^\star)$, we can rewrite the GD update as $x_{k+1} - x^\star = (1 - \eta\mu)(x_k - x^\star) - \eta \nabla h(x_k)$, and thus

$$\|x_{k+1} - x^\star\|^2 = (1-\eta\mu)^2 \|x_k - x^\star\|^2 \underbrace{-2\eta(1-\eta\mu)\langle \nabla h(x_k), x_k - x^\star\rangle + \eta^2 \|\nabla h(x_k)\|^2}_{R_0}.$$

If we set $\eta = \frac{2}{L+\mu}$, using the interpolation condition (2.2), we can conclude that $R_0 \le 0$, which leads to a convergence guarantee. It turns out that this argument is just the one-line proof of GD in the textbook (Theorem 2.1.15, [109]) but looks more structured in our opinion. However, this change of "perspective" is too abstract for more complicated schemes. Our solution is to first fix a template updating rule, and then encode this idea into a technical lemma, which serves as an instantiation of the shifted gradient oracle. To facilitate its usage, we formulate this lemma with a classic inequality whose usage has been well-studied. Proofs in this section are given in Appendix A.1.

Lemma 1 (Shifted mirror descent lemma)**.** *Given a gradient estimator $\mathcal{G}_y$, vectors $z^+, z^-, y \in \mathbb{R}^d$, fix the updating rule $z^+ = \arg\min_x \{ \langle \mathcal{G}_y, x\rangle + \frac{\alpha}{2}\|x - z^-\|^2 + \frac{\mu}{2}\|x - y\|^2 \}$. Suppose that we have a shifted gradient estimator $\mathcal{H}_y$ satisfying the relation $\mathcal{H}_y = \mathcal{G}_y - \mu(y - x^\star)$, it holds that*

$$\langle \mathcal{H}_y, z^- - x^\star\rangle = \frac{\alpha}{2}\left(\|z^- - x^\star\|^2 - \left(1 + \frac{\mu}{\alpha}\right)^2 \|z^+ - x^\star\|^2\right) + \frac{1}{2\alpha}\|\mathcal{H}_y\|^2.$$

Remark 1. *In general convex optimization, a similar lemma (for $\mathcal{G}$) serves as the core lemma for mirror descent*[4] *(e.g., Theorem 5.3.1 in the textbook [14]). This type of lemma also appears frequently in online optimization, which is used as an upper bound on the regret at the current iteration (e.g., Lemma 3 in Shalev-Shwartz and Singer [135]). In the strongly convex setting, unlike the common $(1 + \frac{\mu}{\alpha})^{-1}$ (or $1 - \frac{\mu}{\alpha}$) contraction ratio in existing work (e.g., Lemma 2.5 in Allen-Zhu [4]), Lemma 1 provides a $(1 + \frac{\mu}{\alpha})^{-2}$ ratio, which is one of the keys to the improved worst-case rates achieved in this chapter.*

Lemma 1 allows us to choose various gradient estimators for h directly, given that the relation $\mathcal{H}_x = \mathcal{G}_x - \mu(x - x^\star)$ holds for some practical $\mathcal{G}_x$. Here we provide some examples:

[4]In the Euclidean case, mirror descent coincides with GD. It represents a different approach to the same method.

- Deterministic gradient:

$$\mathcal{H}_x^{\mathrm{GD}} = \nabla h(x) \Rightarrow \mathcal{G}_x^{\mathrm{GD}} = \nabla f(x).$$

- SVRG estimator [64]:

$$\begin{aligned} \mathcal{H}_x^{\mathrm{SVRG}} &= \nabla h_i(x) - \nabla h_i(\tilde{x}) + \nabla h(\tilde{x}) \\ \Rightarrow \mathcal{G}_x^{\mathrm{SVRG}} &= \nabla f_i(x) - \nabla f_i(\tilde{x}) + \nabla f(\tilde{x}). \end{aligned}$$

- SAGA estimator [29]:

$$\begin{aligned} \mathcal{H}_x^{\mathrm{SAGA}} &= \nabla h_i(x) - \nabla h_i(\phi_i) + \frac{1}{n}\sum_{j=1}^{n} \nabla h_j(\phi_j) \\ \Rightarrow \mathcal{G}_x^{\mathrm{SAGA}} &= \nabla f_i(x) - \nabla f_i(\phi_i) + \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(\phi_j) - \mu\big(\frac{1}{n}\sum_{j=1}^{n} \phi_j - \phi_i\big). \end{aligned}$$

- SARAH estimator [111]:

$$\begin{aligned} \mathcal{H}_{x_k}^{\mathrm{SARAH}} &= \nabla h_{i_k}(x_k) - \nabla h_{i_k}(x_{k-1}) + \mathcal{H}_{x_{k-1}}^{\mathrm{SARAH}} \text{ and } \mathcal{H}_{x_0}^{\mathrm{SARAH}} = \nabla h(x_0) \\ \Rightarrow \mathcal{G}_{x_k}^{\mathrm{SARAH}} &= \nabla f_{i_k}(x_k) - \nabla f_{i_k}(x_{k-1}) + \mathcal{G}_{x_{k-1}}^{\mathrm{SARAH}} \text{ and } \mathcal{G}_{x_0}^{\mathrm{SARAH}} = \nabla f(x_0). \end{aligned}$$

It can be verified that the relation $\mathcal{H}_x = \mathcal{G}_x - \mu(x - x^\star)$ holds in all these examples. Note that it is important to ensure that $\mathcal{G}_x$ is practical. For example, the shifted stochastic gradient estimator $\nabla h_i(x) = [\nabla f_i(x) - \nabla f_i(x^\star)] - \mu(x - x^\star)$ does not induce a practical $\mathcal{G}_x$.

We also apply the idea of changing "perspective" to proximal operator prox_i^α as given below.

Lemma 2 (Shifted firm non-expansiveness)**.** *Given relations $z^+ = \mathrm{prox}_i^\alpha(z^-)$ and $y^+ = \mathrm{prox}_i^\alpha(y^-)$, it holds that*

$$\frac{1}{\alpha^2}\left(1 + \frac{2(\alpha+\mu)}{L-\mu}\right)\left\|\nabla h_i(z^+) - \nabla h_i(y^+)\right\|^2 + \left(1 + \frac{\mu}{\alpha}\right)^2\left\|z^+ - y^+\right\|^2 \le \left\|z^- - y^-\right\|^2.$$

Remark 2. *Recall the definition of a firmly non-expansive operator T (e.g., Definition 4.1 in the textbook [11]): $\forall x, y,\ \|Tx - Ty\|^2 + \|(\mathrm{Id} - T)x - (\mathrm{Id} - T)y\|^2 \le$*

$\|x-y\|^2$. *Lemma 2 can be derived by choosing*[5] $T = (1+\frac{\mu}{\alpha}) \cdot \mathrm{prox}_i^{\alpha}$ *and strengthening* $\langle Tx - Ty, (\mathrm{Id} - T)x - (\mathrm{Id} - T)y\rangle \geq 0$ *using the interpolation condition. A similar lemma has also been used in the analysis of the proximal point algorithm [127]. In the problem setting of this chapter, Defazio [27] also strengthened firm non-expansiveness, which produces a* $(1+\frac{\mu}{\alpha})^{-1}$ *contraction ratio instead of the above* $(1+\frac{\mu}{\alpha})^{-2}$ *ratio created by shifting objective.*

Now we have all the building blocks to migrate existing schemes to tackle the shifted objective. To maximize the potential of our methodology, we focus on developing accelerated methods. We can also tighten the analysis of non-accelerated methods, which could lead to new algorithmic schemes.

2.3 Deterministic Objectives

We consider the objective function (2.1) with $n=1$. To begin, we recap the guarantee of NAG to facilitate the comparison. The proof is given in Appendix A.6 for completeness. At iteration $K-1$, NAG produces

$$f(x_K) - f(x^\star) + \frac{\mu}{2}\|z_K - x^\star\|^2 \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right)^K \left(f(x_0) - f(x^\star) + \frac{\mu}{2}\|z_0 - x^\star\|^2\right),$$

where $x_0, z_0 \in \mathbb{R}^d$ are the initial guesses. Denote the initial constant as $C_0^{\mathrm{NAG}} \triangleq f(x_0) - f(x^\star) + \frac{\mu}{2}\|z_0 - x^\star\|^2$. This guarantee shows that in terms of reducing $\|x - x^\star\|^2$ to ϵ, the sequences $\{x_k\}$ (due to $f(x_K) - f(x^\star) \geq \frac{\mu}{2}\|x_K - x^\star\|^2$) and $\{z_k\}$ have the same iteration complexity $\sqrt{\kappa}\log\frac{2C_0^{\mathrm{NAG}}}{\mu\epsilon}$.

2.3.1 Generalized Triple Momentum Method

We present the first application of our methodology in Algorithm 1, which can be regarded as a technical migration[6] of NAG to the shifted objective. It turns out that Algorithm 1, when tuned optimally, is equivalent to TM [134] (except for the first iteration). We thus name it as Generalized Triple Momentum method (G-TM). In comparison with TM, G-TM has the following advantages:

[5]In the strongly convex setting, $(1+\frac{\mu}{\alpha}) \cdot \mathrm{prox}_i^{\alpha}$ is firmly non-expansive (e.g., Proposition 1 in Defazio [27]).

[6]In our opinion, the most important techniques in NAG are Lemma 3 for f and the mirror descent lemma. Algorithm 1 was derived by having a shifted version of Lemma 3 for h and the shifted mirror descent lemma.

Algorithm 1 Generalized Triple Momentum (G-TM)

```
Input: $\{\alpha_k > 0\}, \{\tau_k^x \in ]0,1[\}, \{\tau_k^z > 0\}$, initial guesses $y_{-1}, z_0 \in \mathbb{R}^d$ and iteration number $K$.
1: for $k = 0, \ldots, K-1$ do
2:     $y_k = \tau_k^x z_k + (1-\tau_k^x) y_{k-1} + \tau_k^z \big(\mu(y_{k-1} - z_k) - \nabla f(y_{k-1})\big)$.
3:     $z_{k+1} = \arg\min_x \Big\{ \langle \nabla f(y_k), x\rangle + (\alpha_k/2)\|x - z_k\|^2 + (\mu/2)\|x - y_k\|^2 \Big\}$.
4: end for
Output: $z_K$.
```

- *Refined convergence guarantee.* TM has the guarantee (Eq.(11) in Cyrus et al. [24] with $\rho = 1 - \frac{1}{\sqrt{\kappa}}$):

$$\begin{aligned}&\|z_K - x^\star\|^2 \\ &\le \left(1 - \frac{1}{\sqrt{\kappa}}\right)^{2(K-1)} \left(\|z_1 - x^\star\|^2 + \frac{L-\mu}{L\mu}\left(h(y_0) - \frac{1}{2(L-\mu)}\|\nabla h(y_0)\|^2\right)\right),\end{aligned}$$

  which has an initial state issue: its initial constant correlates with z_1, which is not an initial guess. It can be verified that the first iteration of TM is GD with a $\frac{1}{\sqrt{L\mu}}$ stepsize, which exceeds the $\frac{2}{L+\mu}$ limit, and thus we do not have $\|z_1 - x^\star\|^2 \le \|z_0 - x^\star\|^2$ in general. This issue is possibly the reason for the $\log \sqrt{\kappa}$ factor stated in Scoy et al. [134]. G-TM resolves this issue and removes the log factor.

- *More extensible proof.* Our proof of G-TM is based on Lemma 1, which, as mentioned in Section 2.2, allows shifted stochastic gradients. In comparison, the analysis of TM starts with establishing an algebraic identity and it is unknown whether this identity holds in the stochastic case.

- *General scheme.* The framework of G-TM covers both NAG and TM. See Appendix A.2.1 for detailed derivations. When $\mu = 0$, it also covers the optimized gradient method [68], which is discussed in Section 2.7.

A subtlety of Algorithm 1 is that it requires storing a past gradient vector, and thus at the first iteration, two gradient computations are needed. The analysis of G-TM is based on the same Lyapunov function in Cyrus et al. [24]:

$$T_k = h(y_{k-1}) - \frac{1}{2(L-\mu)}\|\nabla h(y_{k-1})\|^2 + \frac{\lambda}{2}\|z_k - x^\star\|^2, \text{ where } \lambda > 0.$$

In the following theorem, we establish the per-iteration contraction of G-TM and the proof is given in Appendix A.2.2.

Theorem 1. *In Algorithm 1, if we fix $\tau_k^z = \frac{1-\tau_k^x}{L-\mu}, \forall k$ and choose $\{\alpha_k\}, \{\tau_k^x\}$ under the constraints*

$$2\alpha_k \geq L\tau_k^x - \mu \text{ and } \left(1 + \frac{\mu}{\alpha_k}\right)^2 (1 - \tau_k^x) \leq 1,$$

the iterations satisfy the contraction $T_{k+1} \leq (1 + \frac{\mu}{\alpha_k})^{-2} T_k$ with $\lambda = \frac{(\tau_k^x - \mu\tau_k^z)(\alpha_k+\mu)^2}{\alpha_k}$.

When the constraints hold as equality, we derive a simple constant choice for G-TM: $\alpha = \sqrt{L\mu} - \mu, \tau_x = \frac{2\sqrt{\kappa}-1}{\kappa}, \tau_z = \frac{\sqrt{\kappa}-1}{L(\sqrt{\kappa}+1)}$. Here we also provide the parameter choices of NAG and TM under the framework of G-TM for comparison. Detailed derivation is given in Appendix A.2.1.

$$\text{NAG}\begin{cases} \alpha = \sqrt{L\mu} - \mu; \\ \tau_k^x = (\sqrt{\kappa}+1)^{-1}, \tau_k^z = 0, & k = 0; \\ \tau_k^x = (\sqrt{\kappa})^{-1}, \tau_k^z = \frac{1}{L+\sqrt{L\mu}}, & k \geq 1. \end{cases} \quad \text{TM}\begin{cases} \alpha = \sqrt{L\mu} - \mu; \\ \tau_k^x = (\sqrt{\kappa}+1)^{-1}, \tau_k^z = 0, & k = 0; \\ \tau_k^x = \frac{2\sqrt{\kappa}-1}{\kappa}, \tau_k^z = \frac{\sqrt{\kappa}-1}{L(\sqrt{\kappa}+1)}, & k \geq 1. \end{cases}$$

Using the constant choice in Theorem 1, by telescoping the contraction from iteration $K-1$ to 0, we obtain

$$\begin{aligned} &\frac{\mu}{2}\|z_K - x^\star\|^2 \\ &\leq \left(1-\frac{1}{\sqrt{\kappa}}\right)^{2K}\left(\frac{\kappa-1}{2\kappa}\left(h(y_{-1})-\frac{1}{2(L-\mu)}\|\nabla h(y_{-1})\|^2\right)+\frac{\mu}{2}\|z_0 - x^\star\|^2\right). \end{aligned} \tag{2.3}$$

Denoting the initial constant as

$$C_0^{\text{G-TM}} \triangleq \frac{\kappa-1}{2\kappa}(h(y_{-1}) - \frac{1}{2(L-\mu)}\|\nabla h(y_{-1})\|^2) + \frac{\mu}{2}\|z_0 - x^\star\|^2,$$

if we align the initial guesses $y_{-1} = x_0$ with NAG, we have $C_0^{\text{G-TM}} \ll C_0^{\text{NAG}}$. This guarantee yields a $\frac{\sqrt{\kappa}}{2}\log\frac{2C_0^{\text{G-TM}}}{\mu\epsilon}$ iteration complexity for G-TM, which is at least two times lower than that of NAG and does not suffer from an additional $\log\sqrt{\kappa}$ factor as is the case for the original TM.

The Tightness of (2.3)

It is natural to ask how tight the worst-case guarantee (2.3) is. We show that for the quadratic[7] $f(x) = \frac{1}{2}\langle D^\kappa x, x\rangle$ where $D^\kappa \triangleq \text{diag}(L, \mu)$ is a diagonal matrix, G-TM converges exactly at the rate in (2.3). Note that for this objective, $h(x) -$

[7]This is also the example where GD with $\frac{2}{L+\mu}$ stepsize behaves exactly like its worst-case analysis.

$\frac{1}{2(L-\mu)} \|\nabla h(x)\|^2 \equiv 0$, which means that the guarantee becomes

$$\|z_K - x^\star\|^2 \le \left(1 - \frac{1}{\sqrt{\kappa}}\right)^{2K} \|z_0 - x^\star\|^2 .$$

Expanding the recursions in Algorithm 1, we obtain the following result and its proof is given in Appendix A.2.3.

Proposition 1.1. *If $f(x) = \frac{1}{2}\langle D^\kappa x, x\rangle$, G-TM produces*

$$\|z_K - x^\star\|^2 = \left(1 - \frac{1}{\sqrt{\kappa}}\right)^{2K} \|z_0 - x^\star\|^2 .$$

2.4 Finite-Sum Objectives with Incremental First-Order Oracle

We now consider the finite-sum objective (2.1) with $n \ge 1$. We choose SVRG [64] as the base algorithm to implement our boosting technique, and we also show that an accelerated SAGA [29] variant can be similarly constructed in Section 2.4.2. Proofs in this section are given in Appendix A.3.

2.4.1 BS-SVRG

As mentioned in Section 2.2, the shifted SVRG estimator $\mathcal{H}_x^{\text{SVRG}}$ induces a practical $\mathcal{G}_x^{\text{SVRG}}$ (which is just the original SVRG estimator [64]) and thus by using Lemma 1, we obtain a practical updating rule and a classic equality for the shifted estimator. Now we can design an accelerated SVRG variant that minimizes h. To make the notations specific, we define $\mathcal{G}_{x_k}^{\text{SVRG}} \triangleq \nabla f_{i_k}(x_k) - \nabla f_{i_k}(\tilde{x}_s) + \nabla f(\tilde{x}_s)$, where i_k is sampled uniformly in $[n]$ and $\tilde{x}_s$ is a previously chosen random anchor point. For simplicity, in what follows, we only consider constant parameter choices. We name our SVRG variant as BS-SVRG (Algorithm 2), which is designed based on the following thought experiment.

Thought experiment. We design BS-SVRG by extending G-TM, which is natural since almost all the existing stochastic accelerated methods are constructed based on NAG. For SVRG, its (directly) accelerated variants [4, 158, 81] all incorporate the idea of "negative" momentum, which is basically Nesterov's momentum provided by the anchor point $\tilde{x}_s$ instead of the previous iterate. Inspired by their success, we

Algorithm 2 SVRG Boosted by Shifting objective (BS-SVRG)

```
Input: Parameters $\alpha > 0, \tau_x \in ]0,1[$, initial guess $x_0 \in \mathbb{R}^d$, epoch number $S$ and
    epoch length $m$.
Initialize: Vectors $z_0^0 = \tilde{x}_0 = x_0$, constants $\tau_z = \frac{\tau_x}{\mu} - \frac{\alpha(1-\tau_x)}{\mu(L-\mu)}, \widetilde{\omega} = \sum_{k=0}^{m-1}\left(1+\frac{\mu}{\alpha}\right)^{2k}$.
1: for $s = 0, \ldots, S-1$ do
2:     Compute and store $\nabla f(\tilde{x}_s)$.
3:     for $k = 0, \ldots, m-1$ do
4:         $y_k^s = \tau_x z_k^s + (1-\tau_x)\,\tilde{x}_s + \tau_z\left(\mu(\tilde{x}_s - z_k^s) - \nabla f(\tilde{x}_s)\right)$.
5:         $z_{k+1}^s = \arg\min_x \left\{\left\langle \mathcal{G}_{y_k^s}^{\text{SVRG}}, x\right\rangle + (\alpha/2)\left\|x - z_k^s\right\|^2 + (\mu/2)\left\|x - y_k^s\right\|^2\right\}$.
6:     end for
7:     $\tilde{x}_{s+1}$ is sampled from $\left\{P(\tilde{x}_{s+1} = y_k^s) = \frac{1}{\widetilde{\omega}}\left(1+\frac{\mu}{\alpha}\right)^{2k} \,\middle|\, k \in \{0,\ldots,m-1\}\right\}$.
8:     $z_0^{s+1} = z_m^s$.
9: end for
Output: $z_0^S$.
```

design the "momentum step" of BS-SVRG (Step 4) by replacing all the previous iterate y_{k-1} in $y_k = \tau_x z_k + (1-\tau_x)y_{k-1} + \tau_z\big(\mu(y_{k-1} - z_k) - \nabla f(y_{k-1})\big)$ with the anchor point $\tilde{x}_s$. The insight is that the "momentum step" is aggressive and could be erroneous in the stochastic case. Thus, we construct it based on some "stable" point instead of the previous stochastic iterate.

We adopt a similar Lyapunov function as G-TM:

$$T_s \triangleq h(\tilde{x}_s) - c_1\left\|\nabla h(\tilde{x}_s)\right\|^2 + \frac{\lambda}{2}\left\|z_0^s - x^\star\right\|^2, \text{ where } c_1 \in \left[0, \frac{1}{2(L-\mu)}\right] \text{ and } \lambda > 0,$$

and build the per-epoch contraction of BS-SVRG as follows.

Theorem 2. *In Algorithm 2, if we choose α, τ_x under the constraints*

$$\left(1+\frac{\mu}{\alpha}\right)^{2m}(1-\tau_x) \leq 1 \textit{ and } (1+\tau_x)^2(1-\tau_x) \geq 4\left(\left(\frac{\alpha}{\mu}+1\right) - \left(\frac{\alpha}{\mu}+\kappa\right)\tau_x\right)^2,$$

the per-epoch contraction $\mathbb{E}\big[T_{s+1}\big] \leq (1+\frac{\mu}{\alpha})^{-2m}T_s$ holds with $\lambda = \frac{\alpha^2(1-\tau_x)}{\widetilde{\omega}(L-\mu)}(1+\frac{\mu}{\alpha})^{2m}$. The expectation is taken with respect to the information up to epoch s.

In what follows, we provide a simple analytic choice that satisfies the constraints. We consider the ill-conditioned case where $\frac{m}{\kappa} \leq \frac{3}{4}$, and we fix $m = 2n$ to make it specific.[8] In this case, Allen-Zhu [4] derived an $O(\sqrt{6n\kappa}\log\frac{1}{\epsilon})$ expected iteration complexity[9] for Katyusha (cf. Theorem 2.1, [4]).

Proposition 2.1 (Ill condition). *If $\frac{m}{\kappa} \leq \frac{3}{4}$, the choice $\alpha = \sqrt{cm\mu L} - \mu, \tau_x = (1 - \frac{1}{c\kappa})\frac{\sqrt{cm\kappa}}{\sqrt{cm\kappa}+\kappa-1}$, where $c = 2 + \sqrt{3}$, satisfies the constraints in Theorem 2.*

Using this parameter choice in Theorem 2, we obtain an $O(\sqrt{1.87n\kappa}\log\frac{1}{\epsilon})$ expected iteration complexity for BS-SVRG, which is around 1.8 times lower than that of Katyusha.

Remark 2.1. *We are not aware of other parameter choices of Katyusha that have faster rates. Hu et al. [58] made an attempt based on dissipativity theory, but no explicit rate is given. To derive a better choice for Katyusha, significant modification to its proof is required (for its parameter τ_2), which results in complicated constraints and is thus out of the scope of this chapter. We believe that there could be some computer-aided ways to find better choices for both Katyusha and BS-SVRG, which we leave for future work.*

For the other case where $\frac{m}{\kappa} > \frac{3}{4}$ (i.e., $\kappa = O(n)$), almost all the accelerated and non-accelerated incremental gradient methods perform the same, at an $O(n\log\frac{1}{\epsilon})$ oracle complexity (and is indeed fast). Hannah et al. [52] shows that by optimizing the parameters of SVRG and SARAH, a lower $O(n + \frac{n}{1+\max\{\log(n/\kappa),0\}}\log\frac{1}{\epsilon})$ oracle complexity is achievable. Due to these facts, we do not optimize the parameters for this case and provide the following proposition as a basic guarantee.

Proposition 2.2 (Well condition). *If $\frac{m}{\kappa} > \frac{3}{4}$, by choosing $\alpha = \frac{3L}{2} - \mu, \tau_x = (1 - \frac{1}{6m})\frac{3\kappa}{5\kappa-2}$, the epochs of BS-SVRG satisfy $T_{s+1} \leq \frac{1}{2}\cdot T_s$ with $\lambda = \frac{2\alpha^2(1-\tau_x)}{\widetilde{\omega}(L-\mu)}$, which implies an $O(n\log\frac{1}{\epsilon})$ expected iteration complexity.*

There exists a special choice in the constraints: by choosing $\tau_x = \frac{\alpha+\mu}{\alpha+L}$, the second constraint always holds and this leads to $c_1 = 0$ in T_s. In this case, α can be found using numerical tools, which is summarized as follows.

Proposition 2.3 (Numerical choice). *By fixing $\tau_x = \frac{\alpha+\mu}{\alpha+L}$, the optimal choice of α can be found by solving the equation $\left(1 + \frac{\mu}{\alpha}\right)^{2m}\left(1 - \frac{\alpha+\mu}{\alpha+L}\right) = 1$ using numerical tools, and this equation has a unique positive root.*

[8]We choose the setting that is used in the analysis and experiments of Katyusha [4] to make a fair comparison.

[9]We are referring to the expected number of stochastic iterations (e.g., in total Sm in Algorithm 2) required to achieve $\|x - x^\star\|^2 \leq \epsilon$. If $m = 2n$, in average, each stochastic iteration of SVRG requires 1.5 oracle calls.

Compared with Katyusha, BS-SVRG has a simpler scheme, which only requires storing one variable vector $\{z_k\}$ and tuning 2 parameters similar to MiG [158]. Moreover, BS-SVRG achieves the fastest rate among the accelerated SVRG variants.

2.4.2 Accelerated SAGA Variant

As given in Section 2.2, the shifted SAGA estimator $\mathcal{H}_x^{\text{SAGA}}$ also induces a practical gradient estimator, and thus we can design an accelerated SAGA variant in a similar way. Inspired by the existing (directly) accelerated SAGA variant [159], we can design the recursion (updating rule of the table) as $\phi_{i_k}^{k+1} = \tau_x z_k + (1-\tau_x)\,\phi_{i_k}^k + \tau_z\big(\mu(\frac{1}{n}\sum_{i=1}^n \phi_i^k - z_k) - \frac{1}{n}\sum_{i=1}^n \nabla f_i(\phi_i^k)\big)$. We found that for the resulting scheme, we can adopt the following Lyapunov function:

$$T_k = \frac{1}{n}\sum_{i=1}^n h_i(\phi_i^k) - c_1\left\|\frac{1}{n}\sum_{i=1}^n \nabla h_i(\phi_i^k)\right\|^2 + \frac{\lambda}{2}\left\|z_k - x^\star\right\|^2,$$
$$\text{where } c_1 \in \left[0, \frac{1}{2(L-\mu)}\right], \lambda > 0,$$

which is an "incremental version" of T_s. Note that

$$\frac{1}{n}\sum_{i=1}^n h_i(\phi_i^k) - c_1\left\|\frac{1}{n}\sum_{i=1}^n \nabla h_i(\phi_i^k)\right\|^2 \geq \frac{1}{n}\sum_{i=1}^n \big(h_i(\phi_i^k) - c_1\left\|\nabla h_i(\phi_i^k)\right\|^2\big) \geq 0.$$

A similar accelerated rate can be derived for the SAGA variant and its parameter choice shows some interesting correspondence between the variants of SVRG and SAGA. Moreover, the resulting scheme does not need the tricky "doubling sampling" in Zhou et al. [159] and thus it has a lower iteration complexity. However, since its updating rules require the knowledge of point table, the scheme has an undesirable $O(nd)$ memory complexity. We provide this variant in Appendix A.3.4 for interested readers.

2.5 Finite-Sum Objectives with Incremental Proximal Point Oracle

We consider the finite-sum objective (2.1) and assume that the proximal operator oracle $\text{prox}_i^\alpha(\cdot)$ of each f_i is available. Point-SAGA [27] is a typical method that utilizes this oracle, and it achieves the same $O\big((n+\sqrt{n\kappa})\log\frac{1}{\epsilon}\big)$ expected iteration

Algorithm 3 Point-SAGA Boosted by Shifting objective (BS-Point-SAGA)

```
Input: Parameters $\alpha > 0$ and initial guess $x_0 \in \mathbb{R}^d$, iteration number $K$.
Initialize: A point table $\phi^0 \in \mathbb{R}^{d \times n}$ with $\forall i \in [n], \phi_i^0 = x_0$, running averages for the point table and its gradients.
1: for $k = 0, \ldots, K-1$ do
2:     Sample $i_k$ uniformly in $[n]$.
3:     Update $x$: $z_k = x_k + \frac{1}{\alpha}\left(\nabla f_{i_k}(\phi_{i_k}^k) - \frac{1}{n}\sum_{i=1}^n \nabla f_i(\phi_i^k) + \mu\left(\frac{1}{n}\sum_{i=1}^n \phi_i^k - \phi_{i_k}^k\right)\right)$,
4:                 $x_{k+1} = \mathrm{prox}_{i_k}^{\alpha}(z_k)$.
5:     Set $\phi_{i_k}^{k+1} = x_{k+1}$ and keep other entries unchanged (i.e., for $i \neq i_k, \phi_i^{k+1} = \phi_i^k$). Update the running averages according to the change in $\phi^{k+1}$ (note that $\nabla f_{i_k}(\phi_{i_k}^{k+1}) = \alpha(z_k - x_{k+1})$).
6: end for
Output: $x_K$.
```

complexity. Although in general, the incremental proximal operator oracle is much more expensive than the incremental gradient oracle, Point-SAGA is interesting in the following aspects: (1) it has a simple scheme with only 1 parameter; (2) its analysis is elegant and tight, which does not require any Young's inequality; (3) for problems where the proximal point oracle has an analytic solution, it has a very fast rate (i.e., its rate factor is smaller than $1 - (n + \sqrt{n\kappa} + 1)^{-1}$, which is faster than both Katyusha and BS-SVRG).

It might be surprising that by shifting objective, the convergence rate of Point-SAGA can be further boosted. We name the proposed variant as BS-Point-SAGA, which is presented in Algorithm 3. Recall that the Lyapunov function used to analyze Point-SAGA has the form (cf. Theorem 5, [27]):

$$T_k^{\text{Point-SAGA}} = \frac{c}{n}\sum_{i=1}^n \left\|\nabla f_i(\phi_i^k) - \nabla f_i(x^\star)\right\|^2 + \|x_k - x^\star\|^2.$$

We adopt a shifted version of this Lyapunov function:

$$T_k = \lambda \cdot \frac{1}{n}\sum_{i=1}^n \left\|\nabla h_i(\phi_i^k)\right\|^2 + \|x_k - x^\star\|^2, \text{ where } \lambda > 0.$$

The analysis of BS-Point-SAGA is a direct application of Lemma 2. We build the per-iteration contraction in the following theorem, and its proof is given in Appendix A.4.

Theorem 3. *In Algorithm 3, if we choose α as the unique positive root of the cubic equation*

$$2\left(\frac{\alpha}{\mu}\right)^3 - (4n-6)\left(\frac{\alpha}{\mu}\right)^2 - (2n\kappa + 4n - 6)\left(\frac{\alpha}{\mu}\right) - (n\kappa + n - 2) = 0,$$

the per-iteration contraction $\mathbb{E}_{i_k}\left[T_{k+1}\right] \leq (1+\frac{\mu}{\alpha})^{-2}T_k$ holds with $\lambda = \frac{n}{\alpha^2} + \frac{2(\alpha+\mu)(n-1)}{\alpha^2(L-\mu)}$. The root of this cubic equation satisfies $\frac{\alpha}{\mu} = O(n+\sqrt{n\kappa})$, which implies an $O\big((n+\sqrt{n\kappa})\log\frac{1}{\epsilon}\big)$ expected iteration complexity.

The expected worst-case rate factor of BS-Point-SAGA is minimized by solving the cubic equation in Theorem 3 exactly. The analytic solution of this equation is messy, but it can be easily calculated using numerical tools. In Figure 2.1, we numerically compare the rate factors of Point-SAGA and BS-Point-SAGA. When κ is large, the rate factor of BS-Point-SAGA is close to the square of the rate factor of Point-SAGA, which implies an almost 2 times lower expected iteration complexity. In terms of memory requirement, BS-Point-SAGA has an undesirable $O(nd)$ complexity since the update of x_{k+1} involves $\phi_{i_k}^k$. Nevertheless, it achieves the fastest known rate for finite-sum problems (if both L and μ are known), and we present it as a special instance of our design methodology.

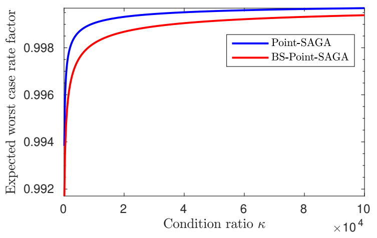
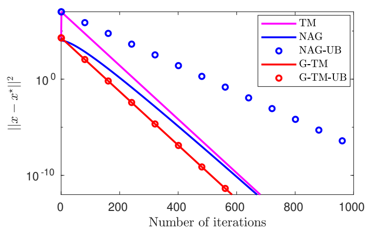


Figure 2.1: A comparison of the expected worst-case rate factors.

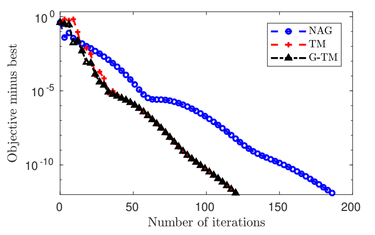
2.6 Performance Evaluations

In general, a faster worst-case rate does not necessarily imply a better empirical performance. It is possible that the slower rate is loose or the worst-case analysis is not representative of reality (e.g., worst-case scenarios are not stable to perturbations). We provide experimental results of the proposed methods in this section. We evaluate them in the ill-conditioned case where the problem has a huge κ to justify the accelerated $\sqrt{\kappa}$ dependence. Detailed experimental setup can be found in Appendix A.5.

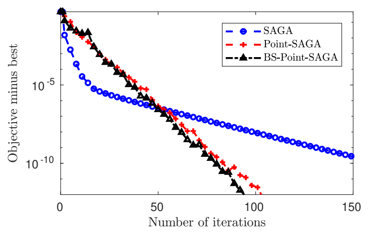
We started with evaluating the deterministic methods: NAG, TM and G-TM. We first did a simulation on the quadratic objective mentioned in Section 2.3.1, which also serves as a justification of Proposition 1.1. In this simulation, the default (constant) parameter choices were used and all the methods were initialized



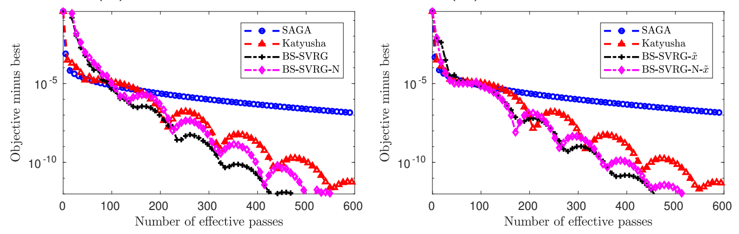
(a) Simulation.



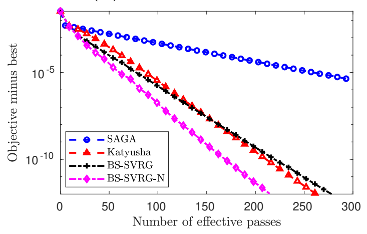
(b) ijcnn1 dataset.



(c) w8a dataset.



(d) a9a dataset. BS-SVRG outputs z (Left) or $\tilde{x}$ (Right).



(e) covtype dataset.

Figure 2.2: Evaluations. (a) Quadratic, $L = 1, \mu = 10^{-3}$. (b) ℓ_2-logistic regression, $\mu = 10^{-3}$. (c) Ridge regression, $\mu = 5 \times 10^{-7}$. (d) (e) ℓ_2-logistic regression, $\mu = 10^{-8}$.

in $(-100, 100)$. We plot their convergences and theoretical guarantees (marked with "UB") in Figure 2.2a (the bound for TM is not shown due to the initial state issue). This simulation shows that after the first iteration, TM and G-TM have the same rate, and the initial state issue of TM can make it slower than NAG. It also suggests that the guarantee of NAG is loose.

Then, we measured their performance on real world datasets from LIBSVM [20]. The task we chose is ℓ_2-logistic regression. We normalized the datasets and thus for this problem, $L = 0.25 + \mu$. For real world tasks, we tracked function value suboptimality, which is easier to compute than $\|x - x^\star\|^2$ in practice. The result is given in Figure 2.2b. In the first 30 iterations, TM is slower than G-TM due to the initial state issue. After that, they are almost identical and are faster than NAG.

We then evaluated BS-SVRG on the same problem, which can fully utilize the finite-sum structure. We evaluated two parameter choices of BS-SVRG: (1) the analytic choice in Proposition 2.1 (marked as "BS-SVRG"); (2) the numerical choice in Proposition 2.3 (marked as "BS-SVRG-N"). We selected SAGA ($\gamma = \frac{1}{2(\mu n + L)}$, [29]) and Katyusha ($\tau_2 = \frac{1}{2}, \tau_1 = \sqrt{\frac{m}{3\kappa}}, \alpha = \frac{1}{3\tau_1 L}$, [4]) with their default parameter choices as the baselines. Since SAGA and SVRG-like algorithms have different iteration complexities, we plot the curve with respect to the number of data passes. The results are given in Figure 2.2d and 2.2e. In the experiment on a9a dataset (Figure 2.2d (Left)), both choices of BS-SVRG perform well after 100 passes. The issue of their early stage performance can be eased by outputting the anchor point $\tilde{x}$ instead, as shown in Figure 2.2d (Right).

We also conducted an empirical comparison between BS-Point-SAGA and Point-SAGA in Figure 2.2c. Their analytic parameter choices were used. We chose ridge regression as the task since its proximal operator has a closed form solution (see Appendix A in Defazio [27]). For this objective, after normalizing the dataset, $L = 1 + \mu$. The performance of SAGA is also plotted as a reference.

2.7 Chapter Summary and Discussion

In this chapter, we focused on unconstrained smooth strongly convex problems and designed new schemes for a shifted objective. Lemma 1 and Lemma 2 are the cornerstones for the new designs, which serve as instantiations of the shifted gradient oracle. Following this methodology, we proposed G-TM, BS-SVRG (and BS-SAGA) and BS-Point-SAGA. The new schemes achieve faster worst-case rates and have tighter and simpler proofs compared with their existing counterparts. Experiments on machine learning tasks show some improvement of the proposed methods.

Although provided only for strongly convex problems, our framework of exploiting the interpolation condition (i.e., Algorithm 1) can also be extended to the non-strongly convex case ($\mu = 0$). It can be easily verified that Theorem 1 holds with $\mu = 0$ and thus we can choose a variable-parameter setting that leads to the $O(1/K^2)$ rate. It turns out that Algorithm 1 in this case is equivalent to the optimized gradient method [68], which is also covered by the second accelerated method (14) studied in Taylor and Bach [141]. Moreover, the Lyapunov function T_k becomes $a_k\big(f(y_{k-1}) - f(x^\star) - \frac{1}{2L}\|\nabla f(y_{k-1})\|^2\big) + \frac{L}{4}\|z_k - x^\star\|^2$ for some $a_k > 0$, which is exactly the one used in Theorem 11, [141].

While the proposed approach boosts the convergence rate, some limitations should be stressed. First, it requires a prior knowledge of the strong convexity constant μ since even if it is applied to a non-accelerated method, the parameter choice is always related to μ. Furthermore, this methodology relies heavily on the interpolation condition, which requires f to be defined everywhere on $\mathbb{R}^d$ [35]. This restriction makes it hardly generalizable to the constrained/proximal setting [108] (for the proximal case, a possible solution is to assume that the smooth part is defined everywhere on $\mathbb{R}^d$ [12, 70, 144]).

Chapter 3

Practical First-Order Methods for Finding Near-Stationary Points

In this chapter, we focus on finding the near-stationary points $\|\nabla f(x)\| \le \epsilon$ of the following smooth convex finite-sum problem:

$$\min_{x\in\mathbb{R}^d}\left\{f(x)=\frac{1}{n}\sum_{i=1}^{n} f_i(x)\right\},$$

where each f_i is L-smooth and convex. We start with clarifying the motivation of finding near-stationary points in convex optimization.

3.1 Motivations

Classic convex optimization usually focuses on providing guarantees for minimizing function value. For this task, the optimal (up to constant factors) NAG [104, 107] has been known for decades, and there are even methods that can exactly match the lower complexity bounds [68, 34, 142, 36]. On the other hand, in general non-convex optimization, near-stationarity is the typical optimality measure, and there has been a flurry of recent research devoted to this topic [47, 48, 45, 63, 42, 157]. Recently, there has been growing interest on devising fast schemes for finding near-stationary points in convex optimization [106, 2, 43, 18, 70, 69, 71, 62, 32, 31, 84]. This line of research is driven by the following applications and facts.

- Nesterov [106] studied the problem that has a linear constraint:

$$f(x^\star)=\min_{x\in Q}\{f(x) : Ax=b\},$$

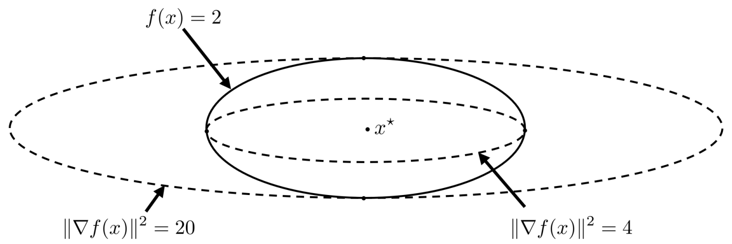


Figure 3.1: Contours of $f(x) = \frac{x_1^2+5x_2^2}{2}$ [114] and its squared gradient norm $\|\nabla f(x)\|^2$.

where Q is a convex set and f is strongly convex. Assuming that Q and f are simple, we can focus on the dual problem

$$\phi(y^\star) = \max_y\{\phi(y) \triangleq \min_{x\in Q}\{f(x) + \langle y, b - Ax\rangle\}\}.$$

Clearly, the dual objective $-\phi(y)$ is smooth convex. Letting x_y be the unique solution to the inner problem, we have $\nabla\phi(y) = b - Ax_y$. Note that

$$f(x_y) - f(x^\star) = \phi(y) - \langle y, \nabla\phi(y)\rangle - \phi(y^\star) \le \|y\| \|\nabla\phi(y)\| .$$

Thus, in this problem, the quantity $\|\nabla\phi(y)\|$ serves as a measure of both primal optimality $f(x_y) - f(x^\star)$ and feasibility $\|b - Ax_y\|$, which is better than just measuring the function value.

- Matrix scaling [129] is a convex problem and its goal is to find near-stationary points [7, 22].
- Gradient norm is readily available, unlike other optimality measures ($f(x) - f(x^\star)$ and $\|x - x^\star\|$), and is thus usable as a stopping criterion. This fact motivates the design of several parameter-free algorithms [108, 92, 62], and their guarantees are established on the gradient norm.
- Designing schemes for minimizing the gradient norm can inspire new non-convex optimization methods. For example, SARAH [111] was designed for convex finite-sums with gradient-norm measure, but was later discovered to be the near-optimal method for non-convex finite-sums [42, 121].

Moreover, function value and gradient norm may have very different geometries around the minimizer (see Figure 3.1), and minimizing the gradient norm is often considered to be a harder task than minimizing the function value, because NAG has the optimal rate for $f(x) - f(x^\star)$ but is only suboptimal for minimizing $\|\nabla f(x)\|$.

Table 3.1: Finding near-stationary points $\|\nabla f(x)\| \le \epsilon$ of convex finite-sums.

	Algorithm		**Complexity**	**Remark**
IFC	GD	[71]	$O(\frac{n}{\epsilon^2})$	
	Regularized NAG*	[18]	$O(\frac{n}{\epsilon}\log\frac{1}{\epsilon})$	
	OGM-G	[71]	$O(\frac{n}{\epsilon})$	$O(\frac{1}{\epsilon}+d)$ memory, optimal in ϵ
	M-OGM-G	[Section 3.2.1]	$O(\frac{n}{\epsilon})$	$O(d)$ memory, optimal in ϵ
	L2S	[88]	$O(n+\frac{\sqrt{n}}{\epsilon^2})$	Loopless variant of SARAH [111]
	Regularized Katyusha*	[2]	$O((n+\frac{\sqrt{n}}{\epsilon})\log\frac{1}{\epsilon})$	Requires the knowledge of Δ_0
	R-Acc-SVRG-G*	[Section 3.4]	$O((n\log\frac{1}{\epsilon}+\frac{\sqrt{n}}{\epsilon})\log\frac{1}{\epsilon})$	Without the knowledge of Δ_0
IDC	GD	[106, 141]	$O(\frac{n}{\epsilon})$	
	NAG / NAG + GD	[69] / [106]	$O(\frac{n}{\epsilon^{2/3}})$	
	Regularized NAG*	[106, 62]	$O(\frac{n}{\sqrt{\epsilon}}\log\frac{1}{\epsilon})$	
	NAG + OGM-G	[110]	$O(\frac{n}{\sqrt{\epsilon}})$	$O(\frac{1}{\sqrt{\epsilon}}+d)$ memory, optimal in ϵ
	NAG + M-OGM-G	[Section 3.2.1]	$O(\frac{n}{\sqrt{\epsilon}})$	$O(d)$ memory, optimal in ϵ
	Katyusha + L2S	[Appendix B.5]	$O(n\log\frac{1}{\epsilon}+\frac{\sqrt{n}}{\epsilon^{2/3}})$	
	Acc-SVRG-G	[Section 3.3]	$O\Big(n\log\frac{1}{\epsilon}+\min\Big\{\frac{n^{2/3}}{\epsilon^{2/3}},\frac{\sqrt{n}}{\epsilon}\Big\}\Big)$	$O(n\log\frac{1}{\epsilon}+\sqrt{\frac{n}{\epsilon}})$ for function at the same time, simple and elegant
	Regularized Katyusha*	[2]	$O((n+\sqrt{\frac{n}{\epsilon}})\log\frac{1}{\epsilon})$	Requires the knowledge of R_0
	R-Acc-SVRG-G*	[Section 3.4]	$O((n\log\frac{1}{\epsilon}+\sqrt{\frac{n}{\epsilon}})\log\frac{1}{\epsilon})$	Without the knowledge of R_0

* Indirect methods (using regularization).

In this chapter, we focus on finding the near-stationary points of convex finite-sum problems. Since non-strongly convex problem may have multiple solutions, we use $\mathcal{X}^\star$ to denote the set of optimal solutions, which is assumed to be nonempty. There are two different assumptions on the initial point x_0, namely, the Initial bounded-Function Condition (**IFC**): $f(x_0) - f(x^\star) \le \Delta_0$, and the Initial bounded-Distance Condition (**IDC**): $\|x_0 - x^\star\| \le R_0$ for some $x^\star \in \mathcal{X}^\star$. This subtlety results in drastically different best achievable rates as studied in [18, 43]. Below we categorize existing techniques into three classes (relating to Table 3.1).

(i) *"IDC + IFC"*. Nesterov [106] showed that we can combine the guarantees of a method minimizing function value under IDC and a method finding near-stationary points under IFC to produce a faster one for minimizing gradient norm under IDC. For example, NAG produces $f(x_{K_1}) - f(x^\star) = O(\frac{LR_0^2}{K_1^2})$ [104] and GD produces $\|\nabla f(x_{K_2})\|^2 = O\big(\frac{L(f(x_0)-f(x^\star))}{K_2}\big)$ [71] under IFC. Letting $x_0 = x_{K_1}$ and $K = K_1 + K_2$, by balancing the ratio of K_1 and K_2, we obtain the guarantee $\|\nabla f(x_K)\|^2 = O(\frac{L^2R_0^2}{K^3})$ for "NAG + GD" (same for "NAG + OGM-G"). We point out that we can use this technique to combine the guarantees of Katyusha [4] and SARAH[1] [111]; see Appendix B.5.

[1]We adopt a loopless variant of SARAH [88], which has a refined analysis for general convex objectives.

(ii) *Regularization.* Nesterov [106] used NAG (the strongly convex variant) to solve the regularized objective, and showed that it achieves near-optimal complexity (optimal up to log factors). Inspired by this technique, Allen-Zhu [2] proposed recursive regularization for stochastic approximation algorithms, which also achieves near-optimal complexities [43].

(iii) *Direct methods.* Due to the lack of insight, existing direct methods are mostly derived or analyzed with the help of computer-aided tools [70, 69, 141, 71]. The computer-aided approach was pioneered by Drori and Teboulle [37], who introduced the performance estimation problem (PEP). The only known optimal method OGM-G [71] was designed based on the PEP approach.

Observe that since $f(x) - f(x^\star) \leq \|\nabla f(x)\| \, \|x - x^\star\|$, the lower bound for finding near-stationary points must be of the same order as for minimizing function value [109]. Thus, under IDC, the lower bound is $\Omega(n + \sqrt{\frac{n}{\epsilon}})$ due to [151]. Under IFC, we can establish an $\Omega(n + \frac{\sqrt{n}}{\epsilon})$ lower bound using the techniques in [18, 151]. The main contributions of this chapter are three new schemes that improve the practicalities of existing methods, which is summarized below (highlighted in Table 3.1).

- (Section 3.2) We propose a memory-saving variant of OGM-G for the deterministic case ($n = 1$), which does not require pre-computed and stored parameters. The derivation of the new variant is inspired by the numerical solution to a PEP problem.

- (Section 3.3) We propose a new accelerated SVRG [64, 152] variant that can *simultaneously* achieve fast rates for minimizing both the gradient norm and function value, that is, $O(n \log \frac{1}{\epsilon} + \min\{\frac{n^{2/3}}{\epsilon^{2/3}}, \frac{\sqrt{n}}{\epsilon}\})$ complexity for gradient norm and $O(n \log \frac{1}{\epsilon} + \sqrt{\frac{n}{\epsilon}})$ complexity for function value. Other stochastic approaches in Table 3.1 do not have this property.

- (Section 3.4) We propose an adaptively regularized accelerated SVRG variant, which does not require the knowledge of R_0 or Δ_0 and achieves a near-optimal complexity under IDC or IFC.

We put in extra efforts to make the proposed schemes as simple and elegant as possible. We believe that the simplicity makes the extensions of the new schemes easier.

3.2 OGM-G: "Momentum" Reformulation and a Memory-Saving Variant

In this section, we focus on the IFC setting, that is, $f(x_0) - f(x^\star) \leq \Delta_0$. We use N to denote the total number of iterations (each computes a full gradient ∇f). Proofs in this section are given in Appendix B.2. Recall that OGM-G has the following updates [71]. Let $y_0 = x_0$. For $k = 0, \ldots, N-1$,

$$\begin{aligned} y_{k+1} &= x_k - \frac{1}{L}\nabla f(x_k), \\ x_{k+1} &= y_{k+1} + \frac{(\theta_k - 1)(2\theta_{k+1} - 1)}{\theta_k(2\theta_k - 1)}(y_{k+1} - y_k) + \frac{2\theta_{k+1} - 1}{2\theta_k - 1}(y_{k+1} - x_k), \end{aligned} \tag{3.1}$$

where the sequence $\{\theta_k\}$ is recursively defined:

$$\theta_N = 1 \text{ and } \begin{cases} \theta_k^2 - \theta_k = \theta_{k+1}^2 & k = 1 \ldots N-1, \\ \theta_0^2 - \theta_0 = 2\theta_1^2 & \text{otherwise.} \end{cases}$$

OGM-G was discovered from the numerical solution to an SDP problem and its analysis is to show that the step coefficients in (3.1) specify a feasible solution to the SDP problem. While this analysis is natural for the PEP approach, it is hard to understand how each coefficient affects the rate, especially if one wants to generalize the scheme. Here we provide a simple algebraic analysis for OGM-G.

We start with a reformulation[2] of OGM-G in Algorithm 4, which aims to simplify the proof. We adopt a consistent sequence $\{\theta_k\}$: $\theta_N = 1$ and $\theta_k^2 - \theta_k = \theta_{k+1}^2$, $k = 0 \ldots N-1$, which only costs a constant factor.[3] Interestingly, the reformulated scheme resembles the heavy-ball momentum method [123]. However, it can be shown that Algorithm 4 is not covered by the heavy-ball momentum scheme. Defining $\theta_{N+1}^2 = \theta_N^2 - \theta_N = 0$, we provide the one-iteration analysis in the following proposition:

Proposition 3.1. *In Algorithm 4, the following holds at iteration $k \in \{0, \ldots, N-1\}$.*

$$\begin{aligned} A_k + B_{k+1} + C_{k+1} + E_{k+1} \leq A_{k+1} + B_k + C_k + E_k - \theta_{k+1} \langle \nabla f(x_{k+1}), v_{k+1} \rangle \\ + \sum_{i=k+1}^{N} \frac{\theta_i}{L\theta_k\theta_{k+1}^2} \langle \nabla f(x_k), \nabla f(x_i) \rangle, \end{aligned} \tag{3.2}$$

[2]It can be verified that this scheme is equivalent to the original one (3.1) through $v_k = \frac{1}{(2\theta_k - 1)\theta_k^2}(y_k - x_k)$.

[3]The original guarantee of OGM-G can be recovered if we set $\theta_0^2 - \theta_0 = 2\theta_1^2$.

Algorithm 4 OGM-G: "Momentum" reformulation

```
Input: initial guess $x_0 \in \mathbb{R}^d$, total iteration number $N$.
Initialize: vector $v_0 = \mathbf{0}$, scalars $\theta_N = 1$ and $\theta_k^2 - \theta_k = \theta_{k+1}^2$, for $k = 0 \dots N-1$.
1: for $k = 0, \dots, N-1$ do
2:     $v_{k+1} = v_k + \frac{1}{L\theta_k\theta_{k+1}^2}\nabla f(x_k)$.
3:     $x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k) - (2\theta_{k+1}^3 - \theta_{k+1}^2)v_{k+1}$.
4: end for
Output: $x_N$.
```

where $A_k \triangleq \frac{1}{\theta_k^2}(f(x_N) - f(x^\star) - \frac{1}{2L}\|\nabla f(x_N)\|^2)$, $B_k \triangleq \frac{1}{\theta_k^2}(f(x_k) - f(x^\star))$, $C_k \triangleq \frac{1}{2L\theta_k^2}\|\nabla f(x_k)\|^2$ *and* $E_k \triangleq \frac{\theta_{k+1}^2}{\theta_k}\langle \nabla f(x_k), v_k\rangle$.

Remark 3.1.1. *A recent work [32] also conducted an algebraic analysis of OGM-G under a potential function framework. Their potential function decrease can be directly obtained from Proposition 3.1 by summing up* (3.2)*. By contrast, our "momentum" vector* $\{v_k\}$ *naturally merges into the analysis, which significantly simplifies the analysis. Moreover, it provides a better interpretation on how OGM-G utilizes the past gradients to achieve acceleration. Lee et al. [84] discovered the potential function of OGM-G while their analysis is much more complicated.*

From (3.2), we see that only the last two terms do not telescope. Note that the "momentum" vector is a weighted sum of past gradients, $v_{k+1} = \sum_{i=0}^{k} \frac{1}{L\theta_i\theta_{i+1}^2}\nabla f(x_i)$. If we sum the terms up from $k = 0, \dots, N-1$, it can be verified that they exactly sum up to 0. Then, by telescoping the remaining terms, we obtain the final guarantee.

Theorem 4. *The output of Algorithm 4 satisfies* $\|\nabla f(x_N)\|^2 \le \frac{8L\Delta_0}{(N+2)^2}$.

We observe two drawbacks of OGM-G (which have been similarly pointed out in [32, 84]): (1) it requires storing a pre-computed parameter sequence, which costs $O(\frac{1}{\epsilon})$ floats; (2) except for the last iterate, all other iterates do not have properly upper-bounded gradient norms. We resolve these issues by proposing another parameterization of Algorithm 4 in the next subsection.

3.2.1 Memory-Saving OGM-G

A straightforward idea to resolve the aforementioned issues is to generalize Algorithm 4. However, we find it rather difficult since the parameters in the analysis are rather

Algorithm 5 M-OGM-G: Memory-saving OGM-G

```
Input: initial guess $x_0 \in \mathbb{R}^d$, total iteration number $N$.
Initialize: vector $v_0 = \mathbf{0}$.
1: for $k = 0, \ldots, N-1$ do
2:     $v_{k+1} = v_k + \frac{12}{L(N-k+1)(N-k+2)(N-k+3)} \nabla f(x_k)$.
3:     $x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k) - \frac{(N-k)(N-k+1)(N-k+2)}{6} v_{k+1}$.
4: end for
Output: $x_N$ or $\arg\min_{x \in \{x_0, \ldots, x_N\}} \|\nabla f(x)\|$.
```

strict (despite that the proof is already simple). We choose to rely on computer-aided techniques [37]. The derivation of this variant (Algorithm 5) is based on the following numerical experiment.

Numerical Experiment. OGM-G was discovered from the relaxed PEP [71]:

$$\max_{\substack{\nabla f(x_0),\ldots,\nabla f(x_N) \in \mathbb{R}^d \\ f(x_0),\ldots,f(x_N),f(x^\star) \in \mathbb{R}}} \|\nabla f(x_N)\|^2$$

$$\text{subject to} \begin{cases} \text{interpolation condition (1.2) at } (x_k, x_{k+1}), & k = 0, \ldots, N-1, \\ \text{interpolation condition (1.2) at } (x_N, x_k), & k = 0, \ldots, N-1, \\ \text{interpolation condition (1.2) at } (x_N, x^\star), & f(x_0) - f(x^\star) \le \Delta_0, \end{cases} \tag{P}$$

where the sequence $\{x_k\}$ is defined as $x_{k+1} = x_k - \frac{1}{L}\sum_{i=0}^{k} h_{k+1,i} \nabla f(x_i), k = 0, \ldots, N-1$ for some step coefficients $h \in \mathbb{R}^{N(N+1)/2}$. Given N, the step coefficients of OGM-G correspond to a numerical solution to the problem: $\arg\min_h \{\text{Lagrangian dual of (P)}\}$, which is denoted as (HD). Conceptually, solving problem (HD) would give us the fastest possible step coefficients under the constraints.[4] We expect there to be some constant-time slower schemes, which are neglected when solving (HD). To identify them, we relax a set of interpolation conditions in problem (P):

$$f(x_N) - f(x_k) - \langle \nabla f(x_k), x_N - x_k \rangle \ge \frac{1}{2L} \|\nabla f(x_N) - \nabla f(x_k)\|^2 - \rho \|\nabla f(x_k)\|^2,$$

for $k = 0, \ldots, N-1$ and some $\rho > 0$. After this relaxation, solving (HD) will no longer give us the step coefficients of OGM-G. Moreover, the subtracted term $\rho \|\nabla f(x_k)\|^2$ forces the PEP tool to not "utilize" it (to cancel out other terms) when searching for

[4]However, since problem (HD) is non-convex, we can only obtain local solutions.

step coefficients. Since such a term is not "utilized" in each of the N interpolation conditions, after summation, these terms appear on the left hand side of (3.3), which gives upper bounds to the gradient norms evaluated at intermediate iterates. By trying different ρ and checking the dependence on N, we discover Algorithm 5 when $\rho = \frac{1}{2L}$. Similar to our analysis of OGM-G, we provide a simple algebraic analysis in the following theorem.

Theorem 5. *Define* $\delta_{k+1} \triangleq \frac{12}{(N-k+1)(N-k+2)(N-k+3)}, k = 0, \ldots, N$. *In Algorithm 5, it holds that*

$$\sum_{k=0}^{N} \frac{\delta_{k+1}}{2} \left\|\nabla f(x_k)\right\|^2 \leq \frac{12L\Delta_0}{(N+2)(N+3)}. \tag{3.3}$$

Remark 5.1. *From* (3.3)*, we can directly conclude that* $\forall k \in \{0, \ldots, N\}, \left\|\nabla f(x_k)\right\|^2 = O(\frac{L\Delta_0}{N^2 \delta_{k+1}})$ *and thus, the rate (in terms of N) on the last iterate is optimal (since* $\delta_{N+1} = 2$*). Moreover, the minimum gradient also achieves the optimal rate since*

$$\min_{k\in\{0,\ldots,N\}} \left\|\nabla f(x_k)\right\|^2 \leq \frac{1}{\sum_{k=0}^{N} \frac{\delta_{k+1}}{2}} \sum_{k=0}^{N} \frac{\delta_{k+1}}{2} \left\|\nabla f(x_k)\right\|^2 \leq \frac{8L\Delta_0}{(N+2)(N+3)-2}.$$

Clearly, the parameters of this variant can be computed on the fly and from the above remark, each iterate has an upper-bounded gradient norm. The constructions in [32, 84] all require pre-computed and stored sequences, which seems to be unavoidable in their analysis as admitted in [32]. Our discovery is another example of the powerfulness of computer-aided methodology, which finds proofs that are difficult or even impossible to find with bare hands. We can extend the benefits into the IDC setting using the ideas in [106] as summarized below.

Corollary 5.1 (IDC case)**.** *If we first run $N/2$ iterations of NAG and then continue with $N/2$ iterations of Algorithm 5, the output satisfies* $\left\|\nabla f(x_N)\right\|^2 = O\big(\frac{L^2 R_0^2}{N^4}\big)$.

3.3 Accelerated SVRG: Fast Rates for Both Gradient Norm and Objective

In this section, we focus on the IDC setting, that is, $\|x_0 - x^\star\| \leq R_0$ for some $x^\star \in \mathcal{X}^\star$. We use K to denote the total number of stochastic iterations. From the development in Section 3.2, it is natural to ask whether we can use the PEP approach to motivate new stochastic schemes. However, due to the exponential growth of the number of possible states $(i_0, i_1, \ldots)$, we cannot directly adopt this

Algorithm 6 Acc-SVRG-G: Accelerated SVRG for Gradient minimization

```
Input: parameters $\{\tau_k\}$, $\{p_k\}$, initial guess $x_0 \in \mathbb{R}^d$, total iteration number $K$.
Initialize: vectors $z_0 = \tilde{x}_0 = x_0$ and scalars $\alpha_k = \frac{L\tau_k}{1-\tau_k}, \forall k$ and $\widetilde{\tau} = \sum_{k=0}^{K-1} \tau_k^{-2}$.
1: for $k = 0, \ldots, K-1$ do
2:     $y_k = \tau_k z_k + (1-\tau_k)\left(\tilde{x}_k - \frac{1}{L}\nabla f(\tilde{x}_k)\right)$.
3:     $z_{k+1} = \arg\min_x \left\{\langle \mathcal{G}_k, x\rangle + (\alpha_k/2)\left\|x - z_k\right\|^2\right\}$.
4:     // $\mathcal{G}_k \triangleq \nabla f_{i_k}(y_k) - \nabla f_{i_k}(\tilde{x}_k) + \nabla f(\tilde{x}_k)$, where $i_k$ is sampled uniformly in $[n]$.
5:     $\tilde{x}_{k+1} = \begin{cases} y_k & \text{with probability } p_k, \\ \tilde{x}_k & \text{with probability } 1-p_k. \end{cases}$
6: end for
Output (for gradient): $x_{\text{out}}$ is sampled from
    $\left\{\text{Prob}\{x_{\text{out}} = \tilde{x}_k\} = \frac{\tau_k^{-2}}{\widetilde{\tau}} \,\middle|\, k \in \{0, \ldots, K-1\}\right\}$.
Output (for function value): $\tilde{x}_K$.
```

approach. A feasible alternative is to first fix an algorithmic framework and a family of potential functions, and then use the potential-based PEP approach in [141]. However, this approach is much more restrictive. For example, it cannot identify special constructions like (3.2) in OGM-G. Fortunately, as we will see, we can get some inspiration from the recent development of deterministic methods. Proofs in this section are given in Appendix B.3.

Our proposed scheme is given in Algorithm 6. We adopt the elegant loopless design of SVRG in [75]. Note that the full gradient $\nabla f(\tilde{x}_k)$ is computed and stored only when $\tilde{x}_{k+1} = y_k$ at Step 5. We summarize our main technical novelty as follows.

Main Algorithmic Novelty. The design of stochastic accelerated methods is largely inspired by NAG. To make it clear, by setting $n = 1$, we see that Katyusha [4], MiG [158], SSNM [159], Varag [81], VRADA [138], ANITA [89], the acceleration framework in [33] and AC-SA [78, 46, 160] all reduce to one of the following variants of NAG [9, 5]. We say that these methods are under the NAG framework.

$$\begin{cases} x_k = \tau_k z_k + (1-\tau_k)y_k, \\ z_{k+1} = z_k - \alpha_k \nabla f(x_k), \\ y_{k+1} = \tau_k z_{k+1} + (1-\tau_k)y_k. \end{cases}$$

Auslender and Teboulle [9]

$$\begin{cases} x_k = \tau_k z_k + (1-\tau_k)y_k, \\ z_{k+1} = z_k - \alpha_k \nabla f(x_k), \\ y_{k+1} = x_k - \eta_k \nabla f(x_k). \end{cases}$$

Linear Coupling [5]

See [148, 28] for other variants of NAG. When $n = 1$, Algorithm 6 reduces to the following scheme.

$$\begin{cases} y_k = \tau_k z_k + (1-\tau_k)\left(y_{k-1} - \frac{1}{L}\nabla f(y_{k-1})\right), \\ z_{k+1} = z_k - \frac{1}{\alpha_k}\nabla f(y_k). \end{cases}$$

Optimized Gradient Method (OGM) [37, 68]

Algorithm 6 reduces to the scheme of OGM when $n = 1$ (this point is clearer in the formulation of ITEM in [142]). Note that although we use OGM as the inspiration, the original OGM has nothing to do with making the gradient small and there is no hint on how a stochastic variant can be designed. OGM has a constant-time faster worst-case rate than NAG, which exactly matches the lower complexity bound in [34]. In the following proposition, we show that the OGM framework helps us conduct a tight one-iteration analysis, which gives room for achieving our goal.

Proposition 5.1. *In Algorithm 6, the following holds at any iteration $k \geq 0$ and $\forall x^\star \in \mathcal{X}^\star$.*

$$\begin{aligned} &\left(\frac{1-\tau_k}{\tau_k^2 p_k}\mathbb{E}\left[f(\tilde{x}_{k+1}) - f(x^\star)\right] + \frac{L}{2}\mathbb{E}\left[\,\|z_{k+1} - x^\star\|^2\,\right]\right) + \frac{(1-\tau_k)^2}{2L\tau_k^2}\mathbb{E}\left[\,\|\nabla f(\tilde{x}_k)\|^2\,\right] \\ &\leq \left(\frac{(1-\tau_k p_k)(1-\tau_k)}{\tau_k^2 p_k}\mathbb{E}\left[f(\tilde{x}_k) - f(x^\star)\right] + \frac{L}{2}\mathbb{E}\left[\,\|z_k - x^\star\|^2\,\right]\right). \end{aligned} \tag{3.4}$$

The terms inside the parentheses form the commonly used potential function of SVRG variants. The additional $\mathbb{E}[\|\nabla f(\tilde{x}_k)\|^2]$ term is created by adopting the OGM framework. In other words, we use the following potential function for Algorithm 6 ($a_k, b_k, c_k \geq 0$):

$$T_k = a_k\mathbb{E}\left[f(\tilde{x}_k) - f(x^\star)\right] + b_k\mathbb{E}\left[\,\|z_k - x^\star\|^2\,\right] + \sum_{i=0}^{k-1} c_i\mathbb{E}\left[\,\|\nabla f(\tilde{x}_i)\|^2\,\right].$$

We first provide a simple parameter choice, which leads to a simple and clean analysis.

Theorem 6 (Single-stage parameter choice)**.** *In Algorithm 6, if $p_k \equiv \frac{1}{n}$, $\tau_k = \frac{3}{k/n+6}$, the following holds at the outputs.*

$$\begin{aligned} \mathbb{E}\left[\,\|\nabla f(x_{\text{out}})\|^2\,\right] &= O\left(\frac{n^3 L\big(f(x_0) - f(x^\star)\big) + n^2L^2R_0^2}{K^3}\right), \\ \mathbb{E}\left[f(\tilde{x}_K)\right] - f(x^\star) &= O\left(\frac{n^2\big(f(x_0) - f(x^\star)\big) + nLR_0^2}{K^2}\right). \end{aligned} \tag{3.5}$$

In other words, to guarantee $\mathbb{E}\big[\,\|\nabla f(x_{\text{out}})\|\,\big] \le \epsilon_g$ *and* $\mathbb{E}\big[f(\tilde{x}_K)\big] - f(x^\star) \le \epsilon_f$*, the complexities are* $O\Big(\frac{n(L(f(x_0)-f(x^\star)))^{1/3}}{\epsilon_g^{2/3}} + \frac{(nLR_0)^{2/3}}{\epsilon_g^{2/3}}\Big)$ *and* $O\Big(n\sqrt{\frac{f(x_0)-f(x^\star)}{\epsilon_f}} + \frac{\sqrt{n}LR_0}{\sqrt{\epsilon_f}}\Big)$*, respectively.*

From (3.5), we see that Algorithm 6 achieves fast $O(\frac{1}{K^{1.5}})$ and $O(\frac{1}{K^2})$ rates for minimizing the gradient norm and function value at the same time. However, despite being a simple choice, the oracle complexities are not better than the deterministic methods in Table 3.1. Below we provide a two-stage parameter choice, which is inspired by the idea of including a "warm-up phase" in [6, 81, 138, 89].

Theorem 7 (Two-stage parameter choice)**.** *In Algorithm 6, let* $p_k = \max\{\frac{6}{k+8}, \frac{1}{n}\}$*,* $\tau_k = \frac{3}{p_k(k+8)}$*. The oracle complexities needed to guarantee that* $\mathbb{E}\big[\,\|\nabla f(x_{\text{out}})\|\,\big] \le \epsilon_g$ *and that* $\mathbb{E}\big[f(\tilde{x}_K)\big] - f(x^\star) \le \epsilon_f$ *are*

$$O\left(n \min\left\{\log\frac{LR_0}{\epsilon_g}, \log n\right\} + \frac{(nLR_0)^{2/3}}{\epsilon_g^{2/3}}\right)$$
$$\textit{and } O\left(n \min\left\{\log\frac{LR_0^2}{\epsilon_f}, \log n\right\} + \frac{\sqrt{nL}R_0}{\sqrt{\epsilon_f}}\right),$$

respectively.

Since $\|\nabla f(\tilde{x}_K)\|^2 = O\big(L\big(f(\tilde{x}_K) - f(x^\star)\big)\big)$, the last iterate has the complexity $O(n\log\frac{1}{\epsilon} + \frac{\sqrt{n}}{\epsilon})$ for minimizing the gradient norm. Then, by outputting the $\tilde{x}$ that attains the minimum gradient, we can combine the results of outputting x_{out} and $\tilde{x}_K$, which leads to the complexity $O(n\log\frac{1}{\epsilon} + \min\{\frac{n^{2/3}}{\epsilon^{2/3}}, \frac{\sqrt{n}}{\epsilon}\})$ in Table 3.1. This complexity has a slightly worse dependence on n than Katyusha + L2S. It is due to the adoption of n-dependent step size in L2S. As studied in [88], despite having a better complexity, n-dependent step size boosts numerical performance only when n is *extremely large*. If the practically fast n-independent step size is used for L2S, Katyusha+L2S and Acc-SVRG-G have similar complexities. See also Appendix B.1.

If ϵ is large or n is very large, the recently proposed ANITA [89] achieves an $O(n)$ complexity, which matches the lower complexity bound $\Omega(n)$ in this case [151]. Since ANITA uses the NAG framework, we show that similar results can be derived under the OGM framework in the following theorem:

Theorem 8 (Low accuracy parameter choice)**.** *In Algorithm 6, let iteration* N *be the first time Step 5 updates* $\tilde{x}_{k+1} = y_k$*. If we choose* $p_k \equiv \frac{1}{n}$*,* $\tau_k \equiv 1 - \frac{1}{\sqrt{n+1}}$ *and* *terminate Algorithm 6 at iteration N, then the following holds at $\tilde{x}_{N+1}$:*

$$\mathbb{E}\big[\,\|\nabla f(\tilde{x}_{N+1})\|^2\,\big] \le \frac{8L^2R_0^2}{5(\sqrt{n+1}+1)} \text{ and } \mathbb{E}\big[f(\tilde{x}_{N+1})\big] - f(x^\star) \le \frac{LR_0^2}{\sqrt{n+1}+1},$$

In particular, if the required accuracies are low (or n is very large), i.e., $\epsilon_g^2 \ge \frac{8L^2R_0^2}{5(\sqrt{n+1}+1)}$ and $\epsilon_f \ge \frac{LR_0^2}{\sqrt{n+1}+1}$, then Algorithm 6 only has an $O(n)$ oracle complexity.

In the low accuracy region (specified above), the choice in Theorem 8 removes the $O(\log \frac{1}{\epsilon})$ factor in the complexity of Theorem 7. From the above two theorems, we see that Algorithm 6 achieves a similar rate for minimizing the function value as ANITA [89], which is the current best rate. We include some numerical justifications of Algorithm 6 in Appendix B.1. We believe that the potential-based PEP approach in [141] can help us identify better parameter choices of Algorithm 6, which we leave for future work.

3.4 Near-Optimal Accelerated SVRG with Adaptive Regularization

Currently, there is no known stochastic method that directly achieves the optimal rate in ϵ. To get near-optimal rates, the existing strategy is to use a carefully designed regularization technique [106, 2] with a method that solves strongly convex problems; see, e.g., [106, 2, 43, 26]. However, the regularization parameter requires the knowledge of R_0 or Δ_0, which significantly limits its practicality.

Inspired by the recently proposed adaptive regularization technique [62], we develop a near-optimal accelerated SVRG variant (Algorithm 7) that does not require the knowledge of R_0 or Δ_0. Note that this technique was originally proposed for NAG under the IDC assumption. Our development extends this technique to the stochastic setting, which brings an $O(\sqrt{n})$ rate improvement compared with adaptive regularized NAG. Moreover, we consider both IFC and IDC settings. Proofs in this section are in Appendix B.4.

Detailed Design. Algorithm 7 has a "guess-and-check" framework. In the outer loop, we first define the regularized objective f^{δ_t} using the current estimate of regularization parameter δ_t, and then we initialize an accelerated SVRG method (the inner loop) to solve the δ_t-strongly convex f^{δ_t}. If the inner loop breaks at Step 9 or

[5]Note that we maintain the full gradient $\nabla f^{\delta_t}(\tilde{x}_k)$ and $\nabla f(\tilde{x}_k) = \nabla f^{\delta_t}(\tilde{x}_k) - \delta_t(\tilde{x}_k - x_0)$.

Algorithm 7 R-Acc-SVRG-G

```
Input: accuracy $\epsilon > 0$, parameters $\delta_0 = L, \beta > 1$, initial guess $x_0 \in \mathbb{R}^d$.
 1: for $t = 0, 1, 2, \ldots$ do
 2:     Define $f^{\delta_t}(x) = (1/n)\sum_{i=1}^n f_i^{\delta_t}(x)$, where $f_i^{\delta_t}(x) = f_i(x) + (\delta_t/2)\left\|x - x_0\right\|^2$.
 3:     Initialize vectors $z_0 = \tilde{x}_0 = x_0$ and set $\tau_x, \tau_z, \alpha, p, C_{\mathrm{IDC}}, C_{\mathrm{IFC}}$ according to
        Proposition 8.1.
 4:     for $k = 0, 1, 2, \ldots$ do
 5:         $y_k = \tau_x z_k + (1 - \tau_x)\,\tilde{x}_k + \tau_z\left(\delta_t(\tilde{x}_k - z_k) - \nabla f^{\delta_t}(\tilde{x}_k)\right)$.
 6:         $z_{k+1} = \arg\min_x \left\{ \left\langle \mathcal{G}_k^{\delta_t}, x\right\rangle + (\alpha/2)\left\|x - z_k\right\|^2 + (\delta_t/2)\left\|x - y_k\right\|^2 \right\}$, where
            $\mathcal{G}_k^{\delta_t} \triangleq \nabla f_{i_k}^{\delta_t}(y_k) - \nabla f_{i_k}^{\delta_t}(\tilde{x}_k) + \nabla f^{\delta_t}(\tilde{x}_k)$, and $i_k$ is sampled uniformly in $[n]$.
 7:         $\tilde{x}_{k+1} = \begin{cases} y_k & \text{with probability } p, \\ \tilde{x}_k & \text{with probability } 1-p. \end{cases}$
 8:         if [5]$\left\|\nabla f(\tilde{x}_k)\right\| \leq \epsilon$ then output $\tilde{x}_k$ and terminate the algorithm.
 9:         if under IDC and $(1 + \frac{\delta_t}{\alpha})^k \geq \sqrt{C_{\mathrm{IDC}}/\delta_t}$ then break the inner loop.
10:         if under IFC and $(1 + \frac{\delta_t}{\alpha})^k \geq \sqrt{C_{\mathrm{IFC}}/2\delta_t}$ then break the inner loop.
11:     end for
12:     $\delta_{t+1} = \delta_t/\beta$.
13: end for
```

10, indicating the poor quality[6] of the current estimate δ_t, δ_t will be divided by a fixed β. Thus, conceptually, we can adopt any method that solves strongly convex finite-sums at the optimal rate as the inner loop. However, since the constructions of Step 9 or 10 require some algorithm-dependent constants, we have to fix one method as the inner loop.

The inner loop we adopted is a loopless variant of BS-SVRG (Algorithm 2) in Chapter 2. This is because (i) BS-SVRG is the fastest known accelerated SVRG variant (for ill-conditioned problems) and (ii) it has a simple scheme, especially after using the loopless construction [75]. However, its original guarantee (Theorem 2) is built upon $\{z_k\}$. Clearly, we cannot implement the stopping criterion (Step 8) on $\|\nabla f(z_k)\|$. Interestingly, we discover that its sequence $\{\tilde{x}_k\}$ works perfectly in our regularization framework, even if we can neither establish convergence on $f(\tilde{x}_k) - f(x^\star)$ nor on $\|\tilde{x}_k - x^\star\|^2$.[7] Moreover, we find that the loopless construction significantly

[6]If Algorithm 7 does not terminate before it breaks at Step 9 or 10 for the current estimate δ_t, it is quite likely that running infinite number of inner iterations, the algorithm still will not terminate.

[7]It is due to the special potential function of BS-SVRG (see (B.19)), which does not contain

simplifies the parameter constraints of BS-SVRG, which originally involves $\Theta(n)$th-order inequality. We provide the detailed parameter choice as follows:

Proposition 8.1 (Parameter choice)**.** *In Algorithm 7, we set* $\tau_x = \frac{\alpha+\delta_t}{\alpha+L+\delta_t}, \tau_z = \frac{\tau_x}{\delta_t} - \frac{\alpha(1-\tau_x)}{\delta_t L}$ *and* $p = \frac{1}{n}$*. We set* α *as the (unique) positive root of the cubic equation* $\left(1 - \frac{p(\alpha+\delta_t)}{\alpha+L+\delta_t}\right)\left(1+\frac{\delta_t}{\alpha}\right)^2 = 1$ *and we specify* $C_{\text{IDC}} = L^2 + \frac{L\alpha^2 p}{L+(1-p)(\alpha+\delta_t)}, C_{\text{IFC}} = 2L + \frac{2L\alpha^2 p}{(L+(1-p)(\alpha+\delta_t))\delta_t}$*. Under these choices, we have* $\frac{\alpha}{\delta_t} = O\big(n + \sqrt{n(L/\delta_t+1)}\big), C_{\text{IDC}} = O\big((L+\delta_t)^2\big)$ *and* $C_{\text{IFC}} = O(L)$*.*

Under the choices of τ_x and τ_z, the α above is the optimal choice in our analysis. Then, we can characterize the progress of the inner loop in the following proposition:

Proposition 8.2 (The inner loop of Algorithm 7)**.** *Using the parameters specified in Proposition 8.1, after running the inner loop (Step 4-11) of Algorithm 7 for k iterations, we can conclude that*

(i) under IDC, i.e., $\|x_0 - x^\star\| \le R_0$ *for some* $x^\star \in \mathcal{X}^\star$*,*

$$\mathbb{E}\big[\,\|\nabla f(\tilde{x}_k)\|\,\big] \le \left(\delta_t + \left(1+\frac{\delta_t}{\alpha}\right)^{-k}\sqrt{C_{\text{IDC}}}\right)R_0,$$

(ii) under IFC, i.e., $f(x_0) - f(x^\star) \le \Delta_0$*,*

$$\mathbb{E}\big[\,\|\nabla f(\tilde{x}_k)\|\,\big] \le \left(\sqrt{2\delta_t} + \left(1+\frac{\delta_t}{\alpha}\right)^{-k}\sqrt{C_{\text{IFC}}}\right)\sqrt{\Delta_0}.$$

The above results motivate the construction of Step 9 and 10. For example, in the IDC setting, when the inner loop breaks at Step 9, using *(i)* above, we obtain $\mathbb{E}\big[\,\|\nabla f(\tilde{x}_k)\|\,\big] \le 2\delta_t R_0$. Then, by discussing the relative size of δ_t and a certain constant, we can estimate the complexity of Algorithm 7. The same methodology is used in the IFC setting.

Theorem 9 (IDC case)**.** *Denote* $\delta^\star_{\text{IDC}} = \frac{\epsilon q}{2R_0}$ *for some* $q \in (0,1)$ *and let the outer iteration* $t = \ell$ *be the first time*[8] $\delta_\ell \le \delta^\star_{\text{IDC}}$*. The following assertions hold.*

(i) At outer iteration ℓ*, Algorithm 7 terminates with probability at least* $1-q$*.*[9]

these two terms.

[8]We assume that ϵ is small such that $\max\{\delta^\star_{\text{IDC}}, \delta^\star_{\text{IFC}}\} \le \delta_0 = L$ for simplicity. In this case, $\ell > 0$.

[9]If Algorithm 7 does not terminate at outer iteration ℓ, it terminates at the next outer iteration with probability at least $1 - q/\beta$. That is, it terminates with higher and higher probability. The same goes for the IFC case.

(ii) The total expected oracle complexity of the $\ell+1$ *outer loops is*

$$O\left(\left(n\log\frac{LR_0}{\epsilon q}+\sqrt{\frac{nLR_0}{\epsilon q}}\right)\log\frac{LR_0}{\epsilon q}\right).$$

Theorem 10 (IFC case)**.** *Denote* $\delta^\star_{\mathrm{IFC}}=\frac{\epsilon^2q^2}{8\Delta_0}$ *for some* $q\in(0,1)$ *and let the outer iteration* $t=\ell$ *be the first time* $\delta_\ell\le\delta^\star_{\mathrm{IFC}}$*. The following assertions hold.*

(i) At outer iteration ℓ*, Algorithm 7 terminates with probability at least* $1-q$*.*

(ii) The total expected oracle complexity of the $\ell+1$ *outer loops is*

$$O\left(\left(n\log\frac{\sqrt{L\Delta_0}}{\epsilon q}+\frac{\sqrt{nL\Delta_0}}{\epsilon q}\right)\log\frac{\sqrt{L\Delta_0}}{\epsilon q}\right).$$

Compared with regularized Katyusha in Table 3.1, the adaptive regularization approach drops the need to estimate R_0 or Δ_0 at the cost of a mere $\log\frac{1}{\epsilon}$ factor in the non-dominant term (if ϵ is small).

3.5 Chapter Summary and Discussion

In this chapter, we proposed several simple and practical schemes that complement existing works (Table 3.1). Admittedly, the new schemes are currently only limited to the unconstrained Euclidean setting, because our techniques heavily rely on the interpolation conditions (1.2) and (2.2). On the other hand, methods such as OGM [68], TM [134] and ITEM [142, 25], which also rely on these conditions, are still not known to have their proximal gradient variants. Lee et al. [84] proposed proximal point variants of these algorithms. Extending their techniques to our schemes is left for future work. Another future work is to conduct extensive experiments to evaluate the proposed schemes. We list some other future directions as follows.

(1) It is not clear how to naturally connect the parameters of M-OGM-G (Algorithm 5) to OGM-G (Algorithm 4). The parameters of both algorithms seem to be quite restrictive and hardly generalizable due to the special construction at (3.2).

(2) Is this new "momentum" in OGM-G beneficial for training deep neural networks? Other classic momentum schemes such as NAG [104] or heavy-ball momentum method [123] are extremely effective for this task (see, e.g., [140]), and they were also originally proposed for convex objectives.

(3) Can we directly accelerate SARAH (L2S)? It seems that existing acceleration techniques fail to accelerate SARAH (or result in poor dependence on n as in [33]). According to its position in Table 3.1, we suspect that there exists an accelerated variant of SARAH which reduces to OGM-G when $n=1$.

Chapter 4

Optimal Asynchronous Lock-Free Stochastic First-Order Method

In this chapter, we focus on the following smooth strongly convex finite-sum problem:

$$x^\star = \arg\min_{x \in \mathbb{R}^d} \left\{ f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}, \tag{4.1}$$

where each f_i is L-smooth and convex, f is μ-strongly convex.[1] We further assume that the computation of each f_i is *sparse*, i.e., the computation is supported on a set of coordinates $T_i \subseteq \{1, \ldots, d\}$. For example, generalized linear models[2] (GLMs) satisfy this assumption. GLMs has the form: $\forall i, f_i(x) = \ell_i(\langle a_i, x \rangle)$, where ℓ_i is some loss function and $a_i \in \mathbb{R}^d$ is a data sample. In this case, the support of each f_i is the set of non-zero coordinates of a_i. Many large real-world datasets have very high sparsity (see Table 4.1 for several examples).

To utilize this sparsity pattern, the lagged update technique [133] is purposed and can be adopted for existing variance reduced gradient estimators (such as SAG [130, 133], SVRG [64, 152], SAGA [29], S2GD [73], SARAH [111]). The benefits of lagged update are that it accelerates the gradient computation by only looking at the non-zero coordinates, and that it maintains the theoretic properties of the gradient estimators. The detailed procedure of lagged update is explained in Section 4.3.2. Another approach to handle the sparsity is to directly design sparse gradient estimators [98, 82], which is even more efficient but requires careful theoretic analysis.

[1] In fact, we will only use a weaker quadratic growth assumption, i.e., the strong convexity at any $x \in \mathbb{R}^d$ and $x^\star$, $f(x) - f(x^\star) \geq \frac{\mu}{2} \|x - x^\star\|^2$.

[2] The examples given in the introduction are all GLMs.

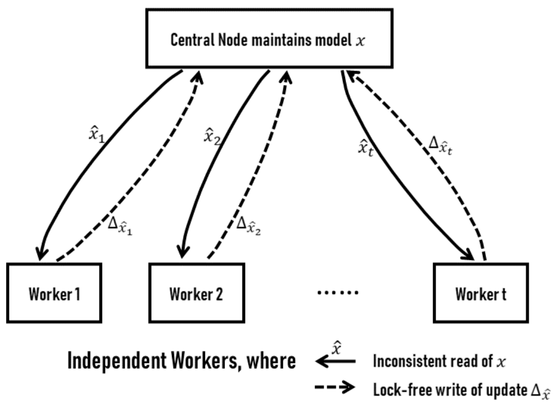

Central Node maintains model x
$\hat{x}_1$
$\Delta_{\hat{x}_1}$
$\hat{x}_2$
$\Delta_{\hat{x}_2}$
$\hat{x}_t$
$\Delta_{\hat{x}_t}$
Worker 1
Worker 2
......
Worker t
Independent Workers, where
$\hat{x}$ Inconsistent read of x
Lock-free write of update $\Delta_{\hat{x}}$

Figure 4.1: An asynchronous lock-free master-worker parallel computation model.

Inspired by the emerging parallel computing architectures such as multi-core computer and distributed system, many parallel variants of the aforementioned methods have been proposed, and there is a vast literature on those attempts. Among them, asynchronous lock-free algorithms are of special interest, which is depicted in Figure 4.1. Asynchrony means that each worker queries the central model x and sends update Δ_x to the central node almost independently. The key benefit of asynchrony is that the slowest worker is no longer the bottleneck of the system, and the lock-free design fully utilizes the parallel computing resource. Intuitively, the lock-free design would cause a lot of conflicts in the updates that happen simultaneously. Such conflicts could hurt the convergence. This is why the sparsity assumption is critical as it reduces the potential conflicts.

Existing Methods. Recht et al. [125] proposed the first asynchronous lock-free variant of SGD called Hogwild! and first proved that it can achieve a *linear speed-up*, i.e., running Hogwild! with ρ parallel processes only requires $O(1/\rho)$-times the running time of the serial variant. The condition on the maximum overlaps among the parallel processes is called the *linear speed-up condition.* Following Recht et al. [125], Sa et al. [132], Lian et al. [90] analyzed asynchronous SGD for non-convex problems, and Duchi et al. [40] analyzed for stochastic optimization; Mania et al. [98] refined the analysis framework (called the perturbed iterate framework) and proposed KroMagnon (asynchronous SVRG); Leblond et al. [82] simplified the perturbed iterate analysis and proposed ASAGA (asynchronous SAGA), and in an extended version of this work, Leblond et al. [83] further improved the analysis of KroMagnon and Hogwild!; Pedregosa et al. [120] derived proximal variant of ASAGA; Nguyen et al. [112]

refined the analysis of Hogwild!; Joulani et al. [65] proposed asynchronous methods in online and stochastic settings; Gu et al. [51] proposed several asynchronous variance reduced algorithms for non-smooth or non-convex problems; Stich et al. [139] studied a broad variety of SGD variants and improved the speed-up condition of asynchronous SGD.

All the above mentioned asynchronous methods are based on non-accelerated schemes such as SGD, SVRG and SAGA. Having witnessed the success of recently proposed stochastic accelerated methods (see Section 1.2), it is natural to ask:

Is it possible to achieve stochastic acceleration in asynchronous lock-free optimization? Will it lead to a worse linear speed-up condition?

The second question comes from a common perception that accelerated methods are less tolerant to gradient noise, e.g., [30, 24].

We answer these two questions in this chapter: We propose the first asynchronous lock-free stochastic accelerated method for solving problem (4.1), and we prove that it requires the identical linear speed-up condition as the non-accelerated counterparts.

The works that are most related to this chapter are [41, 158, 53, 153]. Fang et al. [41] proposed several asynchronous accelerated schemes. However, their analysis requires consistent read and does not consider sparsity. The dependence on the number of overlaps τ in their complexities is $O(\tau)$ compared with $O(\sqrt{\tau})$ in our result. Zhou et al. [158] naively extended their proposed accelerated method into asynchronous lock-free setting. However, even in the serial case, their result imposes strong assumptions on the sparsity. This issue is discussed in detail in Section 4.1.1. In the asynchronous case, no theoretical acceleration is achieved in [158]. Hannah et al. [53], Xiao et al. [153] proposed asynchronous accelerated block-coordinate descent methods (BCD). However, as pointed out in Appendix F in [120], BCD methods perform poorly in sparse problems since they focus only on a single coordinate of the gradient information.

4.1 Serial Sparse Accelerated SVRG

We first introduce a new accelerated SVRG variant with sparse updates in the serial case (Algorithm 8), which serves as the base algorithm for our asynchronous scheme. Our technique is built upon the following sparse approximated SVRG gradient estimator proposed in [98]: For uniformly random $i \in [n]$ and $y, \tilde{x} \in \mathbb{R}^d$, define

$$\mathcal{G}_y \triangleq \nabla f_i(y) - \nabla f_i(\tilde{x}) + D_i \nabla f(\tilde{x}), \tag{4.2}$$

Algorithm 8 SS-Acc-SVRG: Serial Sparse Accelerated SVRG

```
Input: initial guess $x_0 \in \mathbb{R}^d$, constant $\omega > 1$ which controls the restart frequency.
Initialize: set the scalars $m, \vartheta, \varphi, \eta, S, R$ according to Theorem 11, initialize the diagonal matrix $D$.
 1: for $r = 0, \ldots, R-1$ do                                   ▷ performing restarts
 2:     $\tilde{x}_0 = z_0^0 = x_r$.
 3:     for $s = 0, \ldots, S-1$ do
 4:         Compute and store $\nabla f(\tilde{x}_s)$.
 5:         for $k = 0, \ldots, m-1$ do
 6:             Sample $i_k$ uniformly in $[n]$ and let $T_{i_k}$ be the support of $f_{i_k}$.
 7:             $[y_k]_{T_{i_k}} = \vartheta [z_k^s]_{T_{i_k}} + (1-\vartheta)[\tilde{x}_s]_{T_{i_k}} - \varphi [D\nabla f(\tilde{x}_s)]_{T_{i_k}}$.   ▷ sparse coupling
 8:             $[z_{k+1}^s]_{T_{i_k}} = [z_k^s]_{T_{i_k}} - \eta\big(\nabla f_{i_k}([y_k]_{T_{i_k}}) - \nabla f_{i_k}([\tilde{x}_s]_{T_{i_k}}) + D_{i_k}\nabla f(\tilde{x}_s)\big)$.
 9:         end for
10:         $\tilde{x}_{s+1} = y_t$ for uniformly random $t \in \{0, 1, \ldots, m-1\}$.
11:         $z_0^{s+1} = z_m^s$.
12:     end for
13:     $x_{r+1} = \frac{1}{S}\sum_{s=0}^{S-1} \tilde{x}_{s+1}$.
14: end for
Output: $x_R$.
```

where $D_i \triangleq P_i D$ with $P_i \in \mathbb{R}^{d\times d}$ being the diagonal projection matrix of T_i (the support of f_i) and $D = (\frac{1}{n}\sum_{i=1}^n P_i)^{-1}$, which can be computed and stored in the first pass. Note that we assume the coordinates with zero cardinality have been removed in the objective function,[3] and thus $nI_d \succeq D \succeq I_d$. A diagonal element of D corresponds to the inverse probability of the coordinate belonging to a uniformly sampled support T_i. It is easy to verify that this construction ensures the unbiasedness $\mathbb{E}_i\big[D_i \nabla f(\tilde{x})\big] = \nabla f(\tilde{x})$.

This section is organized as follows: In Section 4.1.1, we summarize the key technical novelty and present the convergence result; in Section 4.1.2, we remark on the design of Algorithm 8.

4.1.1 Sparse Variance Correction

Intuitively, the sparse SVRG estimator (4.2) will lead to a larger variance compared with the dense one (when $D = I_d$). In the previous attempt, Zhou et al. [158] naively extends an accelerated SVRG variant into the sparse setting, which results in a fairly

[3]Clearly, the objective value is not supported on those coordinates.

strong restriction on the sparsity as admitted by the authors.[4] In contrast, sparse variants of SVRG and SAGA in [98, 82] require no assumption on the sparsity, and they achieve the same oracle complexities as their original dense versions.

Analytically speaking, the only difference of adopting the sparse estimator (4.2) is on the variance bound. The analysis of non-accelerated dense SVRG typically uses the following variance bound ($D = I_d$) [64, 152]: $\mathbb{E}\big[\,\|\mathcal{G}_y - \nabla f(y)\|^2\,\big] \leq 4L\big(f(y) - f(x^\star) + f(\tilde{x}) - f(x^\star)\big)$. It is shown in [98] that the sparse estimator (4.2) admits the same variance bound for any D. Thus, the analysis in the dense case can be directly applied to the sparse variant, which leads to a convergence guarantee that is independent of the sparsity.

However, things are not as smooth in the accelerated case. To (directly) accelerate SVRG, we typically uses a much tighter bound [4]: $\mathbb{E}\big[\,\|\mathcal{G}_y - \nabla f(y)\|^2\,\big] \leq 2L\big(f(\tilde{x}) - f(y) - \langle \nabla f(y), \tilde{x} - y\rangle\,\big)$ (in the dense case $D = I_d$). Unfortunately, in the sparse case ($D \neq I_d$), we do not have an identical variance bound as before. The variance of the sparse estimator (4.2) can be bounded as follows. The proof is given in Appendix C.1.

Lemma 3 (Variance bound for accelerated SVRG)**.** *The variance of* $\mathcal{G}_y$ (4.2) *can be bounded as*

$$\begin{aligned}\mathbb{E}_i\left[\|\mathcal{G}_y - \nabla f(y)\|^2\right] \leq{}& 2L\big(f(\tilde{x}) - f(y) - \langle \nabla f(y), \tilde{x} - y\rangle\,\big) - \|\nabla f(y)\|^2 \\ &+ \underbrace{2\,\langle \nabla f(y), D\nabla f(\tilde{x})\rangle}_{\mathcal{R}_1} - \langle \nabla f(\tilde{x}), D\nabla f(\tilde{x})\rangle\,. \end{aligned} \tag{4.3}$$

In general, except for the dense case, where we can drop the last three terms above by completing the square, this upper bound will always be correlated with the sparsity (i.e., D). This correlation causes the strong sparsity assumption in [158]. We may consider a more specific case where $D = nI_d$. In this case, the last three terms above can be written as $(n-1)\,\|\nabla f(y)\|^2 - n\,\|\nabla f(y) - \nabla f(\tilde{x})\|^2$, which is not always non-positive.

Inspecting (4.3), we see that it is basically the term $\mathcal{R}_1$, which could be positive, that causes the issue. We thus propose a novel *sparse variance correction* for accelerated SVRG, which is designed to perfectly cancel $\mathcal{R}_1$. The correction is added to the coupling step (Step 7):

$$y_k = \vartheta \cdot z_k + \underbrace{(1-\vartheta)\cdot \tilde{x}_s}_{\text{Negative Momentum}} - \underbrace{\varphi \cdot D\nabla f(\tilde{x}_s)}_{\text{Sparse Variance Correction}}\,.$$

This correction neutralizes all the negative effect of the sparsity, in a similar way as how the negative momentum cancels the term $\langle \nabla f(y), \tilde{x} - y\rangle$ in the analysis [4].

[4]It can be shown that when κ is large, almost no sparsity is allowed in their result.

We can understand this correction as a variance reducer that controls the additional sparse variance. Then we have the following sparsity-independent convergence result for Algorithm 8, and its proof is given in Appendix C.2.

Theorem 11. *For any constant $\omega > 1$, we choose $m = \Theta(n), \vartheta = \frac{\sqrt{m}}{\sqrt{\kappa}+\sqrt{m}}, \varphi = \frac{1-\vartheta}{L}$, $\eta = \frac{1-\vartheta}{L\vartheta}$ and $S = \lceil 2\omega\sqrt{\frac{\kappa}{m}}\rceil$. For any accuracy $\epsilon > 0$, we restart $R = O\big(\log\frac{f(x_0)-f(x^\star)}{\epsilon}\big)$ rounds. Then, Algorithm 8 outputs x_R satisfying $\mathbb{E}\big[f(x_R)\big] - f(x^\star) \leq \epsilon$ in oracle complexity*

$$O\left(\max\left\{n, \sqrt{\kappa n}\right\}\log\frac{f(x_0)-f(x^\star)}{\epsilon}\right).$$

This complexity matches the lower bound established in [151] (up to a log factor), and is substantially faster than the $O\big((n+\kappa)\log\frac{1}{\epsilon}\big)$ complexity of sparse approximated SVRG and SAGA derived in [82, 83] in the ill-conditioned regime ($\kappa \gg n$).

4.1.2 Other Remarks about Algorithm 8

- We adopt a restart framework to handle the strong convexity, which is also used in [158] and is suggested in [4] (footnote 9). This framework allows us to relax the strong convexity assumption to quadratic growth. Other techniques for handling the strong convexity such as (i) assuming a strongly convex regularizer and using proximal update [105, 4], and (ii) the direct constructions in [81, 161, 89] fail to keep the inner iterates sparse, and we are currently not sure how to modify them.

- The restarting point x_{r+1} is chosen as the averaged point instead of a uniformly random one because large deviations are observed in the loss curve if a random restarting point is used.

- We can also use sparse variance correction to fix the sparsity issue in [158], which leads to a slightly different method and somewhat longer proof.

- Algorithm 8 degenerates to a strongly convex variant of Acc-SVRG-G in Chapter 3 in the dense case ($D = I_d$), which summarizes our original inspiration.

- A general convex ($\mu = 0$) variant of Algorithm 8 can be derived by removing the restarts and adopting a variable parameter choice similar to Acc-SVRG-G in Chapter 3, which leads to a similar rate. We also find that our correction can be used in Varag [81] and ANITA [89] in the general convex case. Since the previous works on asynchronous lock-free optimization mainly focus on the strongly convex case, we omit the discussion here.

Algorithm 9 AS-Acc-SVRG: Asynchronous Sparse Accelerated SVRG

```
Input: initial guess $x_0 \in \mathbb{R}^d$, constant $\omega > 1$ which controls the restart frequency.
Initialize: set $m, \vartheta, \varphi, \eta, S, R$ according to Theorem 12, initialize shared variable $z$
    and the diagonal matrix $D$.
 1: for $r = 0, \ldots, R-1$ do                                   ▷ performing restarts
 2:     $\tilde{x}_0 = z = x_r$.
 3:     for $s = 0, \ldots, S-1$ do
 4:         Compute in parallel and store $\nabla f(\tilde{x}_s)$.
 5:         while number of samples $\leq m$ do in parallel
 6:             Sample $i$ uniformly in $[n]$ and let $T_i$ be the support of $f_i$.
 7:             $[\hat{z}]_{T_i}$ = inconsistent read of $z$ on $T_i$.
 8:             $[\hat{y}]_{T_i} = \vartheta \cdot [\hat{z}]_{T_i} + (1-\vartheta) \cdot [\tilde{x}_s]_{T_i} - \varphi \cdot [D \nabla f(\tilde{x}_s)]_{T_i}$.
 9:             $[u]_{T_i} = -\eta \cdot \big(\nabla f_i([\hat{y}]_{T_i}) - \nabla f_i([\tilde{x}_s]_{T_i}) + D_i \nabla f(\tilde{x}_s)\big)$.
10:             for $v \in T_i$ do
11:                 $[z]_v = [z]_v + [u]_v$.                      ▷ coordinate-wise atomic write
12:             end for
13:         end while
14:         $\tilde{x}_{s+1} = \hat{y}_t$, where $\hat{y}_t$ is chosen uniformly at random among the inconsistent
    $\hat{y}$ in the previous epoch.
15:     end for
16:     $x_{r+1} = \frac{1}{S} \sum_{s=0}^{S-1} \tilde{x}_{s+1}$.
17: end for
Output: $x_R$.
```

4.2 Asynchronous Sparse Accelerated SVRG

We then extend Algorithm 8 into the asynchronous setting (Algorithm 9), and analyze it under the perturbed iterate framework proposed in [98]. Note that Algorithm 9 degenerates into Algorithm 8 if there is only one thread (or worker).

Perturbed iterate analysis. Let us denote the kth update as $\mathcal{G}_{\hat{y}_k} = \nabla f_{i_k}(\hat{y}_k) - \nabla f_{i_k}(\tilde{x}_s) + D_{i_k} \nabla f(\tilde{x}_s)$. The precise ordering of the parallel updates will be defined in the next paragraph. Mania et al. [98] proposed to analyze the following virtual iterates:

$$z_{k+1} = z_k - \eta \cdot \mathcal{G}_{\hat{y}_k}, \text{ for } k = 0, \ldots, m-1. \tag{4.4}$$

They are called the virtual iterates because except for z_0 and z_m, other iterates may not exist in the shared memory due to the lock-free design. Then, the inconsistent read $\hat{z}_k$ is interpreted as a perturbed version of z_k, which will be formalized shortly. Note that z_m is precisely z in the shared memory after all the one-epoch updates are completed due to the atomic write requirement, which is critical in our analysis.

Ordering the iterates. An important issue of the analysis in asynchrony is how to correctly order the updates that happen in parallel. Recht et al. [125] increases the counter k after each successful write of the update to the shared memory, and this ordering has been used in many follow-up works. Note that under this ordering, the iterates z_k at (4.4) exist in the shared memory. However, this ordering is incompatible with the unbiasedness assumption (Assumption 1 below), that is, enforcing the unbiasedness would require some additional overly strong assumption. Mania et al. [98] addressed this issue by increasing the counter k just before each inconsistent read of z. In this case, the unbiasedness can be simply enforced by reading all the coordinates of z (not just those on the support). Although this is expensive and is not used in their implementation, the unbiasedness is enforceable under this ordering, which is thus more reasonable. Leblond et al. [83] further refined and simplified their analysis by proposing to increase k after each inconsistent read of z is completed. This modification removes the dependence of $\hat{z}_k$ on "future" updates, i.e., on i_r for $r > k$, which significantly simplifies the analysis and leads to better speed-up conditions. See [83] for more detailed discussion on this issue. We follow the ordering of [83] to analyze Algorithm 9. Given this ordering, the value of $\hat{z}_k$ can be explicitly described as

$$[\hat{z}_k]_v = [z_0]_v - \eta \sum_{\substack{j \in \{0,\ldots,k-1\} \\ \text{s.t. coordinate } v \text{ was} \\ \text{written for } j \text{ before } k}} [\mathcal{G}_{\hat{y}_j}]_v.$$

Since $\hat{y}$ is basically composing $\hat{z}$ with constant vectors, it can also be ordered as

$$\hat{y}_k = \vartheta \hat{z}_k + (1 - \vartheta)\, \tilde{x}_s - \varphi D \nabla f(\tilde{x}_s). \tag{4.5}$$

Assumptions. The analysis in this line of work crucially relies on the following two assumptions.

Assumption 1 (unbiasedness). *$\hat{z}_k$ is independent of the sample i_k. Thus, we have $\mathbb{E}\big[\mathcal{G}_{\hat{y}_k} | \hat{z}_k\big] = \nabla f(\hat{y}_k)$.*

As we mentioned before, the unbiasedness can be enforced by reading all the coordinates of z while in practice one would only read those necessary coordinates.

This inconsistency exists in all the follow-up works that use the revised ordering [98, 83, 120, 158, 65, 51], and it is currently unknown how to avoid such an issue. Another inconsistency in Algorithm 9 is that in the implementation, when a $\hat{y}_k$ is selected as the next snapshot $\tilde{x}_{s+1}$, all the coordinates of $\hat{z}_k$ are loaded. This makes the choice of $\tilde{x}_{s+1}$ not necessarily uniformly random. KroMagnon (the improved version in [83]) also has this issue. In practice, no noticeable negative impact is observed for the two inconsistencies.

Assumption 2 (bounded overlaps)**.** *There exists a uniform bound τ on the maximum number of iterations that can overlap together.*

This is a common assumption in the analysis with stale gradients. Under this assumption, the explicit effect of asynchrony can be modeled as:

$$\hat{z}_k = z_k + \eta \sum_{j=(k-\tau)_+}^{k-1} J_j^k \mathcal{G}_{\hat{y}_j}, \tag{4.6}$$

where J_j^k is a diagonal matrix with its elements in $\{0,1\}$. The 1 elements indicate that $\hat{z}_k$ lacks some "past" updates on those coordinates.

Defining the same sparsity measure as in [125]: $\Delta = \frac{1}{n} \cdot \max_{v\in[d]} |\{i : v \in T_i\}|$, we are now ready to establish the convergence result of Algorithm 9. We first present the following guarantee for any τ given that some upper estimation of τ is available. The proof is provided in Appendix C.3.

Theorem 12. *For some $\widetilde{\tau} \geq \tau$ and any constant $\omega > 1$, we choose $m = \Theta(n), \vartheta = \frac{\sqrt{m}}{\sqrt{\kappa(1+2\sqrt{\Delta}\widetilde{\tau})}+\sqrt{m}}, \varphi = \frac{1-\vartheta}{L}, \eta = \frac{(1-\vartheta)}{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}$ and $S = \left\lceil 2\omega\sqrt{\frac{\kappa}{m}(1+2\sqrt{\Delta}\widetilde{\tau})}\right\rceil$. For any accuracy $\epsilon > 0$, we restart $R = O\big(\log\frac{f(x_0)-f(x^\star)}{\epsilon}\big)$ rounds. In this case, Algorithm 9 outputs x_R satisfying $\mathbb{E}\big[f(x_R)\big] - f(x^\star) \leq \epsilon$ in oracle complexity*

$$O\left(\max\left\{n, \sqrt{\kappa n(1+2\sqrt{\Delta}\widetilde{\tau})}\right\}\log\frac{f(x_0)-f(x^\star)}{\epsilon}\right).$$

An interesting observation is that Theorem 12 establishes an $O(\sqrt{\kappa n \tau})$ dependence, which seems to conflict with the $\Omega(\tau\sqrt{\kappa})$ lower bound in the deterministic $n = 1$ case (by adapting Theorem 3.15 in [16] with delayed gradient). Certainly there is no contradiction. The subtlety is again the periodic synchronization structure of Algorithm 9. When $n = 1$, we have $\tau \leq m = \Theta(1)$ and Algorithm 9 is "almost synchronous". Based on this theorem, it is direct to identify the region of τ where a theoretical linear speed-up is achievable.

Corollary 12.1 (Speed-up condition)**.** *In Theorem 12, let* $\tau \leq O\left(\frac{1}{\sqrt{\Delta}} \max\left\{\frac{n}{\kappa}, 1\right\}\right)$*. Then, setting* $\widetilde{\tau} = O\left(\frac{1}{\sqrt{\Delta}} \max\left\{\frac{n}{\kappa}, 1\right\}\right)$*, we have* $S = O\left(\max\left\{1, \sqrt{\frac{\kappa}{n}}\right\}\right)$*, which leads to the total complexity #grad* $= R \cdot S \cdot (n + 2m) = O\Big(\max\{n, \sqrt{\kappa n}\} \log \frac{f(x_0) - f(x^\star)}{\epsilon}\Big)$*.*

Proof. Note that $\widetilde{\tau} = O\big(\frac{1}{\sqrt{\Delta}} \max\left\{\frac{n}{\kappa}, 1\right\}\big)$ implies that $1 + 2\sqrt{\Delta}\widetilde{\tau} = O\left(\max\left\{\frac{n}{\kappa}, 1\right\}\right)$. In this case, it holds that $S = O\left(\max\left\{1, \sqrt{\frac{\kappa}{n}}\right\}\right)$, and then the total oracle complexity of Algorithm 9 is #grad $= R \cdot S \cdot (n + 2m) = O\Big(\max\{n, \sqrt{\kappa n}\} \log \frac{f(x_0) - f(x^\star)}{\epsilon}\Big)$. □

Note that the construction of Algorithm 9 naturally enforces that $\tau \leq m$. Hence, the precise linear speed-up condition of AS-Acc-SVRG is that $\tau = O(n)$ and $\tau = O(\frac{1}{\sqrt{\Delta}} \max\left\{\frac{n}{\kappa}, 1\right\})$, which is identical to that of ASAGA (cf., Corollary 9 in [83]) and slightly better than that of KroMagnon (cf., Corollary 18 in [83]).

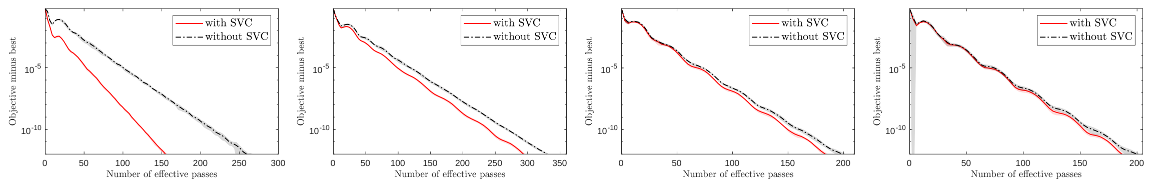
4.2.1 Some Insights about the Asynchronous Acceleration

Let us first consider the serial case (Algorithm 8). Observe that in one epoch, the iterate y_k is basically composing z_k with constant vectors, we can equivalently write the update (Step 8) as $y_{k+1} = y_k - \eta\vartheta \cdot \mathcal{G}_{y_k}$. Thus, the inner loop of Algorithm 8 is identical to that of (sparse) SVRG. The difference is that at the end of each epoch, when the snapshot $\tilde{x}$ is changed, an offset (or momentum) is added to the iterate. This has been similarly observed for accelerated SVRG in [159]. Note that since the sequence z appears in the potential function, the current formulation of Algorithm 8 allows a cleaner analysis. Then, in the asynchronous case, the inner loop of Algorithm 9 can also be equivalently written as the updates of asynchronous SVRG, and the momentum is added at the end of each epoch. This gives us some insights about the identical speed-up condition in Corollary 12.1: Since the asynchronous perturbation only affects the inner loop, the momentum is almost uncorrupted, unlike the cases of noisy gradient oracle. That is, the asynchrony only corrupts the "non-accelerated part" of Algorithm 9, which thus leads to the same speed-up condition as the non-accelerated methods.

4.3 Experiments

We present numerical results for the proposed scheme on optimizing the ℓ_2-logistic regression problem:

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \log\big(1 + \exp(-b_i \langle a_i, x \rangle)\big) + \frac{\mu}{2} \|x\|^2, \tag{4.7}$$



(a) Synthetic (b) KDD2010.S (c) RCV1.train (d) News20

Figure 4.2: Ablation study for the practical effect of sparse variance correction (abbreviated as SVC in the legends). Run 10 seeds. Shaded bands indicate ±1 standard deviation.

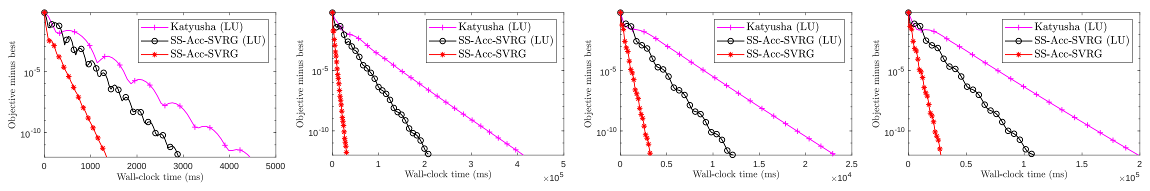
Table 4.1: Summary of the datasets. Density is the ratio of non-zero elements.

Scale	**Dataset**	n	d	μ	**Density**	Δ	**Description**
Small	Synthetic	100 000	100 000	10^{-7}	10^{-5}	10^{-5}	Identity data matrix with random labels
	KDD2010.S	70 000	29 890 095	10^{-7}	10^{-6}	0.15	The first 70 000 data samples of KDD2010
	RCV1.train	20 242	47 236	10^{-6}	$1.6 \cdot 10^{-3}$	0.42	
	News20	19 996	1 355 191	10^{-6}	$3.4 \cdot 10^{-4}$	0.93	
Large	KDD2010	19 264 097	29 890 095	10^{-10}	10^{-6}	0.16	
	RCV1.full	697 641	47 236	10^{-9}	$1.5 \cdot 10^{-3}$	0.43	Combined test and train sets of RCV1
	Avazu-site	23 567 843	999 962	10^{-10}	$1.5 \cdot 10^{-5}$	0.96	Avazu-site.train

where $a_i \in \mathbb{R}^d$, $b_i \in \{-1, +1\}$, $i \in [n]$ are the data samples and μ is the regularization parameter. We use the datasets from LIBSVM website [20], including KDD2010 [154], RCV1 [87], News20 [67], Avazu [66]. All the datasets are normalized to ensure a precise control on κ. We focus on the ill-conditioned case where $\kappa \gg n$ (the case where acceleration is effective). The dataset descriptions and the choices of μ are provided in Table 4.1. We conduct serial experiments on the small datasets and asynchronous experiments on the large ones. The asynchronous experiments were conducted on a multi-core HPC. Detailed setup can be found in Appendix C.7.

Before presenting the empirical results, let us discuss two subtleties in the implementation:

- If each $f_i(x) = \log\left(1 + \exp\left(-b_i \langle a_i, x\rangle\right)\right) + \frac{\mu}{2}\|x\|^2$, then it is supported on every coordinate due to the ℓ_2-regularization. Following [83], we sparsify the gradient of the regularization term as $\mu D_i x$; or equivalently, $f_i(x) = \log\left(1 + \exp\left(-b_i \langle a_i, x\rangle\right)\right) + \frac{\mu}{2}\langle x, D_i x\rangle$. Clearly, this f_i also sums up to the objective (4.7). The difference is that the Lipschitz constant of f_i will be larger, i.e., from $0.25 + \mu$ to at most



(a) Synthetic (b) KDD2010.S (c) RCV1.train (d) News20

Figure 4.3: Running time comparison between using sparse gradient estimator and lagged update (abbreviated as LU in the legends). The wall-clock time and objective value are averaged over 10 runs.

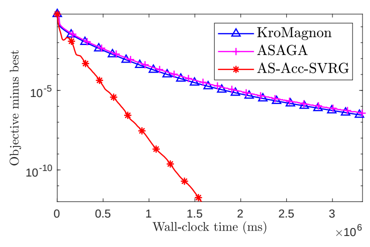
$0.25 + \mu n$. Since we focus on the ill-conditioned case ($L \gg \mu n$), this modification will not have significant effect.

- In Theorems 11 and 12, all the parameters have been chosen optimally in our analysis except for ω, which controls the restart frequency. Despite all the theoretical benefits of the restart framework mentioned in Section 4.1.2, in practice, we do not find performing restarts lead to a faster convergence. Thus, we do not perform restarts in our experiments (or equivalently, we choose a relatively large $\omega = 50$). Clearly, this choice will not make our method a "heuristic" since the theorems hold for any $\omega > 1$. Detailed discussion is given in Appendix C.4.

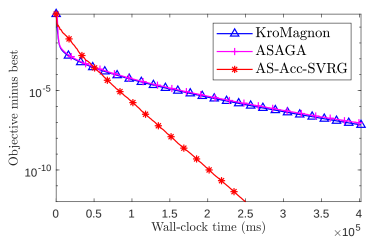
Additional experiments for verifying the $\sqrt{\kappa}$ dependence and a sanity check for our implementation is included in Appendices C.5 and C.6, respectively.

4.3.1 The Effectiveness of Sparse Variance Correction

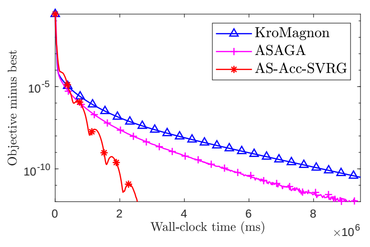
We first study the practical effect of the correction in the serial case. In Figure 4.2, "with SVC" refers to Algorithm 8 and "without SVC" is a naive extension of the dense version of Algorithm 8 into the sparse case (i.e., simply using a sparse estimator (4.2)), which suffers from the same sparsity issue as in [158]. For the two variants, we chose the same parameters in Theorem 11 to conduct an ablation study. In theory, we would expect the sparsity issue to be more severe if D is closer to nI_d (i.e., more sparse). Note that by definition, $\Delta = \max_{i\in[d]} D_{ii}^{-1}$, and thus $D \succeq \Delta^{-1} I_d$. Hence, smaller Δ indicates that D is closer to nI_d. From Figure 4.2d to 4.2a, Δ is decreasing and we observe more improvement from the correction. The improvement is consistent in our experiments, which justifies the effectiveness of sparse variance correction.



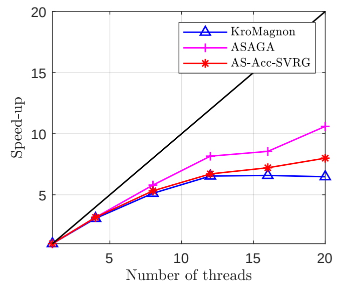
(a) KDD2010, 20 threads



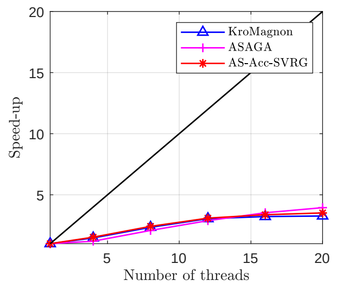
(b) RCV1.full, 20 threads



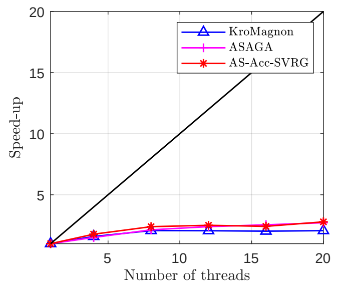
(c) Avazu-site, 20 threads



(d) KDD2010, speed-up



(e) RCV1.full, speed-up



(f) Avazu-site, speed-up

Figure 4.4: Convergence and speed-up for asynchronous sparse methods. Speed-up is the improvement on the wall-clock time to achieve 10^{-5} sub-optimality relative to using a single thread.

4.3.2 Sparse Estimator v.s Lagged Update

We then examine the running time improvement from adopting the sparse estimator compared with using the lagged update technique. Lagged update technique handles sparsity by maintaining a last seen iteration for each coordinate. When a coordinate is involved in the current iteration, it computes an accumulated update from the last seen iteration in closed form. Such computation is dropped when adopting a sparse estimator, which is the source of running time improvement. Moreover, it is extremely difficult to extend the lagged update technique into the asynchronous setting as discussed in Appendix E in [83]. In Figure 4.3, we compare SS-Acc-SVRG (Algorithm 8) with lagged update implementations of (dense) SS-Acc-SVRG and Katyusha. Their default parameters were used. Note that the lagged update technique is much trickier to implement, especially for Katyusha. We need to derive the closed-form solution of some complicated constant recursive sequence for the accumulated update. Plots with respect to effective passes are provided in Appendix C.5, in which SS-Acc-SVRG and Katyusha show similar performance.

4.3.3 Asynchronous Experiments

We compare the practical convergence and speed-up of AS-Acc-SVRG (Algorithm 9) with KroMagnon and ASAGA in Figure 4.4. We do not compare with the empirical method MiG in [158], which requires us to tune two highly correlated parameters with only limited insights. This is expensive or even prohibited for large scale tasks. For the compared methods, we only tune the τ-related constants in their theoretical parameter settings. That is, we tune the constant $1 + 2\sqrt{\Delta}\tilde{\tau}$ in Theorem 12 for AS-Acc-SVRG and the constant c in the step size $\frac{1}{cL}$ for KroMagnon and ASAGA. In fact, in all the experiments, we simply fixed the constant to 1 for AS-Acc-SVRG (the same parameters as the serial case), which worked smoothly. The main tuning effort was devoted to KroMagnon and ASAGA, and we tried to choose their step sizes as large as possible. The detailed choices can be found in Appendix C.7. Due to the scale of the problems, we only conducted a single run. From Figure 4.4, we see significant improvement of AS-Acc-SVRG for ill-conditioned tasks and similar practical speed-ups among the three methods, which verifies Theorem 12 and Corollary 12.1. We also observe a strong correlation between the practical speed-up and Δ, which is predicted by the theoretical $O(1/\sqrt{\Delta})$ dependence. It seems that we cannot reproduce the speed-up results in [83] on the RCV1.full dataset. This could be due to the difference in the programming languages, as we used C++ and they used Scalar. Pedregosa et al. [120] also used C++ implementation and we observe a similar $10\times$ speed-up of ASAGA on the KDD2010 dataset.

4.4 Chapter Summary

In this chapter, we proposed a new asynchronous accelerated SVRG method which achieves the optimal oracle complexity under the perturbed iterate framework. We show that it requires the same linear speed-up condition as the non-accelerated methods. Empirical results justified our findings. The limitations of our algorithm are that it requires a known μ and it does not support proximal operators. Directly incorporating the sparse proximal techniques in [120] results in an accelerated method that requires the knowledge of $\nabla f(x^\star)$.

Chapter 5

Neural Network Optimization: A Robust Nesterov's Momentum

Due to the non-smoothness and non-convexity of the loss function of neural networks, it is generally hard to design tailor-made optimizers for neural networks. The current strategy on training deep neural networks (DNN) in the community is to migrate the existing techniques in convex optimization to deep learning, which has achieved great success, e.g., momentum techniques [140], adaptive stepsize [39]. This chapter follows this trend: we propose a new type of (stochastic) Nesterov's momentum, prove that it enjoys the optimal convergence rate for convex problems, and then we conduct extensive experiments to evaluate its benefits in deep learning applications.

Nesterov's momentum [104, 107] is a widely used momentum technique in deep learning applications[1] (SGD with Nesterov's momentum):

$$\begin{aligned} y_{k+1} &= x_k - \eta \cdot \nabla f_{i_k}(x_k), \\ x_{k+1} &= y_{k+1} + \beta \cdot (y_{k+1} - y_k), \text{ for } k \geq 0, \end{aligned} \tag{5.1}$$

In this chapter, we propose a novel momentum technique called the *Amortized Nesterov's Momentum*. The key feature of this new momentum is that it achieves enhanced robustness[2] and acceleration at the same time, compared with plain SGD. Theoretically speaking, it has a parameter which trades Nesterov's acceleration (not convergence rate) for robustness. At one extreme, the proposed method is equivalent to AC-SA [78] and enjoys the optimal rate. At the other extreme, the method becomes mirror descent SA [103], which has a constant-factor better variance control

[1]See Section 5.4 for an introduction to the usage of Nesterov's momentum in deep learning.

[2]Here robustness refers to how well the method controls the variance of the stochastic noise, i.e., the variance term in the expected error and the probability of large deviations.

than AC-SA. It is important to note that our trade-off does not necessarily lead to a slower convergence rate and our technique has clear intuition.

The high-level idea is rather simple: stochastic Nesterov's momentum in existing Pytorch/Tensorflow implementations (see Section 5.4) can be unreliable since it is provided only by the previous iterate. The iterate potentially has large variance, which may lead to a false momentum that perturbs the training process. We thus propose to use the stochastic Nesterov's momentum based on several past iterates, which provides robust acceleration. In other words, instead of immediately using an iterate to provide momentum, we put the iterate into an "amortization plan" and use it later. This construction is also inspired by our acceleration tricks proposed in Chapter 2.

Another highlight is that we analyze the proposed methods in a general setting that covers smooth/non-smooth, deterministic/stochastic convex problems and allows choosing non-Euclidean norm for the problem space. Establishing the theoretic basis of our methods in this general setup extends the scope of our methods and benefits more applications.

This chapter is organized as follows: In Section 5.1, we specify the notations and general norm setup. In Section 5.2, we formally introduce the amortized momentum technique. In Section 5.3, we establish the convergence results under general norm setup. In Section 5.4, we extend the amortized momentum technique into the deep learning setting. In Section 5.5, we present extensive deep learning experiments to evaluate the new methods.

Practitioners from deep learning community can readily skip all the theoretic parts of this chapter and treat general norm setup as the standard Euclidean one.

5.1 Notations and General Norm Setup

For readers not familiar with the general space notions, the following setup can be regarded as the standard Euclidean one, i.e., $\|\cdot\|$ is the Euclidean norm and $\langle\cdot,\cdot\rangle$ is the inner product, and the composite function h can be treated as $h \equiv 0$.

Notations and Generalities. We use E to denote a finite-dimensional real vector space and E^* is its dual space. The value of a linear function $g \in E^*$ at $x \in E$ is represented by $\langle g, x\rangle$. $\|\cdot\|$ denotes an arbitrary norm in E and the dual norm $\|\cdot\|_*$ on E^* is defined in the standard way: $\|g\|_* \triangleq \max_{\|x\|=1} \langle g, x\rangle$. Scalar multiplication for $v \in E$ and $\beta \in \mathbb{R}$ is denoted as $\beta \cdot v$. The notation $[m]$ refers to the set $\{1, \ldots, m\}$ and the symbol $\leftarrow$ denotes assignment. We use $\mathbb{E}$ to denote expectation and the conditional expectation for a random process $i_0, i_1, \ldots$ is denoted as $\mathbb{E}_{i_k}[\,\cdot\,] \triangleq \mathbb{E}[\,\cdot \mid (i_0, \ldots, i_{k-1})]$.

Problem Setup. We consider the convex composite problem [12, 108]:

$$\min_{x \in X} \left\{ F(x) \triangleq f(x) + h(x) \right\},$$

where $X \subseteq E$ is a non-empty closed convex set and h is a proper convex function. We denote $x^\star \in X$ as a solution to this problem. $\nabla f(x) \in E^*$ represents (one of) the (sub)gradient of f at x. Given an input $x \in E$, the stochastic gradient oracle outputs an unbiased $\nabla f_i(x) \in E^*$, where the random variable i is independent of x.

We introduce the proximal setting, which generalizes the usual Euclidean setting. The *distance generating function* $d : X \to \mathbb{R}$ is required to be continuously differentiable and 1-strongly convex with respect to $\|\cdot\|$, i.e., $d(x) - d(y) - \langle \nabla d(y), x - y \rangle \geq \frac{1}{2}\|x - y\|^2, \forall x, y \in X$. The *prox-term* (*Bregman divergence*) associated with d is $V_d(x, y) \triangleq d(x) - d(y) - \langle \nabla d(y), x - y \rangle, \forall x, y \in X$. By adjusting $\|\cdot\|$ and $d(\cdot)$ to the geometry of the problem, mirror descent achieves a smaller problem-dependent constant than the Euclidean algorithms, which is its key benefit [102]. Typical proximal setups can be found in Section 5.3.3 in Ben-Tal and Nemirovski [14]. At a first reading, this setting can be taken as the standard Euclidean setting: $X = E = \mathbb{R}^n$, $\|\cdot\| = \|\cdot\|_2$, $\langle \cdot, \cdot \rangle$ is the inner product, $d(x) = \frac{1}{2}\|x\|_2^2$ and $V_d(x, y) = \frac{1}{2}\|x - y\|_2^2$.

We assume that V_d is chosen such that the *prox-mapping*,

$$\operatorname{prox}_h(x, \mathcal{G}) \triangleq \arg\min_{u \in X} \left\{ V_d(u, x) + \langle \mathcal{G}, u \rangle + h(u) \right\},$$

can be easily computed for any $x \in X, \mathcal{G} \in E^*$. Examples where this assumption is satisfied can be found in Parikh et al. [117], Ghadimi and Lan [46].

5.2 Amortized Nesterov's Momentum

In this section, we introduce SGD with Amortized Nesterov's Momentum (AM1-SGD) in Algorithm 10, and in Algorithm 11, we reformulate Algorithm 10 into a "momentum scheme" under the Euclidean setting with $h \equiv 0$ and constant momentum $\beta_s = \beta$. It can be verified that Algorithms 10 and 11 are equivalent through $\eta = \alpha_s(1 - \beta_s)$. This momentum scheme is related to how we implement AM1-SGD for deep learning applications and is also clearer for providing intuition. In Section 5.2.1, we propose another method (AM2-SGD) to implement the idea of utilizing several past iterates. To elaborate the features of AM1-SGD, we make the following remarks:

[3]For simplicity, we assume K is divisible by m.

Algorithm 10 AM1-SGD (Theoretic)

```
Input: Initial guess $x_0$, parameter $\{\alpha_s\}$, momentum $\{\beta_s\}$, amortization length $m$, iteration number $K$.
Initialize: $\tilde{x}_0 = z_0 = x_0, S = K/m$.[3]
  for $s = 0, \dots, S-1$ do
    for $j = 0, \dots, m-1$ do
      $k = sm + j$.
      $x_k = (1-\beta_s) \cdot z_k + \beta_s \cdot \tilde{x}_s$.
      $z_{k+1} = \text{prox}_{\alpha_s h}\left(z_k, \alpha_s \cdot \nabla f_{i_k}(x_k)\right)$.
    end for
    $\tilde{x}_{s+1} = \frac{1-\beta_s}{m} \cdot \sum_{j=1}^{m} z_{sm+j} + \beta_s \cdot \tilde{x}_s$.
  end for
Output: $\tilde{x}_S$.
```

Algorithm 11 AM1-SGD (Empirical)

```
Input: Initial guess $x_0$, learning rate $\eta$, momentum $\beta$, amortization length $m$, iteration number $K$.
Initialize: $x \leftarrow x_0$, $\tilde{x} \leftarrow x_0$, $\tilde{x}^+ \leftarrow \vec{0}$.
  for $k = 0, \dots, K-1$ do
    $x \leftarrow x - \eta \cdot \nabla f_{i_k}(x)$.
    $\tilde{x}^+ \leftarrow \tilde{x}^+ + \frac{1}{m} \cdot x$.
    if $(k+1) \mod m = 0$ then
      $x \leftarrow x + \underline{\beta \cdot (\tilde{x}^+ - \tilde{x})}$.
      $\tilde{x} \leftarrow \tilde{x}^+, \tilde{x}^+ \leftarrow \vec{0}$.
    end if
  end for
Output: Option I: $x$, Option II: $\tilde{x}$.
```

A periodical and large momentum. A graphical illustration of Algorithm 11 is included in Figure 5.1, which depicts how AM1-SGD leverages several past iterates to provide momentum. Nesterov's momentum is injected in every iteration. In comparison, the amortized momentum is injected every m iterations, while this momentum $\beta \cdot (\tilde{x}^+ - \tilde{x})$ is expected to be much larger than $\beta \cdot (y_{k+1} - y_k)$ if the same η and β are used. Intuitively, we can understand the amortized momentum as an m times larger Nesterov's momentum, which is applied every m iterations.

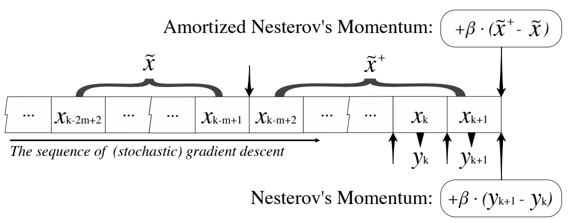

Amortized Nesterov's Momentum:
$+\beta \cdot (\tilde{x}^+ - \tilde{x})$
$\tilde{x}$
$\tilde{x}^+$
...
x_{k-2m+2}
...
...
x_{k-m+1}
x_{k-m+2}
...
...
x_k
x_{k+1}
The sequence of (stochastic) gradient descent
y_k
y_{k+1}
Nesterov's Momentum:
$+\beta \cdot (y_{k+1} - y_k)$

Figure 5.1: Graphical illustration of Amortized Nesterov's Momentum. This figure describes how the momentum is injected into the sequence of gradient descent $\{x_k\}$.

The bridge between accelerated schemes and mirror descent. It can be verified that if $m = 1$, Algorithm 11 is equivalent to (stochastic) Nesterov's scheme (5.1) and Algorithm 10 becomes AC-SA [78]; if $m = K$, Algorithm 11 is the SGD that outputs the average of the whole history and Algorithm 10 is equivalent to mirror descent SA [103, 78].

Acceleration and tail averaging. The main ingredients of AM1-SGD are Nesterov acceleration and tail averaging, namely, the output point $\tilde{x}$ is an m-iterations tail average and the amortized momentum is provided by two consecutive tail averages. It seems that the effects of outputting a tail average and applying the amortized momentum are independent. Option I in Algorithm 11, which we provide as a heuristic option, omits the tail averaging at the output point.

Options. Option II in Algorithm 11, which corresponds to the output of Algorithm 10, is the theoretical option that we analyze in Section 5.3. Option I, in addition to omitting the tail averaging effect, follows the implementations of Nesterov's momentum in PyTorch [118] and Tensorflow [1]. We will see in Section 5.4 that the standard Nesterov's momentum also has this type of heuristic and theoretical options.

Connections with BS-SVRG in Chapter 2. Our original inspiration of AM1-SGD comes from BS-SVRG (Algorithm 2) in Chapter 2, which uses a previously calculated "snapshot" point $\tilde{x}$ to provide momentum. AM1-SGD also uses an aggregated point to provide momentum and it shares many structural similarities with BS-SVRG.

5.2.1 AM2-SGD

We propose another realization of the amortization technique (AM2-SGD) in Algorithm 12, and similar to AM1-SGD, its "momentum scheme" reformulation in Algorithm 13. We were inspired by the constructions of SVRG [64] and SAGA [29], the most popular methods in finite-sum convex optimization—to reuse the information from several past iterates, we can either maintain a "snapshot" that aggregates the information or keep the iterates in a table.

We discuss some interesting characteristics of AM2-SGD by making the following remarks:

Identical iterations. The workload of AM1-SGD varies for different iterations due to the if-clause (or the two-loop structure). This to some extent limits its extensibility to other settings (e.g., asynchronous setting in Chapter 4). AM2-SGD does not have this issue and is structurally simpler. Although AM2-SGD requires storing a table

Algorithm 12 AM2-SGD (Theoretic)

```
Input: Initial $x_0$, amortization length $m$, point table $\phi = \begin{bmatrix}\phi_1 & \cdots & \phi_m\end{bmatrix} \in E^m$, parameter $\{\alpha_k\}$, momentum $\{\beta_k\}$, iteration number $K$.
Initialize: $z_0 = \phi_j^0 = x_0, \forall j \in [m]$.
  for $k = 0, \ldots, K-1$ do
      Sample $j_k$ uniformly in $[m]$.
      $x_k^{j_k} = (1-\beta_k) \cdot z_k + \beta_k \cdot \phi_{j_k}^k$.
      $z_{k+1} = \text{prox}_{\alpha_k h}\left(z_k, \alpha_k \cdot \nabla f_{i_k}(x_k^{j_k})\right)$.
      $\phi_{j_k}^{k+1} = (1-\beta_k) \cdot z_{k+1} + \beta_k \cdot \phi_{j_k}^k$ and keep other entries unchanged (i.e., $\phi_j^{k+1} = \phi_j^k$ for $j \neq j_k$).
  end for
Output: $\bar{\phi}^K = \frac{1}{m}\sum_{j=1}^m \phi_j^K$.
```

Algorithm 13 AM2-SGD (Empirical)

```
Input: Initial $x_0$, amortization length $m$, point table $\phi \in \mathbb{R}^{n \times m}$, learning rate $\eta$, momentum $\beta$, iteration number $K$.
Initialize: $\phi_j^0 = x_0, \forall j \in [m]$. $j_0$ is uniformly sampled in $[m]$. If Option II, store a running average $\bar{\phi}^0 = x_0$.
  for $k = 0, \ldots, K-1$ do
      $\phi_{j_k}^{k+1} = x_k - \eta \cdot \nabla f_{i_k}(x_k)$, keep other entries unchanged ($\phi_j^{k+1} = \phi_j^k$ for $j \neq j_k$).
      Sample $j_{k+1}$ uniformly in $[m]$.
      $x_{k+1} = \phi_{j_k}^{k+1} + \beta \cdot (\phi_{j_{k+1}}^{k+1} - \phi_{j_k}^k)$.
      if Option II then
          $\bar{\phi}^{k+1} = \bar{\phi}^k + \frac{1}{m} \cdot \left(\phi_{j_k}^{k+1} - \phi_{j_k}^k\right)$.
  end for
Output: Option I: $x_K$, Option II: $\bar{\phi}^K$.
```

of vectors, which could be expensive in practice, the table size m is tunable, and we will see that in theory, it is more beneficial to choose relatively small m.

"Random tail averaging". Based on the expectation of geometric distribution, we know that the point table ϕ is expected to store m iterates from the most recent $\Theta(m \log m)$ iterates. Thus, we can regard the output $\bar{\phi}$, the average of the point table, as a "random tail average". The momentum of AM2-SGD is randomly provided by two past iterates in the table. Interestingly, as shown in Section 5.5.2, when using the same (η, β, m), the convergence of AM2-SGD is similar to AM1-SGD while being slightly faster. This suggests that randomly incorporating past iterates beyond m iterations helps.

Options. As is the case for AM1-SGD, we provide Option I in Algorithm 13 following the same heuristics. However, in our preliminary experiments, we found that the performance of Option I is not stable, and thus we do not recommend this option for AM2-SGD. We believe that it is caused by the additional randomness $\{j_k\}$.

Connections with BS-SAGA in Chapter 2. Similar to AM1-SGD, the construction of AM2-SGD is inspired by BS-SAGA (Algorithm 15) in Chapter 2, and they share many structural similarities.

5.3 Convergence Results

In this section, we analyze the theoretic versions of AM1-SGD (Algorithm 10) and AM2-SGD (Algorithm 12) in the convex setting. Comparing Algorithms 10 and 12, we see that their iterations can be generalized as follows ($y^+ = x_{k+1}$ for AM1-SGD):

$$\begin{aligned} x &= (1-\beta)\cdot z + \beta \cdot y, \\ z^+ &= \operatorname{prox}_{\alpha h}\big(z, \alpha \cdot \nabla f_i(x)\big), \\ y^+ &= (1-\beta)\cdot z^+ + \beta \cdot y. \end{aligned} \tag{5.2}$$

This scheme is first proposed by Auslender and Teboulle [9], which represents one of the simplest variants of Nesterov's methods (see Tseng [148] for the others). This scheme is modified into various settings [59, 78, 46, 158, 159, 81] to achieve acceleration.

We impose the following assumptions on the regularity of f and ∇f_i, which are classical in the analysis of stochastic approximation algorithms (identical to the ones in Ghadimi and Lan [46] with $\mu = 0$):

Assumption 3. *For some $L \geq 0, M \geq 0, \sigma \geq 0$,*

(a) $0 \leq f(y) - f(x) - \langle \nabla f(x), y - x\rangle \leq \frac{L}{2}\left\|y-x\right\|^2 + M\left\|y-x\right\|, \forall x, y \in X.$[4]

(b) $\mathbb{E}_i\left[\nabla f_i(x)\right] = \nabla f(x), \forall x \in X.$

(c) $\mathbb{E}_i\big[\left\|\nabla f_i(x) - \nabla f(x)\right\|_*^2\big] \leq \sigma^2, \forall x \in X.$

These assumptions cover several important classes of convex problems. For example, *(a)* covers the cases of f being L-smooth ($M = 0$) or L_0-Lipschitz continuous ($M = 2L_0, L = 0$) convex functions and if $\sigma = 0$ in *(c)*, the assumptions cover several classes of deterministic convex programming problems.

The following lemma serves as a cornerstone for the convergence analysis of AM1-SGD and AM2-SGD. All the proofs in this section are given in Appendix D.

[4]When $M > 0$, f is not necessarily differentiable and $\nabla f(x)$ denotes an arbitrary sub-gradient [107] of f at x.

Lemma 4. *Let $\delta_x \triangleq \nabla f(x) - \nabla f_i(x)$. If $\alpha(1-\beta) < \frac{1}{L}$, the update scheme* (5.2) *satisfies the recursion:*

$$\frac{1}{1-\beta}\big(F(y^+) - F(x^\star)\big) + \frac{1}{\alpha}V_d(x^\star, z^+) \leq \frac{\beta}{1-\beta}\big(F(y) - F(x^\star)\big) + \frac{1}{\alpha}V_d(x^\star, z) + \frac{(\|\delta_x\|_* + M)^2}{2(\alpha^{-1} - L(1-\beta))} + \langle \delta_x, z - x^\star \rangle\,.$$

Based on this key recursion, we establish the convergence rates for AM1-SGD and AM2-SGD as follows.

Theorem 13. *In AM1-SGD, if $\beta_s = \frac{s}{s+2}$, $\alpha_s = \frac{\lambda_1}{L(1-\beta_s)}$ where*

$$\lambda_1 = \min\left\{\frac{2}{3}, \frac{L\sqrt{V_d(x^\star, x_0)}}{\sqrt{2m}\sqrt{\sigma^2 + M^2}(S+1)^{\frac{3}{2}}}\right\}.$$

Then,

(a) The output $\tilde{x}_S$ satisfies

$$\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) \leq \frac{6LmV_d(x^\star, x_0)}{(K+m)^2} + \frac{8\sqrt{2V_d(x^\star, x_0)}\sqrt{\sigma^2 + M^2}}{\sqrt{K+m}} \triangleq \mathcal{K}_0(m).$$

(b) If X is compact and the variance has a "light tail", i.e.,

$$\mathbb{E}_i\left[\exp\left\{\|\nabla f_i(x) - \nabla f(x)\|_*^2/\sigma^2\right\}\right] \leq \exp\{1\}, \forall x \in X,$$

denoting $D_X \triangleq \max_{x\in X}\|x - x^\star\|$, for any $\Lambda \geq 0$, we have

$$\begin{aligned}&\operatorname{Prob}\left\{\{F(\tilde{x}_S) - F(x^\star) \leq \mathcal{K}_0(m) + \mathcal{K}_1(m,\Lambda)\}\right\}\\ &\geq 1 - \big(\exp\{-\Lambda^2/3\} + \exp\{-\Lambda\}\big),\end{aligned}$$

where the deviation term $\mathcal{K}_1(m,\Lambda)$ is

$$\mathcal{K}_1(m,\Lambda) \triangleq \frac{4\sqrt{6}\Lambda\sigma\big(\sqrt{3V_d(x^\star, x_0)} + D_X\big)}{3\sqrt{K+m}}.$$

Remark (a): Theorem 13a gives the expected objective error, from which the trade-off of m is clear: Increasing m improves the dependence on variance σ but deteriorates the $O(L/K^2)$ term (i.e., the acceleration). Note that for AM1-SGD, m

is strictly constrained in $\{1, \ldots, K\}$. When $m = K$, AM1-SGD is equivalent to mirror descent SA, and the convergence rate in Theorem 13a becomes the corresponding $O(L/K + (\sigma + M)/\sqrt{K})$ (cf. Theorem 1 in Lan [78]). By taking derivative, we see that the minimum of the expected error $\mathcal{K}_0(m)$ is obtained at either $m = 1$ or $m = K$. This to some extent undermines the choices of setting $1 < m < K$. However, it is worth noting that in practice, the values $V_d(x^\star, x_0)$, σ, L and M could be unknown, especially $V_d(x^\star, x_0)$. In this case, these values are chosen as some upper estimations and can be very inaccurate. The parameter m allows users to determine the amount of acceleration and variance control for concrete tasks, which is much more flexible than sticking to $m = 1$ or $m = K$.

Remark (b): Theorem 13b provides the probability of the objective value deviating from its expected performance (i.e., $\mathcal{K}_0(m)$). It is clear that increasing m leads to smaller deviations with the same probability and thus improves the robustness of the iterates. The additional compactness and "light tail" assumptions are similarly required in Nemirovski et al. [103], Lan [78], Ghadimi and Lan [46]. Note that the "light tail" assumption is stronger than Assumption (c). Recently, Nazin et al. [101] established similar bounds without the "light tail" assumption by truncating the gradient. However, as indicated by the authors, their technique cannot be used for accelerated algorithms due to the accumulation of bias.

For AM2-SGD, we only give the expected convergence results as follows.

Theorem 14. *In AM2-SGD, if* $\beta_k = \frac{k/m}{k/m+2}$ *and* $\alpha_k = \frac{\lambda_2}{L(1-\beta_k)}$ *where*

$$\lambda_2 = \min\left\{\frac{2}{3}, \frac{L\sqrt{V_d(x^\star, x_0)}}{\sqrt{m}(\sigma + M)\left(\frac{K-1}{m} + 2\right)^{\frac{3}{2}}}\right\},$$

the output $\bar{\phi}^K$ *satisfies*

$$\begin{aligned}&\mathbb{E}\left[F(\bar{\phi}^K)\right] - F(x^\star)\\ &\leq \frac{4(m^2 - m)\big(F(x_0) - F(x^\star)\big) + 6LmV_d(x^\star, x_0)}{(K + 2m - 1)^2} + \frac{8\sqrt{V_d(x^\star, x_0)}(\sigma + M)}{\sqrt{K + 2m - 1}}.\end{aligned} \tag{5.3}$$

Remark: In comparison with Theorem 13a, Theorem 14 has an additional term $F(x_0) - F(x^\star)$ in the upper bound, which is inevitable. This difference comes from different restrictions on the choice of m. For AM2-SGD, $m \geq 1$ is the only requirement. Since it is impossible to let $m \gg K$ to obtain an improved rate, this additional term is inevitable. As a sanity check, we can let $m \to \infty$ to obtain a point table with almost all x_0, and then the upper bound becomes exactly $F(x_0) - F(x^\star)$. Since the first term in (5.3) increases rapidly with m, a smaller m is favored for AM2-SGD. In some cases, there exists an optimal choice of $m > 1$ in Theorem 14. However, the optimal choice could be messy and thus we omit the discussion here. Comparing the rates, we see that when using the same m, AM2-SGD has slightly better dependence on σ, which is related to the observation in Section 5.5.2 that AM2-SGD is always slightly faster than AM1-SGD.

If $m = O(1)$, Theorems 13 and 14 establish the optimal $O(L/K^2 + (\sigma + M)/\sqrt{K})$ rate in the convex setting (see Lan [78] for optimality), which verifies AM1-SGD and AM2-SGD as variants of Nesterov's method [104, 107]. We conducted convex experiments (in Appendix D.1.8) as sanity checkers for the theoretical results. Note that the improvements on stepsize policy proposed in Hu et al. [59] and Ghadimi and Lan [46] are orthogonal to the amortization technique and thus can be directly used in Theorems 13 and 14. From the above analysis, the effect of m can be understood as trading Nesterov's acceleration (the $O(1/K^2)$ term) for variance control (the $O(1/K)$ term). We expect amortization boosts the rate if σ or M is large, which is justified in Appendix D.1.8.

5.4 Amortized Momentum for Deep Learning

We start with reviewing the usage of Nesterov's momentum in deep learning. We discuss some subtleties in the implementation and evaluation, which contributes to the interpretation of our methods.

The standard SGD with Nesterov's momentum has the following scheme

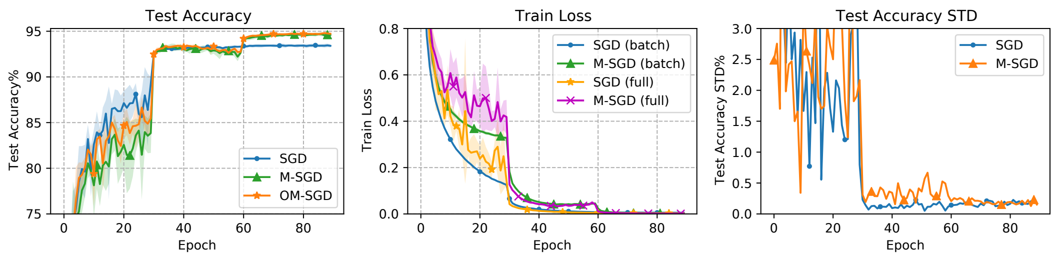
$$\begin{aligned} y_{k+1} &= x_k - \eta \cdot \nabla f_{i_k}(x_k), \\ x_{k+1} &= y_{k+1} + \beta \cdot (y_{k+1} - y_k), \text{ for } k \geq 0, \end{aligned} \tag{5.4}$$

which is widely used in deep learning. To make this point clear, recall that the reformulation in Sutskever et al. [140] (scheme (5.5), also the Tensorflow version) and the PyTorch version (scheme (5.6)) have the following schemes ($v, v^{pt} \in \mathbb{R}^n$ and $v_0 = v_0^{pt} = \vec{0}$): for $k \geq 0$,

$$\textit{TensorFlow:} \quad \begin{aligned} v_{k+1} &= \beta \cdot v_k - \eta \cdot \nabla f_{i_k}(y_k + \beta \cdot v_k), \\ y_{k+1} &= y_k + v_{k+1}. \end{aligned} \tag{5.5}$$

$$\textit{PyTorch:} \quad \begin{aligned} v_{k+1}^{pt} &= \beta \cdot v_k^{pt} + \nabla f_{i_k}(x_k), \\ x_{k+1} &= x_k - \eta \cdot (\beta \cdot v_{k+1}^{pt} + \nabla f_{i_k}(x_k)). \end{aligned} \tag{5.6}$$

Here the notations are modified based on their equivalence to scheme (5.4). It can be verified that schemes (5.5) and (5.6) are equivalent to (5.4) through $v_k = \beta^{-1} \cdot (x_k - y_k)$



(a) (b) (c)

Figure 5.2: ResNet34 on CIFAR-10. For all methods, initial learning rate $\eta_0 = 0.1$, momentum $\beta = 0.9$, run 5 seeds (start at same x_0). In (a) (b), we plot mean curves with shaded bands indicate ± 1 standard deviation. (c) shows the standard deviation of test accuracy and its average over 90 epochs.

and $v_k^{pt} = \eta^{-1}\beta^{-1} \cdot (y_k - x_k)$, respectively (see Defazio [28] for other equivalent forms of scheme (5.4)).

Interestingly, both PyTorch and Tensorflow[5] track $\{x_k\}$, which we refer to as **M-SGD**. This choice allows a consistent implementation when wrapped in a generic optimization layer [28]. The accelerated rate is built upon $\{y_k\}$ in Nesterov [107]. We use **OM-SGD** to refer to the Original M-SGD that outputs $\{y_k\}$.

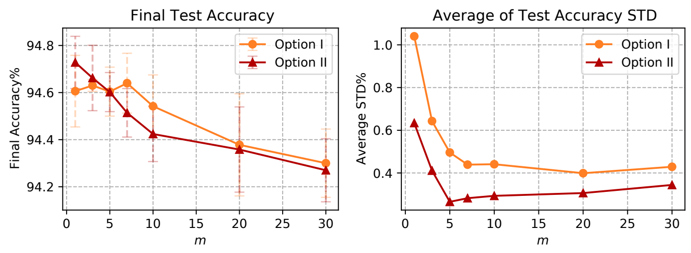
It can be verified that if $m = 1$, AM1-SGD (Algorithm 11) and AM2-SGD (Algorithm 13) with Option I are equivalent to M-SGD, and with Option II, they are equivalent to OM-SGD. By slightly modifying Algorithm 11, we can reduce its amortized iteration cost. We discuss this and other implementation details in Appendix D.1.1.

To introduce some evaluation metrics, we report the results of training ResNet34[6] [55] on CIFAR-10 [76] using SGD and M-SGD in Figure 5.2 and make the following remarks:

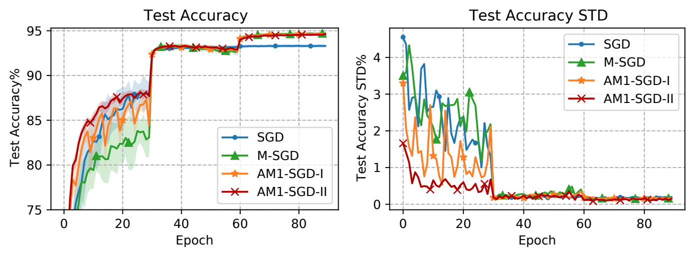
- *The role of SGD.* The performance of SGD is used as a reference in this section. Relating to Figure 5.1, we regard momentum as an add-on to plain SGD, and thus we choose the same learning rates for SGD and the momentum schemes. Such a perspective helps us understand what has been changed when applying momentum. Figure 5.2a shows that Nesterov's momentum hurts the convergence in the first 60 epochs but accelerates the final convergence, which verifies the importance of momentum for achieving high accuracy. Figure 5.2c sug-

[5]Tensorflow tracks the values $\{y_k + \beta \cdot v_k\} = \{x_k\}$.

[6]The settings follow Ma and Yarats [96]. Since 90-epoch training is not standard for CIFAR-10, we choose the models that can achieve decent performance in 90 epochs.



(a) Sweeping m in $\{3, 5, 7, 10, 20, 30\}$. Run 5 seeds.



(b) Fixing $m = 5$. Run 20 seeds.

Figure 5.3: ResNet34 on CIFAR-10. For all methods, $\eta_0 = 0.1, \beta = 0.9$, using same x_0. Labels of AM1-SGD are 'AM1-SGD-{*Option*}'. Shaded bands (or bars) indicate ± 1 standard deviation.

gests that adding Nesterov's momentum slightly increases the uncertainty in the training process of SGD.

- *Train-batch loss vs. Full-batch loss.* In Figure 5.2b, train-batch loss stands for the average of batch losses forwarded in an epoch, which is commonly used to indicate the training process in deep learning. Full-batch loss is the average loss over the entire training dataset evaluated at the end of each epoch. In terms of optimizer evaluation, full-batch loss is much more informative than train-batch loss as it reveals the robustness of an optimizer. However, full-batch loss is expensive to evaluate. On the other hand, test accuracy couples optimization and generalization, but since it is also evaluated at the end of the epoch, its convergence is similar to full-batch loss (see Figure 5.2a, 5.2b). Considering the basic usage of momentum in deep learning, we mainly use test accuracy to evaluate optimizers.

- *Robustness.* Inspired by Theorem 13b, we run a method multiple times with different seeds (same x_0) and measure the standard deviation of accuracy or loss at each iterate. Assuming a Gaussian underlying distribution, we can

Table 5.1: Detailed data of the curves in Figure 5.3b.

METHOD	**FINAL ACCURACY**	**Avg. STD**
SGD	$93.30\% \pm 0.20\%$	0.93%
M-SGD	$94.71\% \pm 0.17\%$	1.00%
AM1-SGD-I	$94.68\% \pm 0.18\%$	**0.59**%
AM1-SGD-II	$94.62\% \pm 0.15\%$	**0.31**%

characterize this deviation by $\mathcal{K}_1(m, \Lambda)$ in Theorem 13b with some fixed Λ.

We also plot the convergence of OM-SGD in Figure 5.2a. Interestingly, OM-SGD performs slightly better in this task: the final accuracies of M-SGD and OM-SGD are $94.61\% \pm 0.15\%$ and $94.73\% \pm 0.11\%$ with average deviations at 1.04% and 0.63%, respectively.

We do not compare with adaptive methods [39, 72], which scale the gradient using a diagonal matrix to speed up training. Wilson et al. [150] showed that these methods always generalize poorly compared with SGD with momentum. We chose the tasks where Nesterov's momentum is very effective and popular to conduct our experiments.

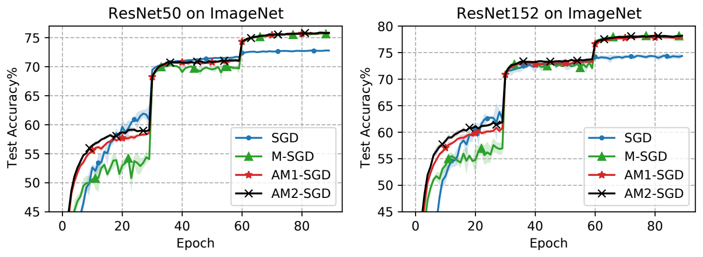
5.5 Experiments

From Theorems 13 and 14, we see that if m is small, the parameters α and β do not change a lot from the case where $m = 1$. This inspired us to align (η, β) of AM1/2-SGD with that of M-SGD and we tune only m. Such a choice facilitates the usage of AM1/2-SGD. In other words, we used the following parameter settings: η for SGD, (η, β) for M-SGD and (η, β, m) for AM1/2-SGD.

5.5.1 Parameter Sweep On CIFAR-10

As mentioned in Section 5.3, the practical effect of amortization is undetermined. Thus, we start with a parameter sweep experiment for m.

In Figure 5.3a, we trained ResNet34 on CIFAR-10 using AM1-SGD with various m. The experiments were repeated 5 times with different random seeds to measure the robustness. The convergence behaviors can be found in Appendix D.1.2. Note that the leftmost points ($m = 1$) in Figure 5.3a correspond to the results of M-SGD and OM-SGD, which are already given in Figure 5.2. From this empirical result, we see that m introduces a trade-off between the final accuracy and robustness while



METHOD	ImageNet (Final Accuracy)	
	ResNet50	ResNet152
SGD	72.78% ± 0.08%	74.36% ± 0.29%
M-SGD	75.71% ± 0.06%	78.07% ± 0.10%
AM1-SGD	**75.78**% ± 0.11%	77.82% ± 0.29%
AM2-SGD	**75.85**% ± 0.07%	**78.19**% ± 0.15%

Figure 5.4 & Table 5.2: ResNet on ImageNet. Run 3 seeds. Shaded bands indicate ± 1 standard deviation.

the improvement on the robustness is much more significant than the negative effect on the final accuracy. Figure 5.3a suggests that $m = 5$ is a good choice for this task. For simplicity, and also as a recommended setting, we fix $m = 5$ for the rest of experiments in this section.

To provide a stronger justification, we ran 20 seeds with $m = 5$ in Figure 5.3b and the detailed data are given in Table 5.1. Recall that Option I omits the tail averaging at the output point. We can thus understand the gap between two options as the effect of tail averaging. Since Option I is basically SGD with the amortized momentum, the results justify that the amortized momentum significantly increases the robustness. It is interesting that the amortized momentum, while being a very large momentum, not only provides acceleration, but also helps the algorithm become more robust than SGD. This observation basically differentiates AM1-SGD from a simple interpolation in-between M-SGD and SGD.

We measured all the wall-clock times in the experiments. However, we observed that even on the same type of GPUs, the running times fluctuate a lot and do not exhibit a clear trend. Roughly speaking, the running time of AM1-SGD ($m = 5$) is improved by $2\% - 3\%$ compared with M-SGD (measured on the same GPU and using the same random batches).

We also did a full-batch loss experiment using a smaller ResNet18 with pre-activation [54]. Since the results resemble Figure 5.3b, we report them in Appendix D.1.3.

Learning rate scheduler issue. We observed that when we use schedulers with a large decay factor and β is too large for the task (e.g., 0.995 for the task of this section), there would be a performance drop after the learning rate reduction. We believe that it is caused by the different cardinalities of iterates being averaged in $\tilde{x}^+$, which leads to a false momentum. This issue is resolved by restarting the algorithm after each learning rate reduction inspired by [115]. We include more discussion and evidence in Appendix D.1.6.

5.5.2 ImageNet

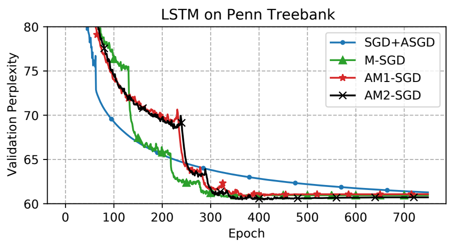
We trained ResNet50 and ResNet152 [55] on the ILSVRC2012 dataset ("ImageNet") [131] shown in Figure 5.4. Here we choose to evaluate Option II for AM1/2-SGD, which corresponds to the analysis. From the experiments on CIFAR-10, we see that Option I is basically a "perturbed" version of Option II, while this "perturbation" could lead to a slightly higher final accuracy (see Table 5.1).

For this task, we used 0.1 initial learning rate and 0.9 momentum for all methods, which is a typical choice. We performed a restart after each learning rate reduction as discussed in Appendix D.1.6. We believe that this helps the training process and also does not incur any additional overhead. We report the final accuracy in Table 5.2.

5.5.3 Language Model

We did a language model experiment on Penn Treebank dataset [99]. We used the LSTM [56] model defined in Merity et al. [100] and followed the experimental setup in its released code. We only changed the learning rate and momentum in the setup. The baseline is SGD+ASGD[7] [124] with constant learning rate 30 as used in Merity et al. [100]. For the choice of (η, β), following Lucas et al. [95], we chose $\beta = 0.99$ and used the scheduler that reduces the learning rate by half when the validation loss has not decreased for 15 epochs. We swept η from $\{5, 2.5, 1, 0.1, 0.01\}$ and found that $\eta = 2.5$ resulted in the lowest validation perplexity for M-SGD. We thus ran AM1-SGD and AM2-SGD (Option II) with this (η, β) and $m = 5$. Due to the small decay factor, we did not restart AM1-SGD and AM2-SGD after learning rate reductions. The convergence of validation perplexity is plotted in Figure 5.5. We report the lowest validation perplexity and test perplexity in Table 5.3. This experiment is directly comparable with the one in Lucas et al. [95].

[7]SGD+ASGD is to run SGD and switch to averaged SGD (ASGD) when a threshold is met.



METHOD	Penn Treebank (Perplexity)	
	Validation	Test
SGD+ASGD	61.28	59.07
M-SGD	60.75	58.36
AM1-SGD	60.73	**57.98**
AM2-SGD	**60.43**	58.23

Figure 5.5 & Table 5.3: LSTM on Penn Treebank.

5.6 Chapter Summary

In summary, we list the advantages of AM1-SGD over M-SGD (or from users' angle, the benefits of setting $m > 1$) that are discovered in this section:

(1) Increasing m improves robustness.

(2) Increasing m reduces the (amortized) iteration cost, which is discussed in Appendix D.1.1.

(3) A suitably chosen m boosts the convergence rate in the early stage of training and produces comparable final generalization performance.

(4) It is easy and safe to tune m. The performance of AM1-SGD are stable for a wide range of m.

Some minor drawbacks of AM1-SGD: it requires one more memory buffer, which is acceptable in most cases, and it shows some undesired behaviors when working with some learning rate schedulers, which can be addressed by performing restarts.

Extra results are provided in the appendices for interested readers: the robustness when using large β in Appendix D.1.4, a CIFAR-100 experiment in Appendix D.1.7 and comparison with classical momentum [123], AggMo [95] and QHM [96] in Appendix D.1.5.

Chapter 6

Neural Network Theory: Finding Approximately Stationary Points

In this chapter, we consider minimizing an L_0-Lipschitz continuous function:

$$\min_{x \in \mathbb{R}^d} f(x). \tag{6.1}$$

This problem is motivated by the non-smoothness and non-convexity of neural networks using rectified linear units (ReLU) activation. Since this is a rather general theoretic setting, we cannot hope for very strong convergence results that precisely capture the empirical behaviors of neural network optimizers. However, given that it is still unclear what property of neural networks can be used to analyze or construct algorithms [97, 13], studying this general problem serves as an importance initial step towards theoretically grounded neural network optimization.

6.1 Prior Arts

It is well-known that solving the general non-convex problem (6.1) is NP-hard. Without any additional problem structure, we can only hope for finding a stationary point of f (i.e., a point where the gradient is small). In f is smooth, we can find a stationary point $\|\nabla f(x)\| \leq \epsilon$ using GD in $O(\epsilon^{-2})$ gradient oracle calls [102], which is indeed the optimal complexity [17]. Then how about the non-smooth (L_0-Lipschitz) case? To answer this question, we need to first formally define the notion of gradient for potentially non-differentiable Lipschitz functions. Clarke subdifferential [21] is a widely used concept for generalized gradient, which is defined as

$$\partial f(x) \triangleq \mathrm{Co}\{s : \exists x' \to x, \nabla f(x') \text{ exists }, \nabla f(x') \to s\},$$

where $\mathrm{Co}\{\cdot\}$ is the convex hull of a set of vectors. Given this notion, the question now is to find a stationary point with $\mathrm{dist}(0, \partial f(x)) \le \epsilon$ in finite time, where $\mathrm{dist}(x, S) \triangleq \inf_{v \in S} \|v - x\|$. Unfortunately, Zhang et al. [155] showed that this goal is impossible for any first-order method. Naturally, one would consider a weaker notion of approximate stationarity, and hope that a finite-time complexity is possible in this case. A somewhat direct approximate stationarity notion is that given $\delta > 0$, $\mathrm{dist}(0, \cup_{y \in \mathbb{B}_\delta(x)} \partial f(y)) \le \epsilon$, where $\mathbb{B}_\delta(x) \triangleq \{v : \|v - x\| \le \delta\}$. This notion indicates that the generalized gradient at some point around x is small, which we call near-approximate stationarity (NAS). However, Kornowski and Shamir [74] showed that computing NAS in dimension-independent finite time is impossible.[1] Then what notion of approximate stationarity is computable in finite-time and dimension-independent complexity? Zhang et al. [155] first pointed out that it is possible for the notion of Goldstein approximate stationarity (GAS), i.e., $\mathrm{dist}(0, \partial_\delta f(x)) \le \epsilon$, where $\partial_\delta f(x) \triangleq \mathrm{Co}\{\cup_{y \in \mathbb{B}_\delta(x)} \partial f(y)\}$ is called the Goldstein δ-subdifferential [49]. Zhang et al. [155] proposed two algorithms to compute GAS in the deterministic and stochastic first-order settings. However, their methods are impractical as they rely on subgradient oracles that are highly nontrivial to compute.

6.2 Perturbed Stochastic INGD

In the work [146] that I coauthored, we resolved the impractical oracle issue of [155] and proposed the first practical first-order methods that are guaranteed to compute GAS with finite-time and dimension-independent complexities in deterministic and stochastic settings. Here I summarize my contribution in this work, which is the development of the stochastic method (Algorithm 14).

The stochastic first-order oracle is formally defined below, which follows the classic stochastic approximation oracle [103]. To make the dependence of random variables explicit, we use σ-field to denote conditional expectation.

Stochastic First-Order Oracle. $g_x = \mathbb{O}_\mathrm{s}(x)$: Given a point x, if f is differentiable at x, then the oracle $\mathbb{O}_\mathrm{s}(x)$ returns a stochastic gradient g_x with $\mathbb{E}\big[g_x \mid \sigma(x)\big] = \nabla f(x)$ where we assume that $\mathbb{E}\big[\, \|g_x - \nabla f(x)\|^2 \mid \sigma(x)\big] \le \sigma^2$. Since L_0-Lipschitz continuity implies $\|\nabla f(x)\| \le L_0$, we further have $\mathbb{E}\big[\, \|g_x\|^2 \mid \sigma(x)\big] \le L_0^2 + \sigma^2 \triangleq G^2$. Otherwise, the oracle sets `error` $= 1$ and terminates the program.

Given a point x, this oracle returns a stochastic gradient if f is differentiable

[1] As mentioned in the introduction, we would always hope for dimension-free methods in this big data era.

Algorithm 14 Perturbed Stochastic INGD

```
Input: $x_1 \in \mathbb{R}^d$ at which $f$ is differentiable.
Initialize: $m_1 = g_1 = \mathbb{O}_s(x_1)$. Set $\beta = 1 - \frac{\epsilon^2}{64G^2}$, $K = \frac{1}{\ln\frac{1}{\beta}} \ln\frac{16G}{\epsilon}$, $\omega = \left(\frac{1}{1-\beta} - \frac{1}{\ln\frac{1}{\beta}}\right)\ln\frac{16G}{\epsilon}$,
    $p = \frac{64G^2}{\delta\epsilon^2}\ln\frac{16G}{\epsilon}$, $q = \frac{256G^3}{\delta\epsilon^2}\ln\frac{16G}{\epsilon}$, $T = \frac{2^{16}G^3\Delta\ln\frac{16G}{\epsilon}}{\epsilon^4\delta}\max\{1, \frac{G\delta}{8\Delta}\}$. Set error $= 0$.
1: for $t \in [T]$ do
2:     $x_{t+1} = x_t - \eta_t m_t$, where $\eta_t = \frac{1}{p\|m_t\| + q}$.
3:     Sample $u_{t+1} \in \mathbb{R}^{d+1}$ uniformly from $\mathbb{S}^d$.
4:     Let $v_{t+1} \in \mathbb{R}^d$ be the first $d$ coordinates of $u_{t+1}$.
5:     If $\|m_t\| > 0, b_{t+1} = v_{t+1} - \frac{\langle v_{t+1}, x_t - x_{t+1}\rangle}{\|x_t - x_{t+1}\|^2}\cdot(x_t - x_{t+1})$; otherwise, $b_{t+1} = v_{t+1}$.
6:     Sample $y_{t+1}$ uniformly from $[x_t, x_{t+1} + \zeta b_{t+1}]$, where $\zeta = \min\{\frac{\omega}{p}, \frac{\epsilon^2}{510q(L_0+G)}\}$.
7:     Call oracle $g_{t+1} = \mathbb{O}_s(y_{t+1})$.
8:     $m_{t+1} = \beta m_t + (1-\beta) g_{t+1}$.
9: end for
Output: $x_{out} \triangleq x_{\max\{1, i-K\}}$, where $i \sim \text{Unif}([T])$.
```

at x, and reports error otherwise. Clearly, this oracle is practical. The key contribution of our work [146] is that we design a randomized mechanism (marked blue in Algorithm 14) to ensure that the algorithm will almost never query the oracle at a non-differentiable point. Without the randomized mechanism, Algorithm 14 is identical to Zhang et al.'s stochastic algorithm (cf., Algorithm 2 in [155]).

The intuition of this randomized mechanism is that we sample y_{t+1} from a probability space with non-zero measure. Then, using Rademacher theorem ([128], Theorem 9.60), we can guarantee that f is almost surely differentiable at y_{t+1}. See Lemma 3.2 in our work [146] for the formal proof of this mechanism.

6.3 Finite-Time & Dimension-Independent Guarantee

After resolving the practicality issue, we still need to carefully choose the perturbation radius ζ to ensure that the iterates are within the δ-ball of some reference point without hurting the convergence. Since m_t is a weighted average of all the stochastic gradient, we need to show that it approximately belongs to the Goldstein δ-subdifferential $\partial_\delta f(x)$ of some reference point x. Moreover, unlike in the deterministic case where we can terminate the algorithm if $\|m_t\|$ (note that $m_t \in \partial_\delta f(x_t)$) is small, in the stochastic case, m_t is a convex combination of stochastic gradients and thus it does not suffice to terminate the algorithm even if $\|m_t\| = 0$. The quantity that we aim to minimize is its expectation $\big\|\mathbb{E}\big[m_t\big]\big\| \leq \mathbb{E}\big[\,\|m_t\|\,\big]$. Due to this subtlety, we cannot let the perturbation size ζ_t adapt to $\|m_t\|$ as in the deterministic case (cf. Algorithm 1 in our work [146]): If $\zeta_t = \frac{\omega_1\|m_t\|}{p\|m_t\|+\omega_2}$ for some nonzero ω_1 and ω_2 in Algorithm 14, then when $\|m_t\| = 0$, we have $y_{t+1} = x_{t+1} = x_t$, and we cannot ensure that f is differentiable at x_t almost surely. We choose a constant $\zeta_t \equiv \zeta$ in Algorithm 14 instead. In this case, when $\|m_t\| = 0$, y_{t+1} is sampled from a ball centered at x_t.

After dealing with these technical subtleties, we formally establish the convergence guarantee of Algorithm 14 as follows. The proof is provided in Appendix E.1.

Theorem 15. *Let f be L_0-Lipschitz continuous. With probability at least $\frac{3}{5}$, the output of Algorithm 14 satisfies* $\operatorname{dist}\big(0, \partial_\delta f(x_{out})\big) \leq \epsilon$ *after at most*

$$\widetilde{O}\left(\frac{G^3\Delta}{\epsilon^4\delta}\right) \textit{ oracle calls}$$

and with $\mathbb{P}(\texttt{error} = 1) = 0$, *where* $f(x_0) - \inf_{x\in\mathbb{R}^d} f(x) \leq \Delta$ *and* $\widetilde{O}(\cdot)$ *hides logarithmic factors.*

About the Constant Probability. Currently we are not sure how to establish a high-probability result in the stochastic case.[2] We probably need to truncate the vector m_t in Algorithm 14 similar to the gradient clipping strategy in [101]. This modification is leave for future work.

About the Empirical Performance. We did some preliminary neural network experiments to evaluate Algorithm 14. However, the performance is not quite promising compared with the state-of-the-art neural network optimizers. This is in fact within our expectations. Our primary goal is to build theoretically grounded methods, but not ones with improved convergence rates. Due to this reason, the experiments are omitted here.

[2]Zhang et al. [155] claimed that their Algorithm 2 computes GAS in high probability by repeatedly restarting their Algorithm 2, which is wrong since we cannot determine the smallest $\operatorname{dist}\big(0, \partial_\delta f(x_{out})\big)$ after several restarts (this distance is in general not computable).

Chapter 7

Conclusion

In this thesis, we proposed several new algorithms that have the properties of fast, practical and scalable. Specifically,

- In Chapter 2, we discovered 4 theoretically fastest methods for various strongly convex finite-sum problems, whose derivation is based on a novel shifting theory inspired by the PEP methodology.
- In Chapter 3, we systematically studied the problem of finding near-stationary points in convex finite-sum problems, and derived 3 new practical schemes based on novel usages of the PEP methodology.
- Inspired by the development in Chapter 3, in Chapter 4, we discovered the first asynchronous lock-free algorithm with the optimal gradient complexity—a fast and scalable method which fully utilizes the parallel computing architectures.
- In Chapter 5, we focused on the challenging neural network optimization, and proposed 2 novel robust momentum methods, which are inspired by the acceleration tricks developed in Chapter 2.
- Finally, in Chapter 6, we built the first practical stochastic first-order method that is guaranteed to compute GAS of general neural networks with finite-time and dimension-independent complexity, thus making an important step towards theoretically grounded neural network optimization.

Here we list some potential future directions:

1. Currently, the PEP methodology only works for first-order methods and mostly under the smooth convex setting. It is natural to ask if we can extend PEP to

analyze or inspire second-order methods, and whether it is possible to consider weakly convex / quasiconvex / non-smooth settings.

2. There is still a gap in the complexities of finding near stationary points with stochastic gradients in Chapter 3. Is it possible to close the gap? That is, to find a stochastic method that has a convergence rate matching the lower bound, or to derive a better lower bound that matches the existing upper bounds.

3. How to extend the optimal asynchronous lock-free method in Chapter 4 to the proximal (or composite) setting [117]?

4. Can we analyze the convergence of AM1/2-SGD proposed in Chapter 5 in the non-convex case? Can we establish an enhanced robustness in theory?

5. Is the iteration complexity of perturbed stochastic INGD in Chapter 6 optimal? Can we establish a lower bound in the stochastic setting?

6. Is there any common property of general neural networks that can be leveraged for algorithm analysis or design?

Appendix A

Appendix for Chapter 2

A.1 Technical Lemmas with Proofs

Lemma 1 (Shifted mirror descent lemma)**.** *Given a gradient estimator $\mathcal{G}_y$, vectors $z^+, z^-, y \in \mathbb{R}^d$, fix the updating rule $z^+ = \arg\min_x \left\{ \langle \mathcal{G}_y, x \rangle + \frac{\alpha}{2} \|x - z^-\|^2 + \frac{\mu}{2} \|x - y\|^2 \right\}$. Suppose that we have a shifted gradient estimator $\mathcal{H}_y$ satisfying the relation $\mathcal{H}_y = \mathcal{G}_y - \mu(y - x^\star)$, it holds that*

$$\left\langle \mathcal{H}_y, z^- - x^\star \right\rangle = \frac{\alpha}{2} \left(\left\| z^- - x^\star \right\|^2 - \left(1 + \frac{\mu}{\alpha}\right)^2 \left\| z^+ - x^\star \right\|^2 \right) + \frac{1}{2\alpha} \left\| \mathcal{H}_y \right\|^2 .$$

Proof. Using the optimality condition,

$$\begin{gathered} \mathcal{G}_y + \alpha(z^+ - z^-) + \mu(z^+ - y) = \mathbf{0}, \\ \mathcal{H}_y + \alpha(z^+ - z^-) + \mu(z^+ - x^\star) = \mathbf{0}, \\ (\alpha + \mu)(z^+ - x^\star) = \alpha(z^- - x^\star) - \mathcal{H}_y, \\ (\alpha + \mu)^2 \left\| z^+ - x^\star \right\|^2 = \alpha^2 \left\| z^- - x^\star \right\|^2 - 2\alpha \left\langle \mathcal{H}_y, z^- - x^\star \right\rangle + \left\| \mathcal{H}_y \right\|^2 . \end{gathered}$$

Re-arranging the last equality completes the proof. □

Lemma 2 (Shifted firm non-expansiveness)**.** *Given relations $z^+ = \operatorname{prox}_i^\alpha(z^-)$ and $y^+ = \operatorname{prox}_i^\alpha(y^-)$, it holds that*

$$\frac{1}{\alpha^2} \left(1 + \frac{2(\alpha + \mu)}{L - \mu} \right) \left\| \nabla h_i(z^+) - \nabla h_i(y^+) \right\|^2 + \left(1 + \frac{\mu}{\alpha}\right)^2 \left\| z^+ - y^+ \right\|^2 \le \left\| z^- - y^- \right\|^2 .$$

Proof. Based on the first-order optimality condition and the definition of h_i,

$$\begin{gathered}\nabla f_i(z^+) + \alpha(z^+ - z^-) = \mathbf{0}, \qquad \nabla f_i(y^+) + \alpha(y^+ - y^-) = \mathbf{0},\\ \nabla h_i(z^+) + \nabla f_i(x^\star) + \mu(z^+ - x^\star) + \alpha(z^+ - z^-) = \mathbf{0},\\ \nabla h_i(y^+) + \nabla f_i(x^\star) + \mu(y^+ - x^\star) + \alpha(y^+ - y^-) = \mathbf{0}.\end{gathered}$$

Subtract the last two equalities,

$$(\alpha + \mu)(z^+ - y^+) = \alpha(z^- - y^-) - \big(\nabla h_i(z^+) - \nabla h_i(y^+)\big), \tag{A.1}$$

which implies

$$\begin{aligned}(\alpha + \mu)^2 \left\|z^+ - y^+\right\|^2 = \alpha^2 \left\|z^- - y^-\right\|^2 - 2\alpha \left\langle \nabla h_i(z^+) - \nabla h_i(y^+), z^- - y^- \right\rangle \\ + \left\|\nabla h_i(z^+) - \nabla h_i(y^+)\right\|^2 .\end{aligned} \tag{A.2}$$

Based on the interpolation condition of h_i, we have

$$\left\langle \nabla h_i(z^+) - \nabla h_i(y^+), z^+ - y^+ \right\rangle \geq \frac{1}{L - \mu} \left\|\nabla h_i(z^+) - \nabla h_i(y^+)\right\|^2 .$$

Together with (A.1), it holds that

$$\left\langle \nabla h_i(z^+) - \nabla h_i(y^+), z^- - y^- \right\rangle \geq \frac{1}{\alpha} \left(1 + \frac{\alpha + \mu}{L - \mu}\right) \left\|\nabla h_i(z^+) - \nabla h_i(y^+)\right\|^2 .$$

It remains to use this bound in (A.2). □

Forming convex combination between vector sequences is a common technique in designing accelerated methods (e.g., [9, 78, 46, 4]). From an analytical perspective, convex combination facilitates building a contraction between function values and the coefficient directly controls the contraction ratio, which is summarized in the following lemma. Unlike previous works, we allow a residual term $\mathcal{R}$ in the convex combination.

Lemma 3 (Function-value contraction)**.** *Given a continuously differentiable and convex function f, vectors $x^+, x^-, z, \mathcal{R} \in \mathbb{R}^d$ and scalar $\tau \in]0, 1[$, if $x^+ = \tau z + (1 - \tau)x^- + \mathcal{R}$, it satisfies that*

$$f(x^+) - f(x^\star) \leq (1 - \tau)\big(f(x^-) - f(x^\star)\big) + \left\langle \nabla f(x^+), \mathcal{R} \right\rangle + \tau \left\langle \nabla f(x^+), z - x^\star \right\rangle .$$

Proof. Using convexity twice,

$$\begin{aligned} f(x^+) - f(x^\star) &\le \left\langle \nabla f(x^+), x^+ - x^\star \right\rangle \\ &= \left\langle \nabla f(x^+), x^+ - z \right\rangle + \left\langle \nabla f(x^+), z - x^\star \right\rangle \\ &= \frac{1-\tau}{\tau} \left\langle \nabla f(x^+), x^- - x^+ \right\rangle + \frac{1}{\tau} \left\langle \nabla f(x^+), \mathcal{R} \right\rangle + \left\langle \nabla f(x^+), z - x^\star \right\rangle \\ &\le \frac{1-\tau}{\tau} \big(f(x^-) - f(x^+)\big) + \frac{1}{\tau} \left\langle \nabla f(x^+), \mathcal{R} \right\rangle + \left\langle \nabla f(x^+), z - x^\star \right\rangle . \end{aligned}$$

Re-arranging this inequality completes the proof. □

This simple trick (with $\mathcal{R} = \mathbf{0}$) appears frequently in the proofs of existing accelerated first-order methods. Note that the convexity arguments in this lemma can be strengthened by the interpolation condition or strong convexity if f satisfies additional assumptions.

A.2 Proofs of Section 2.3

A.2.1 Generality of the Framework of Algorithm 1

First, we show that TM is a parameterization of NAG (Algorithm 16 in Appendix A.6). Note that TM has the following scheme (the notations follow the ones in Cyrus et al. [24]):

$$\begin{aligned} x_{k+1} &= x_k + \beta(x_k - x_{k-1}) - \alpha \nabla f(y_k), \\ y_{k+1} &= x_{k+1} + \gamma(x_{k+1} - x_k), \\ z_{k+1} &= x_{k+1} + \delta(x_{k+1} - x_k). \end{aligned}$$

By casting this scheme into the framework of Algorithm 16, we obtain

$$\begin{aligned} y_k &= \frac{\gamma}{\delta} z_k + \left(1 - \frac{\gamma}{\delta}\right) x_k, \\ z_{k+1} &= \frac{\beta(1+\delta) - \gamma}{\delta - \gamma} z_k + \frac{\delta - \beta(1+\delta)}{\delta - \gamma} y_k - \alpha(1+\delta)\nabla f(y_k), \\ x_{k+1} &= \frac{1}{1+\delta} z_{k+1} + \frac{\delta}{1+\delta} x_k. \end{aligned}$$

Substituting the parameter choice of TM, we see that TM is equivalent to choosing $\alpha = \sqrt{L\mu} - \mu, \tau_y = (\sqrt{\kappa}+1)^{-1}, \tau_x = \frac{2\sqrt{\kappa}-1}{\kappa}$ in Algorithm 16. Interestingly, this choice and the choice of NAG (given in Appendix A.6) only differ in τ_x.

Then, we show that Algorithm 16 is an instance of the framework of Algorithm 1. By expanding the convex combinations of sequences $\{y_k\}$ and $\{x_k\}$ in Algorithm 16, we can conclude that

$$y_k = \tau_x z_k + (1-\tau_x)y_{k-1} + \tau_y(1-\tau_x)(z_k - z_{k-1}).$$

Based on the optimality condition at iteration $k-1$, we have

$$\alpha(z_k - z_{k-1}) = \mu(y_{k-1} - z_k) - \nabla f(y_{k-1}).$$

Now, it is clear that Algorithm 16 is an instance of the framework of Algorithm 1 with the variable-parameter choice (let $y_{-1} = x_0$): at $k = 0, \tau_0^x = \tau_y, \tau_0^z = 0$; at $k \geq 1, \tau_k^x = \tau_x, \tau_k^z = \frac{\tau_y(1-\tau_x)}{\alpha}$.

A.2.2 Proof of Theorem 1

First, we can introduce a contraction between $h(y_k)$ and $h(y_{k-1})$ using Lemma 3. Applying Lemma 3 with $f = h$ for the recursion $y_k = \tau_k^x z_k + (1-\tau_k^x)y_{k-1} + \tau_k^z\big(\mu(y_{k-1} - z_k) - \nabla f(y_{k-1})\big)$ and strengthening the convexity arguments by the interpolation condition, we obtain

$$\begin{aligned}
&h(y_k)\\
&\leq (1-\tau_k^x)h(y_{k-1}) + \tau_k^z \langle \nabla h(y_k), \mu(y_{k-1} - z_k) - \nabla f(y_{k-1})\rangle + \tau_k^x \langle \nabla h(y_k), z_k - x^\star\rangle\\
&\quad - \frac{\tau_k^x}{2(L-\mu)}\|\nabla h(y_k)\|^2 - \frac{1-\tau_k^x}{2(L-\mu)}\|\nabla h(y_{k-1}) - \nabla h(y_k)\|^2.
\end{aligned}$$

Note that $\mu(y_{k-1} - z_k) - \nabla f(y_{k-1}) = \mu(x^\star - z_k) - \nabla h(y_{k-1})$ by definition, and thus

$$\begin{aligned}
&h(y_k)\\
&\leq (1-\tau_k^x)h(y_{k-1}) - \tau_k^z \langle \nabla h(y_k), \nabla h(y_{k-1})\rangle + (\tau_k^x - \mu\tau_k^z)\langle \nabla h(y_k), z_k - x^\star\rangle\\
&\quad - \frac{\tau_k^x}{2(L-\mu)}\|\nabla h(y_k)\|^2 - \frac{1-\tau_k^x}{2(L-\mu)}\|\nabla h(y_{k-1}) - \nabla h(y_k)\|^2.
\end{aligned} \tag{A.3}$$

Then, to build a contraction between $\|z_{k+1} - x^\star\|^2$ and $\|z_k - x^\star\|^2$, we apply Lemma 1 with $\mathcal{G}_y = \nabla f(y_k), \mathcal{H}_y = \nabla h(y_k)$ and $z^+ = z_{k+1}$, which gives

$$\langle \nabla h(y_k), z_k - x^\star\rangle = \frac{\alpha_k}{2}\left(\|z_k - x^\star\|^2 - \left(1 + \frac{\mu}{\alpha_k}\right)^2 \|z_{k+1} - x^\star\|^2\right) + \frac{1}{2\alpha_k}\|\nabla h(y_k)\|^2.$$

Using this relation in (A.3), expanding and re-arranging the terms, we conclude that

$$\begin{aligned}
&h(y_k)-\left(\frac{\tau_k^x-\mu\tau_k^z}{2\alpha_k}-\frac{1}{2(L-\mu)}\right)\|\nabla h(y_k)\|^2+\frac{\alpha_k(\tau_k^x-\mu\tau_k^z)}{2}\left(1+\frac{\mu}{\alpha_k}\right)^2\|z_{k+1}-x^\star\|^2\\
&\leq(1-\tau_k^x)\left(h(y_{k-1})-\frac{1}{2(L-\mu)}\|\nabla h(y_{k-1})\|^2\right)+\frac{\alpha_k(\tau_k^x-\mu\tau_k^z)}{2}\|z_k-x^\star\|^2\\
&\quad+\left(\frac{1-\tau_k^x}{L-\mu}-\tau_k^z\right)\langle\nabla h(y_k),\nabla h(y_{k-1})\rangle\,.
\end{aligned}$$

It remains to impose parameter constraints according to the Lyapunov function.

A.2.3 Proof of Proposition 1.1

First, we can write the kth-update of G-TM with constant parameter as

$$\begin{aligned}
y_k&=(\tau_x-\tau_z\mu)z_k+\big(1-(\tau_x-\tau_z\mu)\big)y_{k-1}-\tau_z\nabla f(y_{k-1}),\\
z_{k+1}&=\frac{\alpha}{\alpha+\mu}z_k+\frac{\mu}{\alpha+\mu}y_k-\frac{1}{\alpha+\mu}\nabla f(y_k).
\end{aligned}$$

Substituting the constant parameter choice, we obtain

$$\begin{aligned}
y_k&=\frac{2}{\sqrt{\kappa}+1}z_k+\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\left(y_{k-1}-\frac{1}{L}\nabla f(y_{k-1})\right),\\
z_{k+1}&=\left(1-\frac{1}{\sqrt{\kappa}}\right)z_k+\frac{1}{\sqrt{\kappa}}y_k-\frac{1}{\sqrt{L\mu}}\nabla f(y_k).
\end{aligned}$$

For the objective function $f(x)=\frac{1}{2}\left\langle\begin{bmatrix}L&0\\0&\mu\end{bmatrix}x,x\right\rangle$, the update can be further expanded as

$$\begin{aligned}
y_k&=\frac{2}{\sqrt{\kappa}+1}z_k+\begin{bmatrix}0&0\\0&\frac{(\sqrt{\kappa}-1)^2}{\kappa}\end{bmatrix}y_{k-1},\\
z_{k+1}&=\left(1-\frac{1}{\sqrt{\kappa}}\right)z_k+\begin{bmatrix}-\frac{\kappa-1}{\sqrt{\kappa}}&0\\0&0\end{bmatrix}y_k.
\end{aligned}$$

Thus,

$$z_{k+1}=\left(1-\frac{1}{\sqrt{\kappa}}\right)\begin{bmatrix}-1&0\\0&1\end{bmatrix}z_k\implies\|z_{k+1}-x^\star\|^2=\left(1-\frac{1}{\sqrt{\kappa}}\right)^2\|z_k-x^\star\|^2\,,$$

as desired.

A.3 Proofs of Section 2.4

A.3.1 Proof of Theorem 2

For simplicity of presentation, we omit the superscript s for iterates in the same epoch.

Using the trick in Lemma 3 for the recursion

$$y_k = \tau_x z_k + (1-\tau_x)\tilde{x}_s + \tau_z\left(\mu(\tilde{x}_s - z_k) - \nabla f(\tilde{x}_s)\right)$$

and strengthening the convexity arguments by interpolation condition, we obtain

$$\begin{aligned} h(y_k) \le{}& \frac{1-\tau_x}{\tau_x}\langle \nabla h(y_k), \tilde{x}_s - y_k\rangle + \frac{\tau_z}{\tau_x}\langle \nabla h(y_k), \mu(\tilde{x}_s - z_k) - \nabla f(\tilde{x}_s)\rangle \\ &+ \langle \nabla h(y_k), z_k - x^\star\rangle - \frac{1}{2(L-\mu)}\|\nabla h(y_k)\|^2. \end{aligned}$$

Note that here the inner product $\langle \nabla h(y_k), \tilde{x}_s - y_k\rangle$ is not upper bounded as before. This term is preserved to deal with the variance.

By the definition of h, $\mu(\tilde{x}_s - z_k) - \nabla f(\tilde{x}_s) = \mu(x^\star - z_k) - \nabla h(\tilde{x}_s)$. Applying Lemma 1 with $\mathcal{H}_y = \mathcal{H}_{y_k}^{\text{SVRG}}, \mathcal{G}_y = \mathcal{G}_{y_k}^{\text{SVRG}}, z^+ = z_{k+1}$ and taking the expectation, we can conclude that

$$\begin{aligned} h(y_k) \le{}& \frac{1-\tau_x}{\tau_x}\langle \nabla h(y_k), \tilde{x}_s - y_k\rangle - \frac{\tau_z}{\tau_x}\langle \nabla h(y_k), \nabla h(\tilde{x}_s)\rangle - \frac{1}{2(L-\mu)}\|\nabla h(y_k)\|^2 \\ &+ \left(1 - \frac{\mu\tau_z}{\tau_x}\right)\frac{\alpha}{2}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\|z_{k+1} - x^\star\|^2\right]\right) \\ &+ \left(\frac{1}{2\alpha} - \frac{\mu\tau_z}{2\alpha\tau_x}\right)\mathbb{E}_{i_k}\left[\left\|\mathcal{H}_{y_k}^{\text{SVRG}}\right\|^2\right]. \end{aligned}$$

To bound the shifted moment, we apply the interpolation condition of h_{i_k}, i.e.,

$$\begin{aligned} \mathbb{E}_{i_k}\left[\left\|\mathcal{H}_{y_k}^{\text{SVRG}}\right\|^2\right] &= \mathbb{E}_{i_k}\left[\|\nabla h_{i_k}(y_k) - \nabla h_{i_k}(\tilde{x}_s)\|^2\right] + 2\langle \nabla h(y_k), \nabla h(\tilde{x}_s)\rangle - \|\nabla h(\tilde{x}_s)\|^2 \\ &\le 2(L-\mu)\big(h(\tilde{x}_s) - h(y_k) - \langle \nabla h(y_k), \tilde{x}_s - y_k\rangle\big) \\ &\quad + 2\langle \nabla h(y_k), \nabla h(\tilde{x}_s)\rangle - \|\nabla h(\tilde{x}_s)\|^2. \end{aligned}$$

After re-arranging the terms, we obtain

$$\begin{aligned} h(y_k) \leq\ & \left(1 - \frac{\mu\tau_z}{\tau_x}\right) \frac{L-\mu}{\alpha} \big(h(\tilde{x}_s) - h(y_k)\big) \\ & + \left[\frac{1-\tau_x}{\tau_x} - \left(1 - \frac{\mu\tau_z}{\tau_x}\right)\frac{L-\mu}{\alpha}\right] \langle \nabla h(y_k), \tilde{x}_s - y_k \rangle \\ & + \left(1 - \frac{\mu\tau_z}{\tau_x}\right)\frac{\alpha}{2}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\|z_{k+1} - x^\star\|^2\right]\right) \\ & + \left(\frac{1}{\alpha} - \frac{\mu\tau_z}{\alpha\tau_x} - \frac{\tau_z}{\tau_x}\right)\langle \nabla h(y_k), \nabla h(\tilde{x}_s)\rangle - \frac{1}{2(L-\mu)}\|\nabla h(y_k)\|^2 \\ & - \left(\frac{1}{2\alpha} - \frac{\mu\tau_z}{2\alpha\tau_x}\right)\|\nabla h(\tilde{x}_s)\|^2 . \end{aligned}$$

To cancel $\langle \nabla h(y_k), \tilde{x}_s - y_k\rangle$, we choose τ_z such that $\frac{1-\tau_x}{\tau_x} = \left(1 - \frac{\mu\tau_z}{\tau_x}\right)\frac{L-\mu}{\alpha}$, which gives

$$\begin{aligned} h(y_k) \leq\ & (1-\tau_x)h(\tilde{x}_s) + \frac{\alpha^2(1-\tau_x)}{2(L-\mu)}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\|z_{k+1}-x^\star\|^2\right]\right) \\ & + \frac{\alpha+\mu-(\alpha+L)\tau_x}{(L-\mu)\mu}\langle \nabla h(y_k), \nabla h(\tilde{x}_s)\rangle - \frac{\tau_x}{2(L-\mu)}\|\nabla h(y_k)\|^2 \\ & - \frac{1-\tau_x}{2(L-\mu)}\|\nabla h(\tilde{x}_s)\|^2 . \end{aligned} \tag{A.4}$$

In view of the Lyapunov function $T_s \triangleq h(\tilde{x}_s) - c_1\|\nabla h(\tilde{x}_s)\|^2 + \frac{\lambda}{2}\|z_0^s - x^\star\|^2$, there are two ways to deal with the inner product $\langle \nabla h(y_k), \nabla h(\tilde{x}_s)\rangle$:
Case I ($c_1 = 0$): Choosing τ_x such that $\alpha + \mu - (\alpha+L)\tau_x = 0 \implies \tau_x = \frac{\alpha+\mu}{\alpha+L}$ and dropping the negative gradient norms in (A.4), we arrive at (A.6) with $c_1 = 0$.
Case II ($c_1 \neq 0$): Denoting $\gamma = \frac{|\alpha+\mu-(\alpha+L)\tau_x|}{(L-\mu)\mu}$ and using Young's inequality for $\langle \nabla h(y_k), \nabla h(\tilde{x}_s)\rangle$ with parameter $\beta > 0$, we can bound (A.4) as

$$\begin{aligned} h(y_k) \leq\ & (1-\tau_x)h(\tilde{x}_s) + \frac{\alpha^2(1-\tau_x)}{2(L-\mu)}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\|z_{k+1}-x^\star\|^2\right]\right) \\ & + \left(\frac{\beta\gamma}{2} - \frac{\tau_x}{2(L-\mu)}\right)\|\nabla h(y_k)\|^2 - \left(\frac{1-\tau_x}{2(L-\mu)} - \frac{\gamma}{2\beta}\right)\|\nabla h(\tilde{x}_s)\|^2 . \end{aligned} \tag{A.5}$$

We require $\gamma \neq 0$ and choose $\beta > 0$ such that

$$\frac{\beta\gamma}{2} - \frac{\tau_x}{2(L-\mu)} = \frac{1}{1-\tau_x}\left(\frac{1-\tau_x}{2(L-\mu)} - \frac{\gamma}{2\beta}\right) = c_1 > 0.$$

It can be verified that this requirement and the existence of β are equivalent to the following constraints:

$$\begin{cases} \tau_x \neq \frac{\alpha+\mu}{\alpha+L}, \\ (1+\tau_x)^2(1-\tau_x) \geq 4\left(\left(\frac{\alpha}{\mu}+1\right) - \left(\frac{\alpha}{\mu}+\kappa\right)\tau_x\right)^2. \end{cases}$$

Under these constraints, denoting $\Delta = \frac{(1+\tau_x)^2}{(L-\mu)^2} - \frac{4\gamma^2}{1-\tau_x} \geq 0$, we can choose $\beta = \frac{1+\tau_x}{2\gamma(L-\mu)} + \frac{\sqrt{\Delta}}{2\gamma}$, which ensures $c_1 \in \left]0, \frac{1}{2(L-\mu)}\right[$.

Let $c_2 \triangleq \frac{\alpha^2(1-\tau_x)}{L-\mu}$. These two cases result in the same inequality:

$$\begin{aligned} h(y_k) - c_1 \left\|\nabla h(y_k)\right\|^2 &\leq (1-\tau_x)\big(h(\tilde{x}_s) - c_1 \left\|\nabla h(\tilde{x}_s)\right\|^2\big) \\ &\quad + \frac{c_2}{2}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\|z_{k+1}-x^\star\|^2\right]\right). \end{aligned} \tag{A.6}$$

Finally, summing the above inequality from $k=0,\dots,m-1$ with weight $\left(1+\frac{\mu}{\alpha}\right)^{2k}$, we conclude that

$$\begin{aligned} &\mathbb{E}\big[h(\tilde{x}_{s+1}) - c_1 \left\|\nabla h(\tilde{x}_{s+1})\right\|^2\big] \\ &= \sum_{k=0}^{m-1} \frac{1}{\widetilde{\omega}}\left(1+\frac{\mu}{\alpha}\right)^{2k} \mathbb{E}\big[h(y_k^s) - c_1 \left\|\nabla h(y_k^s)\right\|^2\big] \\ &\leq (1-\tau_x)\big(h(\tilde{x}_s) - c_1 \left\|\nabla h(\tilde{x}_s)\right\|^2\big) \\ &\quad + \frac{c_2}{2\widetilde{\omega}}\left(\|z_0^s - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^{2m} \mathbb{E}\big[\,\|z_m^s - x^\star\|^2\big]\right). \end{aligned} \tag{A.7}$$

Imposing the constraint $\left(1+\frac{\mu}{\alpha}\right)^{2m}(1-\tau_x) \leq 1$ completes the proof.

A.3.2 Proof of Proposition 2.1

The choice

$$\begin{cases} \alpha = \sqrt{cm\mu L} - \mu, \\ \tau_x = \left(1-\frac{1}{c\kappa}\right)\frac{\alpha+\mu}{\alpha+L} = \left(1-\frac{1}{c\kappa}\right)\frac{\sqrt{cm\kappa}}{\sqrt{cm\kappa}+\kappa-1}, \end{cases}$$

and the constraints

$$(1+\tau_x)^2(1-\tau_x) \geq 4\left(\left(\frac{\alpha}{\mu}+1\right) - \left(\frac{\alpha}{\mu}+\kappa\right)\tau_x\right)^2, \tag{A.8}$$

$$\left(1+\frac{\mu}{\alpha}\right)^{2m}(1-\tau_x) \leq 1, \tag{A.9}$$

are put here for reference.

Note that for $m \in \left(0, \frac{3}{4}\kappa\right]$, $\tau_x = \frac{c\kappa - 1}{c\kappa + \sqrt{\frac{c\kappa}{m}}(\kappa-1)}$ increases monotonically and $\frac{1+\tau_x}{m}$ decreases monotonically as m increases. Thus, for the constraint (A.8), letting

$$\phi(m,\kappa) \triangleq \frac{(1+\tau_x)^2(1-\tau_x)}{\left(\left(\frac{\alpha}{\mu}+1\right) - \left(\frac{\alpha}{\mu}+\kappa\right)\tau_x\right)^2} = \frac{1+\tau_x}{m}\left(1-\tau_x^2\right)c\kappa,$$

we have $\phi(m,\kappa)$ decreases monotonically as m increases.

When $m = \frac{3}{4}\kappa$, $\tau_x = \frac{c\kappa-1}{\left(c+\sqrt{\frac{4c}{3}}\right)\kappa - \sqrt{\frac{4c}{3}}}$. For $\kappa \geq 1$, if $c + \sqrt{\frac{4c}{3}} - c\sqrt{\frac{4c}{3}} \leq 0 \Leftrightarrow c \geq \frac{(\sqrt{3}+\sqrt{19})^2}{16} \approx 2.319$, we have τ_x decreases monotonically as κ increases. In this case, letting $\kappa \to \infty$, we conclude that $\tau_x > \frac{c}{c+\sqrt{\frac{4c}{3}}} > \frac{1}{3}$, which implies that $(1+\tau_x)^2(1-\tau_x)$ increases monotonically as τ_x decreases. Thus,

$$\phi(m,\kappa) \geq \phi\left(\frac{3}{4}\kappa, \kappa\right) \geq \phi\left(\frac{3}{4}, 1\right) = \frac{4}{3}\left(1+\frac{c-1}{c}\right)\left(1-\left(\frac{c-1}{c}\right)^2\right)c.$$

To meet the constraint (A.8), we require $c \geq 2+\sqrt{3} \approx 3.74$.

For constraint (A.9), defining

$$\psi(m,\kappa) \triangleq \left(\frac{\alpha+\mu}{\alpha}\right)^{2m}(1-\tau_x) = \left(1+\frac{1}{\sqrt{cm\kappa}-1}\right)^{2m}\frac{\sqrt{cm\kappa}+c\kappa(\kappa-1)}{(\sqrt{cm\kappa}-1+\kappa)c\kappa},$$

we have $\frac{\partial\psi}{\partial m} =$

$$\left(1+\frac{1}{\sqrt{cm\kappa}-1}\right)^{2m}\Bigg[\left(2\ln\left(1+\frac{1}{\sqrt{cm\kappa}-1}\right) - \frac{1}{\sqrt{cm\kappa}-1}\right)\frac{\sqrt{cm\kappa}+c\kappa(\kappa-1)}{(\sqrt{cm\kappa}-1+\kappa)c\kappa} - \frac{(\kappa-1)(c\kappa-1)}{2\sqrt{cm\kappa}\left(\sqrt{cm\kappa}-1+\kappa\right)^2}\Bigg].$$

Denote $q = \sqrt{cm\kappa} - 1 > 0$. The roots of $\frac{\partial\psi}{\partial m}$ are identified by the following equation:

$$s(q) \triangleq 2\ln\left(1+\frac{1}{q}\right) - \frac{1}{q} - \frac{b_0}{(q+1)(q+\kappa)(q+b_1)} = 0,$$

where $b_0 = \frac{c\kappa}{2}(\kappa-1)(c\kappa-1), b_1 = 1+c\kappa(\kappa-1)$. Taking derivative, we see that when $q \to 0$, $s'(q) \geq \frac{1}{q^2} - \frac{2}{q(1+q)} \to \infty$. We can arrange the equation $s'(q) = 0$ as finding the real roots of a polynomial. By Descartes' rule of signs, this equation has exactly one positive root (with $c \geq 2+\sqrt{3}$, we have $\kappa b_1 - 1 - b_0 \leq 0$ for any $\kappa \geq 1$ and then there is exactly one sign change in the polynomial). Thus, as q increases, $s(q)$ first increases monotonically to the unique root and then decreases monotonically.

To see that $s(q)$ has exactly one root, let $q \to 0, s(q) \leq 2\ln\left(1+\frac{1}{q}\right) - \frac{1}{q} \to -\infty$; when q is large enough (e.g., $q > 2$ and $(q+\kappa)(q+b_1) > 2b_0$), $s(q) > 0$; let $q \to \infty, s(q) \to 0$. These facts suggest that $s(q)$ has a unique root. Thus, we conclude that, as m increases, $\psi(m,\kappa)$ first decreases monotonically to the unique root and then increases monotonically, which means that for $m \in [2, \frac{3}{4}\kappa], \psi(m,\kappa) \leq \max\left\{\psi(2,\kappa), \psi\left(\frac{3}{4}\kappa,\kappa\right)\right\}$.

For $\psi(2,\kappa)$, $\psi'(2,\kappa) = \left(1+\frac{1}{\sqrt{2c\kappa}-1}\right)^4 \left(\sqrt{2c\kappa}+\kappa-1\right)^{-2}\left(\sqrt{2c\kappa}-1\right)^{-1}\ell(\kappa)$, where $\ell(\kappa)$ is a polynomial:

$$\ell(\kappa) \triangleq (c-2)\,\kappa - \frac{5\sqrt{2c}}{2}\kappa^{\frac{1}{2}} + (c+1) - \left(\sqrt{\frac{c}{2}} + \frac{1}{\sqrt{2c}}\right)\kappa^{-\frac{1}{2}} - 3\kappa^{-1} + \frac{3}{\sqrt{2c}}\kappa^{-\frac{3}{2}}.$$

It can be verified that with $c \geq 2+\sqrt{3}$, for any $\kappa \geq \frac{8}{3}, \ell'(\kappa) > 0$, which suggests that $\psi(2,\kappa) \leq \max\left\{\psi\left(2,\frac{8}{3}\right), \psi(2,\infty)\right\} \leq 1$ (with $c \geq 2+\sqrt{3}, \psi\left(2,\frac{8}{3}\right) \leq 0.953$ and $\psi(2,\infty) = 1$).

For $\psi\left(\frac{3}{4}\kappa,\kappa\right)$, $\psi'\left(\frac{3}{4}\kappa,\kappa\right) = \left(1+\frac{2}{\sqrt{3c\kappa}-2}\right)^{\frac{3}{2}\kappa}\left(\left(c+\sqrt{\frac{4c}{3}}\right)\kappa - \sqrt{\frac{4c}{3}}\right)^{-1}\omega_1(\kappa)$, where

$$\omega_1(\kappa)$$
$$\triangleq \left(\ln\left(1+\frac{2}{\sqrt{3c\kappa}-2}\right) - \frac{2}{\sqrt{3c\kappa}-2}\right)\left(\sqrt{3c\kappa}-\sqrt{3c}+\frac{3}{2}\right) + \frac{\sqrt{\frac{4c}{3}}c - c - \sqrt{\frac{4c}{3}}}{\left(c+\sqrt{\frac{4c}{3}}\right)\kappa - \sqrt{\frac{4c}{3}}}.$$

Let $p = \sqrt{3c\kappa}-2 > 0$, the roots of $\omega_1(\kappa)$ are determined by the equation

$$\omega_2(p) \triangleq \ln\left(1+\frac{2}{p}\right) - \frac{2}{p} + \frac{\frac{3}{2+\sqrt{3c}}\left(\sqrt{\frac{4c}{3}}c - c - \sqrt{\frac{4c}{3}}\right)}{\left(p+\frac{4}{2+\sqrt{3c}}\right)\left(p+\frac{7}{2}-\sqrt{3c}\right)} = 0.$$

To ensure that $\omega_2(p)$ increases monotonically as p increases, it suffices to set $c \leq 3.817$ (which ensures that $\omega_2'(p) > 0$). Thus, for any $p > 0$, $\omega_2(p) \leq \lim_{p\to\infty}\omega_2(p) = 0 \Rightarrow$ for any $\kappa \geq 1$, $\omega_1(\kappa) \leq 0$. Finally, we conclude that with $3.817 \geq c \geq 2+\sqrt{3}$, $\psi\left(\frac{3}{4}\kappa,\kappa\right) \leq \psi\left(2,\frac{8}{3}\right) \leq 0.953$, which completes the proof.

A.3.3 Proof of Proposition 2.2

The choice $\begin{cases} \alpha = \frac{3L}{2} - \mu, \\ \tau_x = \left(1 - \frac{1}{6m}\right)\frac{\alpha+\mu}{\alpha+L} = \left(1 - \frac{1}{6m}\right)\frac{3\kappa}{5\kappa-2}, \end{cases}$ is put here for reference.

We examine the constraint $(1+\tau_x)^2(1-\tau_x) \geq 4\left(\left(\frac{\alpha}{\mu}+1\right) - \left(\frac{\alpha}{\mu}+\kappa\right)\tau_x\right)^2$. Let

$$\phi(m,\kappa) \triangleq \frac{(1+\tau_x)^2(1-\tau_x)}{4\left(\left(\frac{\alpha}{\mu}+1\right) - \left(\frac{\alpha}{\mu}+\kappa\right)\tau_x\right)^2} = \frac{(1+\tau_x)^2(1-\tau_x)4m^2}{\kappa^2}.$$

For $m \geq \frac{3}{4}\kappa$, we have τ_x and $(1-\tau_x)m$ increases monotonically as m increases. Thus, $\phi(m,\kappa)$ increases as m increases $\Longrightarrow \phi(m,\kappa) \geq \phi(\frac{3}{4}\kappa,\kappa)$.

$\phi(\frac{3}{4}\kappa,\kappa) = \frac{9}{4}(1+\tau_x)^2(1-\tau_x)$ and $\tau_x = \frac{9\kappa-2}{15\kappa-6}$ in this case. Note that for $\kappa \geq 1$, τ_x decreases as κ increases and let $\kappa \to \infty$, we conclude that $\tau_x > \frac{3}{5} > \frac{1}{3} \Longrightarrow (1+\tau_x)^2(1-\tau_x)$ increases as τ_x decreases. Thus, $\phi(\frac{3}{4}\kappa,\kappa) \geq \phi(\frac{3}{4},1) > 1$, the constraint is satisfied.

Using this choice, we can write the per-epoch contraction (A.7) in Theorem 2 as

$$\begin{aligned}
&\mathbb{E}\big[h(\tilde{x}_{s+1}) - c_1\left\|\nabla h(\tilde{x}_{s+1})\right\|^2\big] + \frac{\alpha^2(1-\tau_x)}{2\widetilde{\omega}(L-\mu)}\left(1+\frac{\mu}{\alpha}\right)^{2m}\mathbb{E}\big[\left\|z_0^{s+1} - x^\star\right\|^2\big] \\
&\leq (1-\tau_x)\big(h(\tilde{x}_s) - c_1\left\|\nabla h(\tilde{x}_s)\right\|^2\big) + \frac{\alpha^2(1-\tau_x)}{2\widetilde{\omega}(L-\mu)}\left\|z_0^s - x^\star\right\|^2.
\end{aligned}$$

Note that for $\frac{m}{\kappa} > \frac{3}{4}$, $\tau_x > \frac{1}{2}$ and by Bernoulli's inequality, $\left(1+\frac{\mu}{\alpha}\right)^{2m} \geq 1 + \frac{2m\mu}{\alpha} = 1 + \frac{4m}{3\kappa-2} > 2$. Let $\lambda = \frac{2\alpha^2(1-\tau_x)}{\widetilde{\omega}(L-\mu)}$. The above contraction becomes

$$\begin{aligned}
&\mathbb{E}\big[h(\tilde{x}_{s+1}) - c_1\left\|\nabla h(\tilde{x}_{s+1})\right\|^2\big] + \frac{\lambda}{2}\mathbb{E}\big[\left\|z_0^{s+1} - x^\star\right\|^2\big] \\
&\leq \frac{1}{2}\cdot\left(h(\tilde{x}_s) - c_1\left\|\nabla h(\tilde{x}_s)\right\|^2 + \frac{\lambda}{2}\left\|z_0^s - x^\star\right\|^2\right).
\end{aligned}$$

Telescoping this inequality from $S-1$ to 0, we obtain $T_S \leq \frac{1}{2^S}T_0$, and since $m = 2n$, these imply an $O(n\log\frac{1}{\epsilon})$ iteration complexity.

Algorithm 15 SAGA Boosted by Shifting objective (BS-SAGA)

```
Input: Parameters $\alpha > 0, \tau_x \in ]0,1[$ and initial guess $x_0 \in \mathbb{R}^d$, iteration number $K$.
Initialize: $z_0 = x_0, \tau_z = \frac{\tau_x}{\mu} - \frac{\alpha(1-\tau_x)}{\mu(L-\mu)}$, a point table $\phi^0 \in \mathbb{R}^{d\times n}$ with $\forall i \in [n], \phi_i^0 = x_0$,
    running averages for the point table and its gradients.
1: for $k = 0, \ldots, K-1$ do
2:     Sample $i_k$ uniformly in $[n]$, set $\phi_{i_k}^{k+1} = \tau_x z_k + (1-\tau_x)\phi_{i_k}^k + \tau_z\big(\mu(\bar{\phi}^k - z_k) - \frac{1}{n}\sum_{i=1}^n \nabla f_i(\phi_i^k)\big)$ and keep other entries unchanged (i.e., for $i \neq i_k, \phi_i^{k+1} = \phi_i^k$).
3:     $z_{k+1} = \arg\min_x \left\{\left\langle \mathcal{G}^{\text{SAGA}}_{\phi_{i_k}^{k+1}}, x\right\rangle + (\alpha/2)\|x - z_k\|^2 + (\mu/2)\left\|x - \phi_{i_k}^{k+1}\right\|^2\right\}$.
4:     Update the running averages according to the change in $\phi^{k+1}$.
5: end for
Output: $z_K$.
```

A.3.4 BS-SAGA

To make the notations specific, we define

$$\mathcal{H}^{\text{SAGA}}_{x_k} \triangleq \nabla h_{i_k}(x_k) - \nabla h_{i_k}(\phi_{i_k}^k) + \frac{1}{n}\sum_{i=1}^n \nabla h_i(\phi_i^k)$$

$$\Rightarrow \mathcal{G}^{\text{SAGA}}_{x_k} \triangleq \nabla f_{i_k}(x_k) - \nabla f_{i_k}(\phi_{i_k}^k) + \frac{1}{n}\sum_{i=1}^n \nabla f_i(\phi_i^k) - \mu\left(\bar{\phi}^k - \phi_{i_k}^k\right),$$

where $\phi^k \in \mathbb{R}^{d\times n}$ is a point table that stores n previously chosen random anchor points and $\bar{\phi}^k \triangleq \frac{1}{n}\sum_{i=1}^n \phi_i^k$ denotes the average of point table.

The Lyapunov function (with $c_1 \in \left[0, \frac{1}{2(L-\mu)}\right], \lambda > 0$) is put here for reference:

$$T_k = \frac{1}{n}\sum_{i=1}^n h_i(\phi_i^k) - c_1\left\|\frac{1}{n}\sum_{i=1}^n \nabla h_i(\phi_i^k)\right\|^2 + \frac{\lambda}{2}\|z_k - x^\star\|^2. \tag{A.10}$$

We present the SAGA variant in Algorithm 15. In the following theorem, we only consider a simple case with $c_1 = 0$ in T_k. It is possible to analyze BS-SAGA with $c_1 \neq 0$ as is the case for BS-SVRG (the analysis in Appendix A.3.1). However, it leads to highly complicated parameter constraints. We provide a simple parameter choice similar to the one in Proposition 2.3.

Theorem A.3.1. *In Algorithm 15, if we choose α, τ_x as*

$$\begin{cases} \alpha \text{ is solved from the equation } \left(1 + \frac{\mu}{\alpha}\right)^2\left(1 - \frac{\alpha+\mu}{(\alpha+L)n}\right) = 1, \\ \tau_x = \frac{\alpha+\mu}{\alpha+L}, \end{cases} \tag{A.11}$$

the following per-iteration contraction holds for the Lyapunov function defined at (A.10) *(with $c_1 = 0$).*

$$\textit{With } \lambda = \frac{(1-\tau_x)\,(\alpha+\mu)^2}{(L-\mu)n}, \quad \mathbb{E}_{i_k}\left[T_{k+1}\right] \le \left(1+\frac{\mu}{\alpha}\right)^{-2} T_k, \ \textit{for } k \ge 0.$$

Regrading the rate, from (A.11), we can figure out that α is the unique positive root of the cubic equation:

$$\left(\frac{\alpha}{\mu}\right)^3 - (2n-3)\left(\frac{\alpha}{\mu}\right)^2 - (2n\kappa+n-3)\left(\frac{\alpha}{\mu}\right) - (n\kappa-1) = 0.$$

Using a similar argument as in Theorem 3, we can show that $\frac{\alpha}{\mu} = O(n+\sqrt{n\kappa})$, and thus conclude an $O\big((n+\sqrt{n\kappa})\log\frac{1}{\epsilon}\big)$ expected complexity for BS-SAGA. Interestingly, this rate is always slightly slower than that of BS-Point-SAGA.

Proof of Theorem A.3.1

To simplify the notations in this proof, we let $\Phi^k \triangleq \frac{1}{n}\sum_{i=1}^n h_i(\phi_i^k)$ and $\nabla\Phi^k \triangleq \frac{1}{n}\sum_{i=1}^n \nabla h_i(\phi_i^k)$.

Using the trick in Lemma 3 (with $f = h_{i_k}$) for $\phi_{i_k}^{k+1}$, strengthening the convexity with the interpolation condition and taking the expectation, we obtain

$$\begin{aligned}\mathbb{E}_{i_k}\left[h_{i_k}(\phi_{i_k}^{k+1})\right] \le & \frac{1-\tau_x}{\tau_x}\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), \phi_{i_k}^k - \phi_{i_k}^{k+1}\right\rangle\right] + \mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), z_k - x^\star\right\rangle\right] \\ & + \frac{\tau_z}{\tau_x}\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), \mu(\bar{\phi}^k - z_k) - \frac{1}{n}\sum_{i=1}^n \nabla f_i(\phi_i^k)\right\rangle\right] \\ & - \frac{1}{2(L-\mu)}\mathbb{E}_{i_k}\left[\left\|\nabla h_{i_k}(\phi_{i_k}^{k+1})\right\|^2\right].\end{aligned}$$

Note that by the definition of h_i, $\mu(\bar{\phi}^k - z_k) - \frac{1}{n}\sum_{i=1}^n \nabla f_i(\phi_i^k) = \mu(x^\star - z_k) - \nabla\Phi^k$, and thus

$$\begin{aligned}\mathbb{E}_{i_k}\left[h_{i_k}(\phi_{i_k}^{k+1})\right] \le & \frac{1-\tau_x}{\tau_x}\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), \phi_{i_k}^k - \phi_{i_k}^{k+1}\right\rangle\right] - \frac{\tau_z}{\tau_x}\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), \nabla\Phi^k\right\rangle\right] \\ & + \left(1 - \frac{\mu\tau_z}{\tau_x}\right)\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), z_k - x^\star\right\rangle\right] \\ & - \frac{1}{2(L-\mu)}\left\|\mathbb{E}_{i_k}\left[\nabla h_{i_k}(\phi_{i_k}^{k+1})\right]\right\|^2,\end{aligned} \tag{A.12}$$

which also uses Jensen's inequality, i.e., $\mathbb{E}_{i_k}\left[\left\|\nabla h_{i_k}(\phi_{i_k}^{k+1})\right\|^2\right] \geq \left\|\mathbb{E}_{i_k}\left[\nabla h_{i_k}(\phi_{i_k}^{k+1})\right]\right\|^2$.

Using Lemma 1 with $\mathcal{H}_y = \mathcal{H}^{\text{SAGA}}_{\phi_{i_k}^{k+1}}, \mathcal{G}_y = \mathcal{G}^{\text{SAGA}}_{\phi_{i_k}^{k+1}}, z^+ = z_{k+1}$ and taking the expectation, we obtain

$$\begin{aligned}
&\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), z_k - x^\star\right\rangle\right] \\
&= \frac{\alpha}{2}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\|z_{k+1} - x^\star\|^2\right]\right) \\
&\quad + \frac{1}{2\alpha}\mathbb{E}_{i_k}\left[\left\|\mathcal{H}^{\text{SAGA}}_{\phi_{i_k}^{k+1}}\right\|^2\right].
\end{aligned} \tag{A.13}$$

Using the interpolation condition of h_{i_k} to bound the stochastic moment,

$$\begin{aligned}
&\mathbb{E}_{i_k}\left[\left\|\mathcal{H}^{\text{SAGA}}_{\phi_{i_k}^{k+1}}\right\|^2\right] \\
&= \mathbb{E}_{i_k}\left[\left\|\nabla h_{i_k}(\phi_{i_k}^{k+1}) - \nabla h_{i_k}(\phi_{i_k}^{k})\right\|^2\right] + 2\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), \nabla\Phi^k\right\rangle\right] \\
&\quad - \left\|\nabla\Phi^k\right\|^2 \\
&\leq 2(L-\mu)\left(\Phi^k - \mathbb{E}_{i_k}\left[h_{i_k}(\phi_{i_k}^{k+1})\right] - \mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), \phi_{i_k}^k - \phi_{i_k}^{k+1}\right\rangle\right]\right) \\
&\quad + 2\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), \nabla\Phi^k\right\rangle\right] - \left\|\nabla\Phi^k\right\|^2.
\end{aligned} \tag{A.14}$$

Based on the updating rules of ϕ^{k+1}, the following relations hold

$$\mathbb{E}_{i_k}\left[\Phi^{k+1}\right] = \frac{1}{n}\mathbb{E}_{i_k}\left[h_{i_k}(\phi_{i_k}^{k+1})\right] + \frac{n-1}{n}\Phi^k, \tag{A.15}$$

$$\mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right] = \frac{1}{n}\mathbb{E}_{i_k}\left[\nabla h_{i_k}(\phi_{i_k}^{k+1})\right] + \frac{n-1}{n}\nabla\Phi^k, \tag{A.16}$$

where (A.16) implies that

$$\begin{aligned}
\left\|\mathbb{E}_{i_k}\left[\nabla h_{i_k}(\phi_{i_k}^{k+1})\right]\right\|^2 &= n^2\left\|\mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right]\right\|^2 - 2(n^2-n)\left\langle \mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right], \nabla\Phi^k\right\rangle \\
&\quad + (n-1)^2\left\|\nabla\Phi^k\right\|^2,
\end{aligned} \tag{A.17}$$

$$\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), \nabla\Phi^k\right\rangle\right] = n\left\langle \mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right], \nabla\Phi^k\right\rangle - (n-1)\left\|\nabla\Phi^k\right\|^2. \tag{A.18}$$

Then, expanding (A.12) using (A.13), (A.14), (A.17) and (A.18), we obtain

$$
\begin{aligned}
&\frac{1}{n}\mathbb{E}_{i_k}\left[h_{i_k}(\phi_{i_k}^{k+1})\right] \\
\leq\ & \left[\frac{1-\tau_x}{\tau_x n} - \left(1-\frac{\mu\tau_z}{\tau_x}\right)\frac{L-\mu}{\alpha n}\right]\mathbb{E}_{i_k}\left[\left\langle \nabla h_{i_k}(\phi_{i_k}^{k+1}), \phi_{i_k}^{k} - \phi_{i_k}^{k+1}\right\rangle\right] \\
&+ \left(1-\frac{\mu\tau_z}{\tau_x}\right)\frac{L-\mu}{\alpha n}\Big(\Phi^k - \mathbb{E}_{i_k}\left[h_{i_k}(\phi_{i_k}^{k+1})\right]\Big) \\
&+ \left(1-\frac{\mu\tau_z}{\tau_x}\right)\frac{\alpha}{2n}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2\mathbb{E}_{i_k}\left[\|z_{k+1}-x^\star\|^2\right]\right) \\
&+ \left[\frac{1}{\alpha} - \frac{\mu\tau_z}{\alpha\tau_x} - \frac{\tau_z}{\tau_x} + \frac{n-1}{L-\mu}\right]\left\langle \mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right], \nabla\Phi^k\right\rangle \\
&- \left[\frac{(n-1)^2}{2(L-\mu)n} + \left(1-\frac{\mu\tau_z}{\tau_x}\right)\frac{1}{2\alpha n} + \left(\frac{1}{\alpha} - \frac{\mu\tau_z}{\alpha\tau_x} - \frac{\tau_z}{\tau_x}\right)\frac{n-1}{n}\right]\left\|\nabla\Phi^k\right\|^2 \\
&- \frac{n}{2(L-\mu)}\left\|\mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right]\right\|^2.
\end{aligned}
$$

Choosing τ_z such that $\frac{1-\tau_x}{\tau_x} = \left(1-\frac{\mu\tau_z}{\tau_x}\right)\frac{L-\mu}{\alpha}$, multiplying both sides by τ_x and using (A.15), we can simplify the above inequality as

$$
\begin{aligned}
\mathbb{E}_{i_k}\left[\Phi^{k+1}\right] \leq\ & \left(1-\frac{\tau_x}{n}\right)\Phi^k + \frac{\alpha^2(1-\tau_x)}{2(L-\mu)n}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2\mathbb{E}_{i_k}\left[\|z_{k+1}-x^\star\|^2\right]\right) \\
&+ \frac{\alpha+\mu-\tau_x(\alpha+L+\mu-\mu n)}{(L-\mu)\mu}\left\langle \mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right], \nabla\Phi^k\right\rangle \\
&- \frac{(n-2)\tau_x + \frac{1}{n} + \left(\frac{\alpha}{\mu} + 1 - \left(\frac{\alpha}{\mu}+\kappa\right)\tau_x\right)\left(2-\frac{2}{n}\right)}{2(L-\mu)}\left\|\nabla\Phi^k\right\|^2 \\
&- \frac{n\tau_x}{2(L-\mu)}\left\|\mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right]\right\|^2.
\end{aligned}
$$

Fixing $\tau_x = \frac{\alpha+\mu}{\alpha+L}$, we obtain

$$
\begin{aligned}
\mathbb{E}_{i_k}\left[\Phi^{k+1}\right] \leq\ & \left(1-\frac{\tau_x}{n}\right)\Phi^k + \frac{\alpha^2(1-\tau_x)}{2(L-\mu)n}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2\mathbb{E}_{i_k}\left[\|z_{k+1}-x^\star\|^2\right]\right) \\
&+ \frac{(n-1)\tau_x}{L-\mu}\left\langle \mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right], \nabla\Phi^k\right\rangle - \frac{n\tau_x}{2(L-\mu)}\left\|\mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right]\right\|^2 \\
&- \frac{(n-2)\tau_x + \frac{1}{n}}{2(L-\mu)}\left\|\nabla\Phi^k\right\|^2.
\end{aligned}
$$

Using Young's inequality with $\beta > 0$,

$$\mathbb{E}_{i_k}\left[\Phi^{k+1}\right] \leq \left(1 - \frac{\tau_x}{n}\right)\Phi^k + \frac{\alpha^2(1-\tau_x)}{2(L-\mu)n}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2\mathbb{E}_{i_k}\left[\|z_{k+1}-x^\star\|^2\right]\right)$$
$$+ \frac{\beta(n-1)\tau_x - n\tau_x}{2(L-\mu)}\left\|\mathbb{E}_{i_k}\left[\nabla\Phi^{k+1}\right]\right\|^2 + \frac{\frac{(n-1)\tau_x}{\beta} - (n-2)\tau_x - \frac{1}{n}}{2(L-\mu)}\left\|\nabla\Phi^k\right\|^2.$$

Let $\beta \in \left[\frac{n-1}{n-2+\frac{1}{n\tau_x}}, \frac{n}{n-1}\right]$. The last two terms become non-positive, and thus we have

$$\mathbb{E}_{i_k}\left[\Phi^{k+1}\right] \leq \left(1 - \frac{\tau_x}{n}\right)\Phi^k + \frac{\alpha^2(1-\tau_x)}{2(L-\mu)n}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)^2\mathbb{E}_{i_k}\left[\|z_{k+1}-x^\star\|^2\right]\right).$$

Letting $\left(1-\frac{\tau_x}{n}\right)\left(1+\frac{\mu}{\alpha}\right)^2 = 1$ completes the proof.

A.4 Proof of Section 2.5 (Theorem 3)

Using Lemma 2 with the relations

$$x_{k+1} = \operatorname{prox}_{i_k}^{\alpha}\left(x_k + \frac{1}{\alpha}\left(\nabla f_{i_k}(\phi_{i_k}^k) - \frac{1}{n}\sum_{i=1}^n \nabla f_i(\phi_i^k) + \mu\left(\frac{1}{n}\sum_{i=1}^n \phi_i^k - \phi_{i_k}^k\right)\right)\right),$$
$$x^\star = \operatorname{prox}_{i_k}^{\alpha}\left(x^\star + \frac{1}{\alpha}\nabla f_{i_k}(x^\star)\right) \text{ and } \phi_{i_k}^{k+1} = x_{k+1},$$

and based on that $\nabla h_i(x) = \nabla f_i(x) - \nabla f_i(x^\star) - \mu(x - x^\star)$, we have

$$\left(1 + \frac{2(\alpha+\mu)}{L-\mu}\right)\left\|\nabla h_{i_k}(\phi_{i_k}^{k+1})\right\|^2 + (\alpha+\mu)^2\|x_{k+1}-x^\star\|^2$$
$$\leq \alpha^2\left\|x_k - x^\star + \frac{1}{\alpha}\left(\nabla h_{i_k}(\phi_{i_k}^k) - \frac{1}{n}\sum_{i=1}^n \nabla h_i(\phi_i^k)\right)\right\|^2.$$

Expanding the right side, taking the expectation and using $\mathbb{E}\left[\,\|X - \mathbb{E}X\|^2\,\right] \leq \mathbb{E}\left[\,\|X\|^2\,\right]$, we obtain

$$\left(1 + \frac{2(\alpha+\mu)}{L-\mu}\right)\mathbb{E}_{i_k}\left[\left\|\nabla h_{i_k}(\phi_{i_k}^{k+1})\right\|^2\right] + (\alpha+\mu)^2\mathbb{E}_{i_k}\left[\|x_{k+1}-x^\star\|^2\right]$$
$$\leq \alpha^2\|x_k - x^\star\|^2 + \frac{1}{n}\sum_{i=1}^n\left\|\nabla h_i(\phi_i^k)\right\|^2.$$

Note that by construction,

$$\mathbb{E}_{i_k}\left[\sum_{i=1}^{n}\left\|\nabla h_i(\phi_i^{k+1})\right\|^2\right] = \frac{n-1}{n}\sum_{i=1}^{n}\left\|\nabla h_i(\phi_i^{k})\right\|^2 + \mathbb{E}_{i_k}\left[\left\|\nabla h_{i_k}(\phi_{i_k}^{k+1})\right\|^2\right].$$

We can thus arrange the terms as

$$\begin{aligned}&\left(\frac{n}{\alpha^2}+\frac{2(\alpha+\mu)n}{\alpha^2(L-\mu)}\right)\mathbb{E}_{i_k}\left[\frac{1}{n}\sum_{i=1}^{n}\left\|\nabla h_i(\phi_i^{k+1})\right\|^2\right] + \left(1+\frac{\mu}{\alpha}\right)^2\mathbb{E}_{i_k}\left[\|x_{k+1}-x^\star\|^2\right]\\ \le\ &\left(\frac{n}{\alpha^2}+\frac{2(\alpha+\mu)(n-1)}{\alpha^2(L-\mu)}\right)\cdot\frac{1}{n}\sum_{i=1}^{n}\left\|\nabla h_i(\phi_i^{k})\right\|^2 + \|x_k-x^\star\|^2.\end{aligned}$$

In view of the Lyapunov function, we choose α to be the positive root of the following equation:

$$\left(1+\frac{\mu}{\alpha}\right)^2\left(1-\frac{2(\alpha+\mu)}{n(L-\mu)+2n(\alpha+\mu)}\right) = 1.$$

Let $q = \frac{\alpha}{\mu} > 0$, the above is a cubic equation:

$$s(q) \triangleq 2q^3 - (4n-6)q^2 - (2n\kappa+4n-6)q - (n\kappa+n-2) = 0,$$

which has a unique positive root (denoted as $q^\star$).

Note that $s(-\infty) < 0, s(-\frac{1}{2}) = \frac{1}{4}$ and $s(0) \le 0$. These facts suggest that if for some $u > 0$, $s(u) > 0$, we have $q^\star < u$. It can be verified that $s(2n+\sqrt{n\kappa}) > 0$, and thus $q^\star = O(n+\sqrt{n\kappa})$.

A.5 Experimental Setup

We ran experiments on an HP Z440 machine with a single Intel Xeon E5-1630v4 with 3.70GHz cores, 16GB RAM, Ubuntu 18.04 LTS with GCC 4.8.0, MATLAB R2017b. We were optimizing the following binary problems with $a_i \in \mathbb{R}^d$, $b_i \in \{-1,+1\}$, $i \in [n]$:

$$\ell_2\text{-Logistic Regression: } \frac{1}{n}\sum_{i=1}^{n}\log\left(1+\exp\left(-b_i\langle a_i, x\rangle\right)\right) + \frac{\mu}{2}\|x\|^2,$$

$$\text{Ridge Regression: } \frac{1}{2n}\sum_{i=1}^{n}\left(\langle a_i, x\rangle - b_i\right)^2 + \frac{\mu}{2}\|x\|^2.$$

We used datasets from the LIBSVM website [20], including a9a (32,561 samples, 123 features), covtype.binary (581,012 samples, 54 features), w8a (49,749 samples, 300 features), ijcnn1 (49,990 samples, 22 features). We added one dimension as bias to all the datasets.

We choose SAGA and Katyusha as the baselines in the finite-sum experiments due to the following reasons: SAGA has low iteration cost and good empirical performance with support for non-smooth regularizers, and is thus implemented in machine learning libraries such as scikit-learn [119]; Katyusha achieves the state-of-the-art performance for ill-conditioned problems[1].

A.6 Analyzing NAG using Lyapunov Function

In this section, we review the convergence of NAG in the strongly convex setting for a better comparison with the convergence guarantee and proof of G-TM. This Lyapunov analysis has been similarly presented in many existing works, e.g., [149, 57, 10, 116]. We adopt a simplified version of NAG in Algorithm 16 (1-memory accelerated methods, [148]) and only consider constant parameter choices. It is known that NAG can be analyzed based on the following Lyapunov function ($\lambda > 0$):

$$T_k = f(x_k) - f(x^\star) \; + \frac{\lambda}{2} \left\| z_k - x^\star \right\|^2, \tag{A.19}$$

which is somehow suggested in the construction of the *estimate sequence* in Nesterov [109]. This choice requires neither $f(x_k) - f(x^\star)$ nor $\left\| z_k - x^\star \right\|^2$ to be monotone decreasing over iterations, which is called the non-relaxational property in Nesterov [104]. By re-organizing the proof in Nesterov [109] under the notion of Lyapunov function, we obtain the per-iteration contraction of NAG in Theorem A.6.1.

Theorem A.6.1. *In Algorithm 16, suppose we choose α, τ_x, τ_y under the constraints* (A.20)*, the iterations satisfy the contraction* (A.21) *for the Lyapunov function* (A.19).

$$\begin{cases} \alpha \geq \frac{L(1-\tau_x)\tau_y}{1-\tau_y}, \tau_x \geq \tau_y, \\ \mu \geq \frac{L(\tau_x - \tau_y)}{1-\tau_y}, \\ \left(1 + \frac{\mu}{\alpha}\right)(1-\tau_x) \leq 1. \end{cases} \tag{A.20}$$

$$\text{With } \lambda = (\alpha + \mu)\tau_x, \quad T_{k+1} \leq \left(1 + \frac{\mu}{\alpha}\right)^{-1} T_k, \; \text{for } k \geq 0. \tag{A.21}$$

[1]Zhou et al. [159] shows that SSNM can be faster than Katyusha in some cases. In theory, SSNM and Katyusha achieve the same rate if we set $m = n$ for Katyusha (both require 2 oracle calls per-iteration). In practice, if $m = n$, they have similar performance (SSNM is often faster). Considering the stability and memory requirement, Katyusha still achieves the state-of-the-art performance both theoretically and empirically.

Algorithm 16 Nesterov's Accelerated Gradient (NAG)

```
Input: Parameters $\alpha > 0, \tau_y, \tau_x \in ]0,1[$ and initial guesses $x_0, z_0 \in \mathbb{R}^d$, iteration
    number $K$.
1: for $k = 0, \ldots, K-1$ do
2:     $y_k = \tau_y z_k + (1-\tau_y)\, x_k$.
3:     $z_{k+1} = \arg\min_x \left\{ \langle \nabla f(y_k), x \rangle + (\alpha/2) \|x - z_k\|^2 + (\mu/2) \|x - y_k\|^2 \right\}$.
4:     $x_{k+1} = \tau_x z_{k+1} + (1-\tau_x) x_k$.
5: end for
Output: $x_K$.
```

When the inequalities in constraints (A.20) (except $\tau_x \geq \tau_y$) hold as equality, we derive the standard choice of NAG: $\alpha = \sqrt{L\mu} - \mu, \tau_y = (\sqrt{\kappa}+1)^{-1}, \tau_x = (\sqrt{\kappa})^{-1}$. By substituting this choice and eliminating sequence $\{z_k\}$, we recover the widely-used scheme (Constant Step scheme III in Nesterov [109]):

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k),$$
$$y_{k+1} = x_{k+1} + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}(x_{k+1} - x_k).$$

Telescoping (A.21), we obtain the original guarantee of NAG (cf. Theorem 2.2.3 in Nesterov [109]),

$$f(x_K) - f(x^\star) + \frac{\mu}{2}\|z_K - x^\star\|^2 \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right)^K \left(f(x_0) - f(x^\star) + \frac{\mu}{2}\|z_0 - x^\star\|^2\right).$$

If we regard the constraints (A.20) as an optimization problem with a target of minimizing the rate factor $(1+\frac{\mu}{\alpha})^{-1}$, the rate factor $1 - 1/\sqrt{\kappa}$ is optimal. Combining $\alpha \geq \frac{L(1-\tau_x)\tau_y}{1-\tau_y}$ and $\mu \geq \frac{L(\tau_x-\tau_y)}{1-\tau_y}$, we have $\alpha \geq L\tau_x - \mu$. To minimize α, we fix $\alpha = L\tau_x - \mu$, and it can be easily verified that in this case, the smallest rate factor is achieved when $\left(1+\frac{\mu}{\alpha}\right)(1-\tau_x) = 1$. Note that these arguments do not consider variable-parameter choices and are limited to the current analysis framework only.

Denote the initial constant as $C_0^{\text{NAG}} \triangleq f(x_0) - f(x^\star) + \frac{\mu}{2}\|z_0 - x^\star\|^2$. This guarantee shows that in terms of reducing $\|x - x^\star\|^2$ to ϵ, sequences $\{x_k\}$ and $\{z_k\}$ have the same iteration complexity $\sqrt{\kappa}\log\frac{2C_0^{\text{NAG}}}{\mu\epsilon}$. Since $\{y_k\}$ is a convex combination of them, it also converges with the same complexity.

A.6.1 Proof of Theorem A.6.1

For the convex combination $y_k = \tau_y z_k + (1-\tau_y)\, x_k$, we can use the trick in Lemma 3 to obtain

$$\begin{aligned} f(y_k) - f(x^\star) &\le \frac{1-\tau_y}{\tau_y}\langle \nabla f(y_k), x_k - y_k\rangle + \langle \nabla f(y_k), z_k - x^\star\rangle - \frac{\mu}{2}\left\|y_k - x^\star\right\|^2 \\ &= \frac{1-\tau_y}{\tau_y}\langle \nabla f(y_k), x_k - y_k\rangle + \underbrace{\langle \nabla f(y_k), z_k - z_{k+1}\rangle}_{R_1} \\ &\quad + \underbrace{\langle \nabla f(y_k), z_{k+1} - x^\star\rangle}_{R_2} - \frac{\mu}{2}\left\|y_k - x^\star\right\|^2 . \end{aligned} \tag{A.22}$$

For R_1, based on the L-smoothness, we have

$$f(x_{k+1}) - f(y_k) + \langle \nabla f(y_k), y_k - x_{k+1}\rangle \le \frac{L}{2}\left\|x_{k+1} - y_k\right\|^2 .$$

Note that $y_k - x_{k+1} = \tau_x(z_k - z_{k+1}) + (\tau_y - \tau_x)(z_k - x_k)$, we can arrange the above inequality as

$$f(x_{k+1}) - f(y_k) + \langle \nabla f(y_k), \tau_x(z_k - z_{k+1}) + (\tau_y - \tau_x)(z_k - x_k)\rangle \le \frac{L}{2}\left\|x_{k+1} - y_k\right\|^2 ,$$

$$R_1 \le \frac{L}{2\tau_x}\left\|x_{k+1} - y_k\right\|^2 + \frac{1}{\tau_x}\big(f(y_k) - f(x_{k+1})\big) - \frac{\tau_y - \tau_x}{\tau_x}\langle \nabla f(y_k), z_k - x_k\rangle . \tag{A.23}$$

For R_2, based on the optimality condition of the 3rd step in Algorithm 16, which is for any $u \in \mathbb{R}^d$,

$$\langle \nabla f(y_k) + \alpha(z_{k+1} - z_k) + \mu(z_{k+1} - y_k), u - z_{k+1}\rangle = 0,$$

we have (by choosing $u = x^\star$),

$$\begin{aligned} R_2 &= \alpha\langle z_{k+1} - z_k, x^\star - z_{k+1}\rangle + \mu\langle z_{k+1} - y_k, x^\star - z_{k+1}\rangle \\ &= \frac{\alpha}{2}(\left\|z_k - x^\star\right\|^2 - \left\|z_{k+1} - x^\star\right\|^2 - \left\|z_{k+1} - z_k\right\|^2) \\ &\quad + \frac{\mu}{2}(\left\|y_k - x^\star\right\|^2 - \left\|z_{k+1} - x^\star\right\|^2 - \left\|z_{k+1} - y_k\right\|^2). \end{aligned} \tag{A.24}$$

By upper bounding (A.22) using (A.23), (A.24), we can conclude that

$$\begin{aligned} f(y_k) - f(x^\star) \le{}& \frac{1-\tau_x}{\tau_x}\langle \nabla f(y_k), x_k - y_k\rangle + \frac{1}{\tau_x}\big(f(y_k) - f(x_{k+1})\big) \\ &+ \frac{\alpha}{2}\left(\left\|z_k - x^\star\right\|^2 - \left(1 + \frac{\mu}{\alpha}\right)\left\|z_{k+1} - x^\star\right\|^2\right) \\ &+ \frac{L}{2\tau_x}\left\|x_{k+1} - y_k\right\|^2 - \frac{\alpha}{2}\left\|z_{k+1} - z_k\right\|^2 - \frac{\mu}{2}\left\|z_{k+1} - y_k\right\|^2 , \end{aligned}$$

Re-arrange the terms,

$$\begin{aligned} & f(x_{k+1}) - f(x^\star) \\ \leq{} & (1-\tau_x)\big(f(x_k) - f(x^\star)\big) + \frac{\alpha\tau_x}{2}\left(\|z_k - x^\star\|^2 - \left(1+\frac{\mu}{\alpha}\right)\|z_{k+1} - x^\star\|^2\right) \\ & + \frac{L}{2}\|x_{k+1} - y_k\|^2 - \frac{\alpha\tau_x}{2}\|z_{k+1} - z_k\|^2 - \frac{\mu\tau_x}{2}\|z_{k+1} - y_k\|^2 . \end{aligned} \tag{A.25}$$

Note that the following relation holds:

$$x_{k+1} - y_k = \tau_x\left(\frac{(1-\tau_x)\tau_y}{(1-\tau_y)\tau_x}(z_{k+1} - z_k) + \frac{\tau_x - \tau_y}{(1-\tau_y)\tau_x}(z_{k+1} - y_k)\right),$$

and thus if $\tau_x \geq \tau_y$, based on the convexity of $\|\cdot\|^2$, we have

$$\frac{L}{2}\|x_{k+1} - y_k\|^2 \leq \frac{L(1-\tau_x)\tau_x\tau_y}{2(1-\tau_y)}\|z_{k+1} - z_k\|^2 + \frac{L(\tau_x - \tau_y)\tau_x}{2(1-\tau_y)}\|z_{k+1} - y_k\|^2 .$$

Finally, suppose that the following relations hold

$$\begin{cases} \tau_x \geq \tau_y, \\ \alpha \geq \frac{L(1-\tau_x)\tau_y}{1-\tau_y}, \\ \mu \geq \frac{L(\tau_x - \tau_y)}{1-\tau_y}, \\ \left(1+\frac{\mu}{\alpha}\right)(1-\tau_x) \leq 1, \end{cases}$$

we can arrange (A.25) as

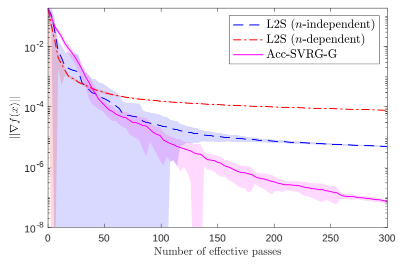
$$\begin{aligned} & f(x_{k+1}) - f(x^\star) + \frac{\alpha\tau_x}{2}\left(1+\frac{\mu}{\alpha}\right)\|z_{k+1} - x^\star\|^2 \\ \leq{} & \left(1+\frac{\mu}{\alpha}\right)^{-1}\left(f(x_k) - f(x^\star) + \frac{\alpha\tau_x}{2}\left(1+\frac{\mu}{\alpha}\right)\|z_k - x^\star\|^2\right), \end{aligned}$$

which completes the proof.

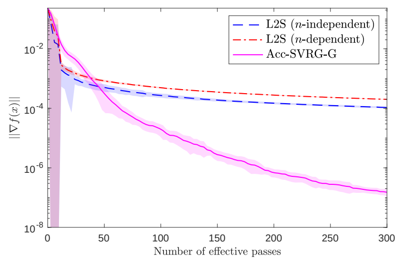
Appendix B

Appendix for Chapter 3

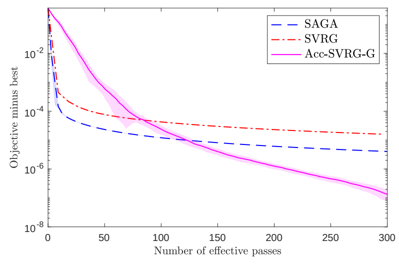
B.1 Numerical Results of Acc-SVRG-G



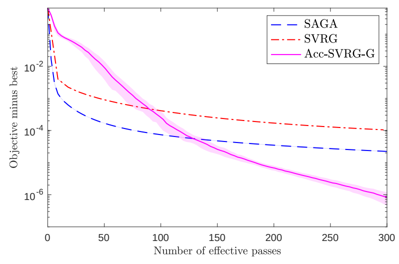
(a) a9a dataset. Measuring the gradient norm.



(b) w8a dataset. Measuring the gradient norm.



(c) a9a dataset. Measuring the function value.



(d) w8a dataset. Measuring the function value.

Figure B.1: Run 20 seeds. Shaded bands indicate ±1 standard deviation.

We did some experiments to justify the theoretical results (Theorem 7) of Acc-SVRG-G. We compared it with non-accelerated methods including L2S [88], SVRG [64, 152] and SAGA [29] under their original optimality measures. Note that other stochastic approaches in Table 3.1 require fixing the accuracy ϵ in advance, and thus it is not convenient to compare them in the form of Figure B.1. For measuring the gradient norm, we simply tracked the smallest norm of all the full gradient computed to reduce complexity. Since the figures are in logarithmic scale, the deviation bands are asymmetric, and will emphasize the passes that have large deviations.

Setups. We ran the experiments on a Macbook Pro with a quad-core Intel Core i7-4870HQ with 2.50GHz cores, 16GB RAM, macOS Big Sur with Clang 12.0.5 and MATLAB R2020b. We were optimizing the binary logistic regression problem $f(x) = \frac{1}{n}\sum_{i=1}^{n} \log\big(1 + \exp\left(-b_i \langle a_i, x\rangle\right)\big)$ with dataset $a_i \in \mathbb{R}^d$, $b_i \in \{-1, +1\}$, $i \in [n]$. We used datasets from the LIBSVM website [20], including a9a [38] (32,561 samples, 123 features) and w8a [122] (49,749 samples, 300 features). We added one dimension as bias to all the datasets. We normalized the datasets and thus for this problem, $L = 0.25$. For Acc-SVRG-G, we chose the parameters according to Theorem 7. For L2S, we set $m = n$ and for its n-independent step size, we chose $\eta = \frac{c}{L}$ and tuned c using the same grid specified in [88]; for the n-dependent step size, we set $\eta = \frac{1}{L\sqrt{n}}$ according to Corollary 3 in [88]. For SAGA [29], we chose $\eta = \frac{1}{3L}$ following its theory. For SVRG [152], we set $\eta = \frac{1}{4L}$.

B.2 Proofs of Section 3.2

To simplify the proof, we denote $D_k \triangleq f(x_k) - f(x^\star)$. And we use the following reformulation of interpolation condition (1.2) (at (x, y)) to facilitate our proof.

$$\begin{aligned}&\frac{1}{2L}\left(\|\nabla f(x)\|^2 + \|\nabla f(y)\|^2\right) + \left\langle \nabla f(y), x - y - \frac{1}{L}\nabla f(x)\right\rangle \\ &\leq f(x) - f(y), \forall x, y \in \mathbb{R}^d.\end{aligned} \tag{B.1}$$

B.2.1 Proof of Proposition 3.1

We define $\theta_{N+1}^2 = \theta_N^2 - \theta_N = 0$. At iteration k, we are going to combine the reformulated interpolation conditions (B.1) at (x_k, x_{k+1}) and (x_N, x_k) with multipliers $\frac{1}{\theta_{k+1}^2}$ and $\frac{1}{\theta_k \theta_{k+1}^2}$, respectively.

$$\begin{aligned}&\frac{1}{2L\theta_{k+1}^2}\left(\|\nabla f(x_k)\|^2 + \|\nabla f(x_{k+1})\|^2\right) \\ &\leq \frac{1}{\theta_{k+1}^2}(D_k - D_{k+1}) - \frac{1}{\theta_{k+1}^2}\left\langle \nabla f(x_{k+1}), x_k - x_{k+1} - \frac{1}{L}\nabla f(x_k)\right\rangle,\end{aligned} \tag{B.2}$$

$$\begin{aligned}&\frac{1}{2L\theta_k\theta_{k+1}^2}\left(\|\nabla f(x_N)\|^2 + \|\nabla f(x_k)\|^2\right) \\ &\leq \frac{1}{\theta_k\theta_{k+1}^2}(D_N - D_k) - \frac{1}{\theta_k\theta_{k+1}^2}\left\langle \nabla f(x_k), x_N - x_k - \frac{1}{L}\nabla f(x_N)\right\rangle.\end{aligned} \tag{B.3}$$

Using the construction: $x_k - x_{k+1} = \frac{1}{L}\nabla f(x_k) + (2\theta_{k+1}^3 - \theta_{k+1}^2)v_{k+1}$, we can write (B.2) as

$$\begin{aligned}&\frac{1}{2L\theta_{k+1}^2}\left(\|\nabla f(x_k)\|^2 + \|\nabla f(x_{k+1})\|^2\right) + (2\theta_{k+1} - 1)\left\langle \nabla f(x_{k+1}), v_{k+1}\right\rangle \\ &\leq \frac{1}{\theta_{k+1}^2}(D_k - D_{k+1}).\end{aligned} \tag{B.4}$$

Note that using $\theta_k^2 - \theta_k = \theta_{k+1}^2$, we have $2\theta_{k+1}^3 - \theta_{k+1}^2 = \theta_{k+1}^4 - \theta_{k+2}^4$. Then,

$$\begin{aligned} x_k - x_N = \sum_{i=k}^{N-1} (x_i - x_{i+1}) &= \frac{1}{L}\sum_{i=k}^{N-1} \nabla f(x_i) + \sum_{i=k}^{N-1} (\theta_{i+1}^4 - \theta_{i+2}^4) v_{i+1} \\ &= \frac{1}{L}\sum_{i=k}^{N-1} \nabla f(x_i) + \theta_{k+1}^4 v_{k+1} + \sum_{i=k}^{N-2} \theta_{i+2}^4 (v_{i+2} - v_{i+1}) \\ &\overset{(a)}{=} \frac{1}{L}\sum_{i=k}^{N-1} \nabla f(x_i) + \theta_{k+1}^4 v_{k+1} + \sum_{i=k}^{N-2} \frac{\theta_{i+2}^2}{L\theta_{i+1}} \nabla f(x_{i+1}) \\ &\overset{(b)}{=} \theta_{k+1}^4 v_k + \sum_{i=k}^{N-1} \frac{\theta_i}{L} \nabla f(x_i), \end{aligned}$$

where (a) and (b) use the construction: $v_{k+1} = v_k + \frac{1}{L\theta_k\theta_{k+1}^2}\nabla f(x_k)$.

Thus, (B.3) can be written as

$$\begin{aligned} \frac{1}{\theta_k\theta_{k+1}^2}(D_N - D_k) \geq \frac{1}{2L\theta_k\theta_{k+1}^2} \left\|\nabla f(x_N)\right\|^2 - \frac{\theta_k^2 + \theta_{k+1}^2}{2L\theta_k^2\theta_{k+1}^2} \left\|\nabla f(x_k)\right\|^2 \\ - \frac{\theta_{k+1}^2}{\theta_k} \langle \nabla f(x_k), v_k \rangle - \sum_{i=k+1}^{N} \frac{\theta_i}{L\theta_k\theta_{k+1}^2} \langle \nabla f(x_k), \nabla f(x_i) \rangle. \end{aligned}$$

Summing this inequality and (B.4), and using the relation $\theta_k^2 - \theta_k = \theta_{k+1}^2$, we obtain

$$\begin{aligned} &\left(\frac{1}{\theta_{k+1}^2} - \frac{1}{\theta_k^2}\right)\left(D_N - \frac{1}{2L}\left\|\nabla f(x_N)\right\|^2\right) + \left(\frac{1}{\theta_k^2} D_k - \frac{1}{\theta_{k+1}^2} D_{k+1}\right) \\ \geq\ &\left(\frac{1}{2L\theta_{k+1}^2}\left\|\nabla f(x_{k+1})\right\|^2 - \frac{1}{2L\theta_k^2}\left\|\nabla f(x_k)\right\|^2\right) \\ &+ \left(\frac{\theta_{k+2}^2}{\theta_{k+1}} \langle \nabla f(x_{k+1}), v_{k+1} \rangle - \frac{\theta_{k+1}^2}{\theta_k} \langle \nabla f(x_k), v_k \rangle\right) \\ &+ \underbrace{\theta_{k+1} \langle \nabla f(x_{k+1}), v_{k+1} \rangle - \sum_{i=k+1}^{N} \frac{\theta_i}{L\theta_k\theta_{k+1}^2} \langle \nabla f(x_k), \nabla f(x_i) \rangle}_{\mathcal{R}_1}. \end{aligned} \tag{B.5}$$

B.2.2 Proof of Theorem 4

Clearly, except for $\mathcal{R}_1$, all terms in (B.5) telescope. Since $v_{k+1} = \sum_{i=0}^{k} \frac{1}{L\theta_i\theta_{i+1}^2}\nabla f(x_i)$, by defining a matrix $P \in \mathbb{R}^{(N+1)\times(N+1)}$ with $P_{ki} = \frac{\theta_k}{L\theta_i\theta_{i+1}^2}\langle\nabla f(x_k), \nabla f(x_i)\rangle$, we can write $\mathcal{R}_1$ as $\sum_{i=0}^{k} P_{(k+1)i} - \sum_{i=k+1}^{N} P_{ik}$. Summing these terms from $k = 0$ to $N-1$, we obtain

$$\sum_{k=0}^{N-1}\sum_{i=0}^{k} P_{(k+1)i} - \sum_{k=0}^{N-1}\sum_{i=k+1}^{N} P_{ik} = \sum_{k=1}^{N}\sum_{i=0}^{k-1} P_{ki} - \sum_{i=0}^{N-1}\sum_{k=i+1}^{N} P_{ki} = 0.$$

Both of the summations are equal to the sum of the lower triangular entries of P.

Then, telescoping (B.5) from $k = 0$ to $N-1$ (note that $v_0 = \mathbf{0}$), we obtain

$$\left(1 - \frac{1}{\theta_0^2}\right)\left(D_N - \frac{1}{2L}\left\|\nabla f(x_N)\right\|^2\right) \geq D_N - \frac{1}{\theta_0^2}D_0 + \frac{1}{2L}\left\|\nabla f(x_N)\right\|^2 - \frac{1}{2L\theta_0^2}\left\|\nabla f(x_0)\right\|^2.$$

Using $D_0 \geq \frac{1}{2L}\left\|\nabla f(x_0)\right\|^2$ and $D_N \geq \frac{1}{2L}\left\|\nabla f(x_N)\right\|^2$, we obtain

$$\left\|\nabla f(x_N)\right\|^2 \leq \frac{2LD_0}{\theta_0^2}.$$

Since $\theta_k = \frac{1+\sqrt{1+4\theta_{k+1}^2}}{2} \geq \frac{1}{2} + \theta_{k+1} \Rightarrow \theta_k \geq \frac{N-k}{2} + 1 \Rightarrow \theta_0 \geq \frac{N+2}{2}$, we have

$$\left\|\nabla f(x_N)\right\|^2 \leq \frac{8L\big(f(x_0) - f(x^\star)\big)}{(N+2)^2}.$$

B.2.3 Proof of Theorem 5

Define for $k = 0, \ldots, N$,

$$\begin{aligned}\tau_k &\triangleq \frac{(N-k+2)(N-k+3)}{6},\\ \delta_{k+1} &\triangleq \frac{12}{(N-k+1)(N-k+2)(N-k+3)} = \frac{1}{\tau_{k+1}} - \frac{1}{\tau_k}.\end{aligned}$$

At iteration k, we are going to combine the reformulated interpolation conditions (B.1) at (x_k, x_{k+1}) and (x_N, x_k) with multipliers $\frac{1}{\tau_{k+1}}$ and δ_{k+1}, respectively.

$$\begin{aligned}&\frac{1}{2L\tau_{k+1}}\big(\left\|\nabla f(x_k)\right\|^2 + \left\|\nabla f(x_{k+1})\right\|^2\big) + \frac{1}{\tau_{k+1}}\left\langle\nabla f(x_{k+1}), x_k - x_{k+1} - \frac{1}{L}\nabla f(x_k)\right\rangle\\ &\leq \frac{1}{\tau_{k+1}}(D_k - D_{k+1}),\end{aligned} \tag{B.6}$$

$$\begin{aligned}&\frac{\delta_{k+1}}{2L}\left(\|\nabla f(x_N)\|^2+\|\nabla f(x_k)\|^2\right)+\delta_{k+1}\left\langle \nabla f(x_k), x_N-x_k-\frac{1}{L}\nabla f(x_N)\right\rangle\\&\leq \delta_{k+1}(D_N-D_k).\end{aligned}\tag{B.7}$$

Note that from the construction of Algorithm 5,

$$\begin{aligned}x_k-x_{k+1}-\frac{1}{L}\nabla f(x_k)&=\frac{(N-k)(N-k+1)(N-k+2)}{6}v_{k+1},\\ x_k-x_N&=\sum_{i=k}^{N-1}\frac{1}{L}\nabla f(x_i)+\sum_{i=k}^{N-1}\frac{(N-i)(N-i+1)(N-i+2)}{6}v_{i+1}.\end{aligned}$$

Thus, (B.6) can be written as

$$\begin{aligned}&\frac{1}{2L\tau_{k+1}}\left(\|\nabla f(x_k)\|^2+\|\nabla f(x_{k+1})\|^2\right)+(N-k)\left\langle \nabla f(x_{k+1}), v_{k+1}\right\rangle\\&\leq \frac{1}{\tau_{k+1}}(D_k-D_{k+1}).\end{aligned}\tag{B.8}$$

Defining $\mathcal{Q}(j)\triangleq(j+3)(j+2)(j+1)j$, we have $\mathcal{Q}(j)-\mathcal{Q}(j-1)=4j(j+1)(j+2)$. Then,

$$\begin{aligned}x_k-x_N&=\sum_{i=k}^{N-1}\frac{1}{L}\nabla f(x_i)+\frac{1}{24}\sum_{i=k}^{N-1}\left(\mathcal{Q}(N-i)-\mathcal{Q}(N-i-1)\right)v_{i+1}\\&=\sum_{i=k}^{N-1}\frac{1}{L}\nabla f(x_i)+\frac{1}{24}\left(\mathcal{Q}(N-k)v_{k+1}+\sum_{i=k+1}^{N-1}\mathcal{Q}(N-i)(v_{i+1}-v_i)\right)\\&\overset{(a)}{=}\frac{\mathcal{Q}(N-k)}{24}v_{k+1}+\frac{1}{L}\nabla f(x_k)+\sum_{i=k+1}^{N-1}\frac{1}{L}\left(\frac{\mathcal{Q}(N-i)\delta_{i+1}}{24}+1\right)\nabla f(x_i)\\&\overset{(b)}{=}\frac{\mathcal{Q}(N-k)}{24}v_k+\sum_{i=k}^{N-1}\frac{N-i+2}{2L}\nabla f(x_i),\end{aligned}$$

where (a) and (b) use the construction $v_{k+1}=v_k+\frac{\delta_{k+1}}{L}\nabla f(x_k)$.

Thus, (B.7) can be written as

$$\begin{aligned}\delta_{k+1}(D_N-D_k)\geq\ &\frac{\delta_{k+1}}{2L}\left(\|\nabla f(x_N)\|^2+\|\nabla f(x_k)\|^2\right)-\frac{N-k}{2}\left\langle \nabla f(x_k), v_k\right\rangle\\&-\frac{(N-k+2)\delta_{k+1}}{2L}\|\nabla f(x_k)\|^2\\&-\sum_{i=k+1}^{N}\frac{(N-i+2)\delta_{k+1}}{2L}\left\langle \nabla f(x_k), \nabla f(x_i)\right\rangle.\end{aligned}$$

Summing the above inequality and (B.8), we obtain

$$\begin{aligned}
&\left(\frac{1}{\tau_{k+1}}-\frac{1}{\tau_k}\right)\left(D_N-\frac{1}{2L}\left\|\nabla f(x_N)\right\|^2\right)+\left(\frac{1}{\tau_k}D_k-\frac{1}{\tau_{k+1}}D_{k+1}\right)\\
\geq\ &\left(\frac{1}{2L\tau_{k+1}}\left\|\nabla f(x_{k+1})\right\|^2-\frac{1}{2L\tau_k}\left\|\nabla f(x_k)\right\|^2\right)+\frac{\delta_{k+1}}{2L}\left\|\nabla f(x_k)\right\|^2\\
&+\left(\frac{N-k-1}{2}\left\langle\nabla f(x_{k+1}),v_{k+1}\right\rangle-\frac{N-k}{2}\left\langle\nabla f(x_k),v_k\right\rangle\right)\\
&+\frac{N-k+1}{2}\left\langle\nabla f(x_{k+1}),v_{k+1}\right\rangle-\sum_{i=k+1}^{N}\frac{(N-i+2)\delta_{k+1}}{2L}\left\langle\nabla f(x_k),\nabla f(x_i)\right\rangle.
\end{aligned}\tag{B.9}$$

Since $v_{k+1}=\sum_{i=0}^{k}\frac{\delta_{i+1}}{L}\nabla f(x_i)$, the last two terms above have a similar structure as $\mathcal{R}_1$ at (B.5). Define a matrix $P\in\mathbb{R}^{(N+1)\times(N+1)}$ with $P_{ki}=\frac{(N-k+2)\delta_{i+1}}{2L}\left\langle\nabla f(x_k),\nabla f(x_i)\right\rangle$. The last two terms above can be written as $\sum_{i=0}^{k}P_{(k+1)i}-\sum_{i=k+1}^{N}P_{ik}$. If we sum these terms from $k=0,\dots,N-1$, they sum up to 0 (see Section B.2.2). Then, by telescoping (B.9) from $k=0,\dots,N-1$, we obtain

$$\begin{aligned}
&\frac{1}{2L}\left\|\nabla f(x_N)\right\|^2-\frac{1}{2L\tau_0}\left\|\nabla f(x_0)\right\|^2+\frac{1-\frac{1}{\tau_0}}{2L}\left\|\nabla f(x_N)\right\|^2+\sum_{k=0}^{N-1}\frac{\delta_{k+1}}{2L}\left\|\nabla f(x_k)\right\|^2\\
\leq\ &\left(1-\frac{1}{\tau_0}\right)D_N+\frac{1}{\tau_0}D_0-D_N.
\end{aligned}$$

Finally, using $D_0\geq\frac{1}{2L}\left\|\nabla f(x_0)\right\|^2$ and $D_N\geq\frac{1}{2L}\left\|\nabla f(x_N)\right\|^2$, we obtain

$$\left\|\nabla f(x_N)\right\|^2+\sum_{k=0}^{N-1}\frac{\delta_{k+1}}{2}\left\|\nabla f(x_k)\right\|^2\leq\frac{2L}{\tau_0}D_0=\frac{12L\big(f(x_0)-f(x^\star)\big)}{(N+2)(N+3)}.\tag{B.10}$$

B.2.4 Proof of Corollary 5.1

We assume N is divisible by 2 for simplicity. After running $N/2$ iterations of NAG, we obtain an output $x_{N/2}$ satisfying (cf. Theorem 2.2.2 in [109])

$$f(x_{N/2})-f(x^\star)=O\left(\frac{LR_0^2}{N^2}\right).$$

Then, let $x_{N/2}$ be the input of Algorithm 5. Using (B.10), after running another $N/2$ iterations of Algorithm 5, we obtain

$$\left\|\nabla f(x_N)\right\|^2=O\left(\frac{L^2R_0^2}{N^4}\right).$$

B.3 Proofs of Section 3.3

B.3.1 Proof of Proposition 5.1

Using the interpolation condition (1.2) at $(x^\star, y_k)$, we obtain

$$\begin{aligned} f(y_k) - f(x^\star) &\leq \langle \nabla f(y_k), y_k - x^\star \rangle - \frac{1}{2L} \|\nabla f(y_k)\|^2 \\ &\overset{(\star)}{\leq} \frac{1-\tau_k}{\tau_k} \langle \nabla f(y_k), \tilde{x}_k - y_k \rangle - \frac{1-\tau_k}{L\tau_k} \langle \nabla f(y_k), \nabla f(\tilde{x}_k) \rangle \\ &\quad + \langle \nabla f(y_k), z_k - x^\star \rangle - \frac{1}{2L} \|\nabla f(y_k)\|^2, \end{aligned} \tag{B.11}$$

where $(\star)$ follows from the construction $y_k = \tau_k z_k + (1-\tau_k)\big(\tilde{x}_k - \frac{1}{L}\nabla f(\tilde{x}_k)\big)$.

From the optimality condition of Step 3, we can conclude that

$$\begin{aligned} &\mathcal{G}_k + \alpha_k (z_{k+1} - z_k) = \mathbf{0} \\ &\overset{(a)}{\Rightarrow} \langle \mathcal{G}_k, z_k - x^\star \rangle = \frac{1}{2\alpha_k} \|\mathcal{G}_k\|^2 + \frac{\alpha_k}{2} \left(\|z_k - x^\star\|^2 - \|z_{k+1} - x^\star\|^2 \right) \\ &\overset{(b)}{\Rightarrow} \langle \nabla f(y_k), z_k - x^\star \rangle = \frac{1}{2\alpha_k} \mathbb{E}_{i_k} \left[\|\mathcal{G}_k\|^2 \right] + \frac{\alpha_k}{2} \left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k} \left[\|z_{k+1} - x^\star\|^2 \right] \right), \end{aligned} \tag{B.12}$$

where (a) uses $\langle u, v \rangle = \frac{1}{2}(\|u\|^2 + \|v\|^2 - \|u-v\|^2)$ and (b) follows from taking the expectation wrt sample i_k.

Using the interpolation condition (1.2) at $(\tilde{x}_k, y_k)$, we can bound $\mathbb{E}_{i_k}\left[\|\mathcal{G}_k\|^2\right]$ as

$$\begin{aligned} \mathbb{E}_{i_k} \left[\|\mathcal{G}_k\|^2 \right] &= \mathbb{E}_{i_k} \left[\|\nabla f_{i_k}(y_k) - \nabla f_{i_k}(\tilde{x}_k)\|^2 \right] + 2 \langle \nabla f(y_k), \nabla f(\tilde{x}_k) \rangle - \|\nabla f(\tilde{x}_k)\|^2 \\ &\leq 2L \big(f(\tilde{x}_k) - f(y_k) - \langle \nabla f(y_k), \tilde{x}_k - y_k \rangle \big) + 2 \langle \nabla f(y_k), \nabla f(\tilde{x}_k) \rangle \\ &\quad - \|\nabla f(\tilde{x}_k)\|^2. \end{aligned} \tag{B.13}$$

Combine (B.11), (B.12) and (B.13).

$$\begin{aligned} f(y_k) - f(x^\star) \leq \frac{L}{\alpha_k} \big(f(\tilde{x}_k) - f(y_k) \big) &+ \left(\frac{1-\tau_k}{\tau_k} - \frac{L}{\alpha_k} \right) \langle \nabla f(y_k), \tilde{x}_k - y_k \rangle \\ &+ \left(\frac{1}{\alpha_k} - \frac{1-\tau_k}{L\tau_k} \right) \langle \nabla f(y_k), \nabla f(\tilde{x}_k) \rangle \\ &+ \frac{\alpha_k}{2} \left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k} \left[\|z_{k+1} - x^\star\|^2 \right] \right) \\ &- \frac{1}{2L} \|\nabla f(y_k)\|^2 - \frac{1}{2\alpha_k} \|\nabla f(\tilde{x}_k)\|^2. \end{aligned}$$

Substitute the choice $\alpha_k = \frac{L\tau_k}{1-\tau_k}$.

$$\begin{aligned}\frac{1-\tau_k}{\tau_k^2}\big(f(y_k)-f(x^\star)\big) \le{}& \frac{(1-\tau_k)^2}{\tau_k^2}\big(f(\tilde{x}_k)-f(x^\star)\big)\\ &+\frac{L}{2}\left(\|z_k-x^\star\|^2-\mathbb{E}_{i_k}\left[\|z_{k+1}-x^\star\|^2\right]\right)\\ &-\frac{1-\tau_k}{2L\tau_k}\|\nabla f(y_k)\|^2-\frac{(1-\tau_k)^2}{2L\tau_k^2}\|\nabla f(\tilde{x}_k)\|^2. \end{aligned}\tag{B.14}$$

Note that by construction, $\mathbb{E}\big[f(\tilde{x}_{k+1})\big] = p_k\mathbb{E}\big[f(y_k)\big]+(1-p_k)\mathbb{E}\big[f(\tilde{x}_k)\big]$, and thus

$$\begin{aligned}\frac{1-\tau_k}{\tau_k^2 p_k}\mathbb{E}\big[f(\tilde{x}_{k+1})-f(x^\star)\big] \le{}& \frac{(1-\tau_k p_k)(1-\tau_k)}{\tau_k^2 p_k}\mathbb{E}\big[f(\tilde{x}_k)-f(x^\star)\big]\\ &+\frac{L}{2}\left(\mathbb{E}\big[\,\|z_k-x^\star\|^2\,\big]-\mathbb{E}\big[\,\|z_{k+1}-x^\star\|^2\,\big]\right)\\ &-\frac{1-\tau_k}{2L\tau_k}\mathbb{E}\big[\,\|\nabla f(y_k)\|^2\,\big]-\frac{(1-\tau_k)^2}{2L\tau_k^2}\mathbb{E}\big[\,\|\nabla f(\tilde{x}_k)\|^2\,\big].\end{aligned}$$

B.3.2 Proof of Theorem 6

It can be easily verified that under this choice ($p_k \equiv \frac{1}{n}, \tau_k = \frac{3}{k/n+6}$), for any $k \ge 0, n \ge 1$,

$$\frac{(1-\tau_{k+1}p_{k+1})(1-\tau_{k+1})}{\tau_{k+1}^2 p_{k+1}} \le \frac{1-\tau_k}{\tau_k^2 p_k}.$$

Then, using Proposition 5.1, after summing (3.4) from $k = 0,\ldots,K-1$, we obtain

$$\begin{aligned}&\frac{n(1-\tau_{K-1})}{\tau_{K-1}^2}\mathbb{E}\big[f(\tilde{x}_K)-f(x^\star)\big]+\frac{L}{2}\mathbb{E}\big[\,\|z_K-x^\star\|^2\,\big]+\sum_{k=0}^{K-1}\frac{(1-\tau_k)^2}{2L\tau_k^2}\mathbb{E}\big[\,\|\nabla f(\tilde{x}_k)\|^2\,\big]\\ &\le (2n-1)\big(f(x_0)-f(x^\star)\big)+\frac{L}{2}\|x_0-x^\star\|^2.\end{aligned}$$

Note that $\tau_k \le \frac{1}{2}, \forall k$. We have the following two consequences of the above inequality.

$$\mathbb{E}\big[f(\tilde{x}_K)\big] - f(x^\star) \le \tau_{K-1}^2 \left(4\big(f(x_0) - f(x^\star)\big) + \frac{L}{n}\left\|x_0 - x^\star\right\|^2\right),$$

$$\begin{aligned}\mathbb{E}\big[\left\|\nabla f(x_{\text{out}})\right\|^2\big] &= \frac{1}{\sum_{k=0}^{K-1}\tau_k^{-2}}\sum_{k=0}^{K-1}\frac{1}{\tau_k^2}\mathbb{E}\big[\left\|\nabla f(\tilde{x}_k)\right\|^2\big] \\ &\le \frac{16nL\big(f(x_0) - f(x^\star)\big) + 4L^2\left\|x_0 - x^\star\right\|^2}{\sum_{k=0}^{K-1}\tau_k^{-2}}.\end{aligned}$$

Substituting the parameter choice, we obtain

$$\mathbb{E}\big[f(\tilde{x}_K)\big] - f(x^\star) \le \frac{36n^2\big(f(x_0) - f(x^\star)\big) + 9nL\left\|x_0 - x^\star\right\|^2}{(K + 6n - 1)^2} = \epsilon_f,$$

$$\mathbb{E}\big[\left\|\nabla f(x_{\text{out}})\right\|^2\big] \le \frac{144nL\big(f(x_0) - f(x^\star)\big) + 36L^2\left\|x_0 - x^\star\right\|^2}{\sum_{k=0}^{K-1}\left(\frac{k}{n} + 6\right)^2}.$$

Note that

$$\sum_{k=0}^{K-1}\left(\frac{k}{n} + 6\right)^2 \ge \int_0^K \left(\frac{x-1}{n} + 6\right)^2 dx = \frac{(K + 6n - 1)^3 - (6n - 1)^3}{3n^2}.$$

Thus,

$$\begin{aligned}\mathbb{E}\big[\left\|\nabla f(x_{\text{out}})\right\|\big]^2 &\le \mathbb{E}\big[\left\|\nabla f(x_{\text{out}})\right\|^2\big] \\ &\le \frac{432n^3L\big(f(x_0) - f(x^\star)\big) + 108n^2L^2\left\|x_0 - x^\star\right\|^2}{(K + 6n - 1)^3 - (6n - 1)^3} = \epsilon_g^2.\end{aligned}$$

Since the expected iteration cost of Algorithm 6 is $\mathbb{E}\big[\#\text{grad}_k\big] = p_k(n+2) + (1 - p_k)2 = 3$, to guarantee $\mathbb{E}\big[\left\|\nabla f(x_{\text{out}})\right\|\big] \le \epsilon_g$ and $\mathbb{E}\big[f(\tilde{x}_K)\big] - f(x^\star) \le \epsilon_f$, the total complexities are $O\Big(\frac{n(L(f(x_0)-f(x^\star)))^{1/3}}{\epsilon_g^{2/3}} + \frac{(nLR_0)^{2/3}}{\epsilon_g^{2/3}}\Big)$ and $O\Big(n\sqrt{\frac{f(x_0)-f(x^\star)}{\epsilon_f}} + \frac{\sqrt{nL}R_0}{\sqrt{\epsilon_f}}\Big)$, respectively.

B.3.3 Proof of Theorem 7

First, it can be verified that for any $k \ge 0, n \ge 1$, the following inequality holds.

$$\frac{(1 - \tau_{k+1}p_{k+1})(1 - \tau_{k+1})}{\tau_{k+1}^2 p_{k+1}} \le \frac{1 - \tau_k}{\tau_k^2 p_k}.$$

The verification can be done by considering the two cases: (i) $k+8<6n$, where $p_k=\frac{6}{k+8}, \tau_k=\frac{1}{2}$, (ii) $k+8\geq 6n$, in which $p_k=\frac{1}{n}, \tau_k=\frac{3n}{k+8}$.

Then, using Proposition 5.1, after summing (3.4) from $k=0,\dots,K-1$, we obtain

$$\frac{1-\tau_{K-1}}{\tau_{K-1}^2 p_{K-1}}\mathbb{E}\big[f(\tilde{x}_K)-f(x^\star)\big]+\frac{L}{2}\mathbb{E}\big[\left\|z_K-x^\star\right\|^2\big]+\sum_{k=0}^{K-1}\frac{(1-\tau_k)^2}{2L\tau_k^2}\mathbb{E}\big[\left\|\nabla f(\tilde{x}_k)\right\|^2\big]$$
$$\leq \frac{5}{3}\big(f(x_0)-f(x^\star)\big)+\frac{L}{2}\left\|x_0-x^\star\right\|^2\leq\frac{4}{3}LR_0^2.$$

Note that $\tau_k\leq\frac{1}{2},\forall k$. We can conclude the following two consequences.

$$\mathbb{E}\big[f(\tilde{x}_K)\big]-f(x^\star)\leq\frac{8}{3}\tau_{K-1}^2 p_{K-1}LR_0^2, \tag{B.15}$$

$$\mathbb{E}\big[\left\|\nabla f(x_{\text{out}})\right\|^2\big]=\frac{1}{\sum_{k=0}^{K-1}\tau_k^{-2}}\sum_{k=0}^{K-1}\frac{1}{\tau_k^2}\mathbb{E}\big[\left\|\nabla f(\tilde{x}_k)\right\|^2\big]\leq\frac{32L^2R_0^2}{3\sum_{k=0}^{K-1}\tau_k^{-2}}. \tag{B.16}$$

Now we consider two stages.

Stage I (low accuracy stage): $K+8\leq 6n$. In this stage, let the accuracies be $\epsilon_g^2=\frac{8L^2R_0^2}{3K}\geq\frac{8L^2R_0^2}{3(6n-8)}$ and $\epsilon_f=\frac{4LR_0^2}{K+7}\geq\frac{4LR_0^2}{6n-1}$. By substituting the parameter choice, (B.15) and (B.16) can be written as

$$\mathbb{E}\big[f(\tilde{x}_K)\big]-f(x^\star)\leq\frac{4LR_0^2}{K+7}=\epsilon_f,$$
$$\mathbb{E}\big[\left\|\nabla f(x_{\text{out}})\right\|\big]^2\leq\mathbb{E}\big[\left\|\nabla f(x_{\text{out}})\right\|^2\big]\leq\frac{8L^2R_0^2}{3K}=\epsilon_g^2.$$

Note that the expected iteration cost of Algorithm 6 is $\mathbb{E}\big[\#\text{grad}_k\big]=p_k(n+2)+(1-p_k)2=np_k+2$, and thus the total complexity in this stage is

$$\sum_{k=0}^{K-1}\mathbb{E}\big[\#\text{grad}_k\big]=n\sum_{k=0}^{K-1}\frac{6}{k+8}+2K\leq 6n\log{(K+7)}+12n=O(n\log K).$$

Thus, the expected complexities in this stage are $O(n\log\frac{LR_0}{\epsilon_g})$ and $O(n\log\frac{LR_0^2}{\epsilon_f})$, respectively.

Stage II (high accuracy stage): $K+8>6n$. In this stage, Algorithm 6 proceeds to find highly accurate solutions (i.e., $\epsilon_g^2<\frac{8L^2R_0^2}{3(6n-8)}$ and $\epsilon_f<\frac{4LR_0^2}{6n-1}$).

Substituting the parameter choice, we can write (B.15) and (B.16) as

$$\mathbb{E}\big[f(\tilde{x}_K)\big] - f(x^\star) \le \frac{24nLR_0^2}{(K+7)^2} = \epsilon_f, \tag{B.17}$$

$$\begin{aligned}\mathbb{E}\big[\,\|\nabla f(x_{\text{out}})\|^2\,\big] &\le \frac{32L^2R_0^2}{3\left(24n - 28 + \sum_{k=6n-7}^{K-1}\tau_k^{-2}\right)} \\ &\overset{(\star)}{\le} \frac{288n^2L^2R_0^2}{(K+7)^3 + 432n^3 - 756n^2} = \epsilon_g^2,\end{aligned} \tag{B.18}$$

where $(\star)$ follows from

$$\sum_{k=6n-7}^{K-1}\tau_k^{-2} = \frac{1}{9n^2}\sum_{k=6n-7}^{K-1}(k+8)^2 \ge \frac{1}{9n^2}\int_{6n-7}^{K}(x+7)^2dx = \frac{(K+7)^3}{27n^2} - 8n.$$

Then, we count the expected complexity in this stage.

$$\sum_{k=0}^{K-1}\mathbb{E}\big[\#\text{grad}_k\big] = n\left(\sum_{k=0}^{6n-8}\frac{6}{k+8} + \sum_{k=6n-7}^{K-1}\frac{1}{n}\right) + 2K \le 6n\log(6n) + 3K - 6n + 7.$$

Finally, combining with (B.17) and (B.18), we can conclude that the total expected complexities in this stage are $O\Big(n\log n + \frac{(nLR_0)^{2/3}}{\epsilon_g^{2/3}}\Big)$ and $O\Big(n\log n + \frac{\sqrt{nL}R_0}{\sqrt{\epsilon_f}}\Big)$, respectively.

B.3.4 Proof of Theorem 8

We start at inequality (B.14) in the proof of Proposition 5.1, which is the consequence of one iteration k in Algorithm 6. Due to the constant choice of $\tau_k \equiv \tau$, we have

$$\begin{aligned}f(y_k) - f(x^\star) \le (1-\tau)\big(f(\tilde{x}_k) - f(x^\star)\big) + \frac{L\tau^2}{2(1-\tau)}\left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k}\left[\|z_{k+1} - x^\star\|^2\right]\right) \\ - \frac{\tau}{2L}\|\nabla f(y_k)\|^2 - \frac{1-\tau}{2L}\|\nabla f(\tilde{x}_k)\|^2.\end{aligned}$$

Since we fix $p_k \equiv p$ as a constant and terminate Algorithm 6 at the first time $\tilde{x}_{k+1} = y_k$ (denoted as the iteration N), it is clear that the random variable N follows the geometric distribution with parameter p, i.e., for $k = 0, 1, 2, \ldots$, $\text{Prob}\{N = k\} =$

$(1-p)^k p$. Moreover, since we have $\tilde{x}_N = \tilde{x}_{N-1} = \cdots = \tilde{x}_0 = x_0$, using the above inequality at iteration N, we obtain

$$\begin{aligned}
&\mathbb{E}\big[f(\tilde{x}_{N+1})\big] - f(x^\star) \\
&\leq (1-\tau)\big(f(x_0) - f(x^\star)\big) + \frac{L\tau^2}{2(1-\tau)}\left(\mathbb{E}\big[\,\|z_N - x^\star\|^2 - \|z_{N+1} - x^\star\|^2\,\big]\right) \\
&\quad - \frac{\tau}{2L}\mathbb{E}\big[\,\|\nabla f(\tilde{x}_{N+1})\|^2\,\big] - \frac{1-\tau}{2L}\|\nabla f(x_0)\|^2 \\
&\overset{(\star)}{=} (1-\tau)\big(f(x_0) - f(x^\star)\big) + \frac{L\tau^2 p}{2(1-\tau)}\left(\|x_0 - x^\star\|^2 - \mathbb{E}\big[\,\|z_{N+1} - x^\star\|^2\,\big]\right) \\
&\quad - \frac{\tau}{2L}\mathbb{E}\big[\,\|\nabla f(\tilde{x}_{N+1})\|^2\,\big] - \frac{1-\tau}{2L}\|\nabla f(x_0)\|^2,
\end{aligned}$$

where $(\star)$ follows from

$$\begin{aligned}
\mathbb{E}\big[\,\|z_{N+1} - x^\star\|^2\,\big] &= \frac{1}{1-p}\left(\sum_{k=0}^{\infty}(1-p)^k p\,\mathbb{E}\big[\,\|z_k - x^\star\|^2\,\big] - p\,\|z_0 - x^\star\|^2\right) \\
&= \frac{1}{1-p}\left(\mathbb{E}\big[\,\|z_N - x^\star\|^2\,\big] - p\,\|z_0 - x^\star\|^2\right).
\end{aligned}$$

Thus, we can conclude that

$$\mathbb{E}\big[f(\tilde{x}_{N+1})\big] - f(x^\star) + \frac{\tau}{2L}\mathbb{E}\big[\,\|\nabla f(\tilde{x}_{N+1})\|^2\,\big] \leq \frac{L}{2}\left(1 - \tau + \frac{\tau^2 p}{1-\tau}\right)R_0^2.$$

Note that $\mathbb{E}\big[N\big] = \frac{1-p}{p}$ and the total expected oracle complexity is $n + 2(\mathbb{E}\big[N\big] + 1) = n + \frac{2}{p}$. We choose $p = \frac{1}{n}$, which leads to an $O(n)$ expected complexity. And we choose τ by minimizing the ratio $\left(1 - \tau + \frac{\tau^2 p}{1-\tau}\right)$ wrt τ. This gives $\tau = 1 - \frac{1}{\sqrt{n+1}} \geq \frac{1}{4}$ and

$$\mathbb{E}\big[f(\tilde{x}_{N+1})\big] - f(x^\star) + \frac{1}{8L}\mathbb{E}\big[\,\|\nabla f(\tilde{x}_{N+1})\|^2\,\big] \leq \frac{LR_0^2}{\sqrt{n+1}+1}.$$

B.4 Proofs of Section 3.4

We analyze Algorithm 7 following the shifting methodology in Chapter 2, which explores the tight interpolation condition (2.2) and leads to a simple and clean proof.

Note that after the regularization at Step 2, each $f_i^{\delta_t}$ is $(L+\delta_t)$-smooth and δ_t-strongly convex. We denote $x^\star_{\delta_t}$ as the unique minimizer of $\min_x f^{\delta_t}(x)$. We define a shifted version of this problem: $\min_x h^{\delta_t}(x) = \frac{1}{n}\sum_{i=1}^n h_i^{\delta_t}(x)$, where

$$h_i^{\delta_t}(x) = f_i^{\delta_t}(x) - f_i^{\delta_t}(x^\star_{\delta_t}) - \left\langle \nabla f_i^{\delta_t}(x^\star_{\delta_t}), x - x^\star_{\delta_t} \right\rangle - \frac{\delta_t}{2}\left\| x - x^\star_{\delta_t} \right\|^2, \forall i.$$

It can be easily verified that each $h_i^{\delta_t}$ is L-smooth and convex. Note that $h_i^{\delta_t}(x^\star_{\delta_t}) = h^{\delta_t}(x^\star_{\delta_t}) = 0$ and $\nabla h_i^{\delta_t}(x^\star_{\delta_t}) = \nabla h^{\delta_t}(x^\star_{\delta_t}) = \mathbf{0}$, which means that h^{δ_t} and f^{δ_t} share the same minimizer $x^\star_{\delta_t}$.

Then, conceptually, we attempts to solve the shifted problem using a shifted SVRG gradient estimator: $\mathcal{H}_k^{\delta_t} \triangleq \nabla h_{i_k}^{\delta_t}(y_k) - \nabla h_{i_k}^{\delta_t}(\tilde{x}_k) + \nabla h^{\delta_t}(\tilde{x}_k)$ from Chapter 2. Since the relation $\mathcal{H}_k^{\delta_t} = \mathcal{G}_k^{\delta_t} - \delta_t(y_k - x^\star_{\delta_t})$ holds, we can use Lemma 1 in Chapter 2 as an instantiation of the shifted gradient oracle.

B.4.1 Technical Lemma

Lemma 5 (The regularization technique [106]). *For an L-smooth and convex function f and $\delta > 0$, defining $f^\delta(x) = f(x) + \frac{\delta}{2}\|x - x_0\|^2, \forall x$ and denoting $x^\star_\delta$ as the unique minimizer of f^δ, we have*

(i) f^δ is $(L+\delta)$-smooth and δ-strongly convex.
(ii) $f^\delta(x_0) - f^\delta(x^\star_\delta) \le f(x_0) - f(x^\star)$.
(iii) $\|x_0 - x^\star_\delta\|^2 \le \|x_0 - x^\star\|^2, \forall x^\star \in \mathcal{X}^\star$.
(iv) $\|x_0 - x^\star_\delta\|^2 \le \frac{2}{\delta}\big(f(x_0) - f(x^\star)\big)$.

Proof. *(i)* can be easily checked by the definition of L-smoothness and strong convexity. *(ii)* follows from $f^\delta(x_0) = f(x_0)$ and $f^\delta(x^\star_\delta) \ge f(x^\star_\delta) \ge f(x^\star)$. For *(iii)*, using the strong convexity of f^δ at $(x^\star, x^\star_\delta), \forall x^\star \in \mathcal{X}^\star$, we obtain

$$\begin{aligned} & f^\delta(x^\star) - f^\delta(x^\star_\delta) \ge \frac{\delta}{2}\|x^\star - x^\star_\delta\|^2 \\ \Rightarrow\ & f(x^\star) + \frac{\delta}{2}\|x^\star - x_0\|^2 - f(x^\star_\delta) - \frac{\delta}{2}\|x^\star_\delta - x_0\|^2 \ge \frac{\delta}{2}\|x^\star - x^\star_\delta\|^2 \\ \Rightarrow\ & \frac{\delta}{2}\|x_0 - x^\star\|^2 - \big(f(x^\star_\delta) - f(x^\star)\big) \ge \frac{\delta}{2}\|x_0 - x^\star_\delta\|^2 + \frac{\delta}{2}\|x^\star - x^\star_\delta\|^2. \end{aligned}$$

Then *(iii)* follows from the non-negativeness of $f(x^\star_\delta) - f(x^\star)$ and $\|x^\star - x^\star_\delta\|^2$. For *(iv)*, using the strong convexity of f^δ at $(x_0, x^\star_\delta)$ and *(ii)*, we have $\|x_0 - x^\star_\delta\|^2 \le \frac{2}{\delta}\big(f^\delta(x_0) - f^\delta(x^\star_\delta)\big) \le \frac{2}{\delta}\big(f(x_0) - f(x^\star)\big)$. □

B.4.2 Proof of Proposition 8.1

Denoting $\kappa_t = \frac{L+\delta_t}{\delta_t}$, we can write the equation $\left(1 - \frac{p(\alpha+\delta_t)}{\alpha+L+\delta_t}\right)\left(1 + \frac{\delta_t}{\alpha}\right)^2 = 1$ as

$$s\left(\frac{\alpha}{\delta_t}\right) \triangleq \left(\frac{\alpha}{\delta_t}\right)^3 - (2n-3)\left(\frac{\alpha}{\delta_t}\right)^2 - (2n\kappa_t + n - 3)\left(\frac{\alpha}{\delta_t}\right) - n\kappa_t + 1 = 0.$$

It can be verified that $s(2n + 2\sqrt{n\kappa_t}) > 0$ for any $n \geq 1, \kappa_t > 1$. Since $s(0) < 0$ and $s(\frac{\alpha}{\delta_t}) \to \infty$ as $\frac{\alpha}{\delta_t} \to \infty$, the unique positive root satisfies $\frac{\alpha}{\delta_t} \leq 2n + 2\sqrt{n\kappa_t} = O(n + \sqrt{n\kappa_t})$.

To bound C_{IDC} and C_{IFC}, it suffices to note that

$$\frac{\frac{\alpha^2}{\delta_t^2}p}{\frac{L}{\delta_t} + (1-p)(\frac{\alpha}{\delta_t} + 1)} \overset{(a)}{=} \frac{(\frac{\alpha}{\delta_t} + 1)^2}{n(\frac{\alpha}{\delta_t} + \kappa_t)} \overset{(b)}{\leq} \frac{(2n + 2\sqrt{n\kappa_t} + 1)^2}{n(2n + 2\sqrt{n\kappa_t} + \kappa_t)} \leq 6,$$

where (a) uses the cubic equation and (b) holds because $\frac{x+1}{x+\kappa_t}$ increases monotonically as x increases. Then,

$$\begin{aligned} C_{\text{IDC}} &\leq L^2 + 6L\delta_t = O\big((L+\delta_t)^2\big), \\ C_{\text{IFC}} &\leq 14L = O(L). \end{aligned}$$

B.4.3 Proof of Proposition 8.2

Using the interpolation condition (2.2) of h^{δ_t} at $(x^\star_{\delta_t}, y_k)$, we obtain

$$\begin{aligned} h^{\delta_t}(y_k) &\leq \left\langle \nabla h^{\delta_t}(y_k), y_k - x^\star_{\delta_t} \right\rangle - \frac{1}{2L}\left\|\nabla h^{\delta_t}(y_k)\right\|^2 \\ &\overset{(a)}{\leq} \frac{1-\tau_x}{\tau_x}\left\langle \nabla h^{\delta_t}(y_k), \tilde{x}_k - y_k \right\rangle + \frac{\tau_z}{\tau_x}\left\langle \nabla h^{\delta_t}(y_k), \delta_t(\tilde{x}_k - z_k) - \nabla f^{\delta_t}(\tilde{x}_k) \right\rangle \\ &\quad + \left\langle \nabla h^{\delta_t}(y_k), z_k - x^\star_{\delta_t} \right\rangle - \frac{1}{2L}\left\|\nabla h^{\delta_t}(y_k)\right\|^2 \\ &\overset{(b)}{=} \frac{1-\tau_x}{\tau_x}\left\langle \nabla h^{\delta_t}(y_k), \tilde{x}_k - y_k \right\rangle - \frac{\tau_z}{\tau_x}\left\langle \nabla h^{\delta_t}(y_k), \nabla h^{\delta_t}(\tilde{x}_k) \right\rangle \\ &\quad + \left(1 - \frac{\delta_t\tau_z}{\tau_x}\right)\left\langle \nabla h^{\delta_t}(y_k), z_k - x^\star_{\delta_t} \right\rangle - \frac{1}{2L}\left\|\nabla h^{\delta_t}(y_k)\right\|^2, \end{aligned}$$

where (a) follows from the construction

$$y_k = \tau_x z_k + (1-\tau_x)\tilde{x}_k + \tau_z\left(\delta_t(\tilde{x}_k - z_k) - \nabla f^{\delta_t}(\tilde{x}_k)\right),$$

and (b) uses that $\delta_t(\tilde{x}_k - z_k) - \nabla f^{\delta_t}(\tilde{x}_k) = \delta_t(x^\star_{\delta_t} - z_k) - \nabla h^{\delta_t}(\tilde{x}_k)$.

Using Lemma 1 with $\mathcal{H}_y = \mathcal{H}_k^{\delta_t}, \mathcal{G}_y = \mathcal{G}_k^{\delta_t}, z^+ = z_{k+1}, x^\star = x^\star_{\delta_t}$ and taking the expectation (note that $\mathbb{E}_{i_k}\left[\mathcal{H}_k^{\delta_t}\right] = \nabla h^{\delta_t}(y_k)$), we can conclude that

$$\begin{aligned} h^{\delta_t}(y_k) \le& \frac{1-\tau_x}{\tau_x}\left\langle \nabla h^{\delta_t}(y_k), \tilde{x}_k - y_k\right\rangle - \frac{\tau_z}{\tau_x}\left\langle \nabla h^{\delta_t}(y_k), \nabla h^{\delta_t}(\tilde{x}_k)\right\rangle - \frac{1}{2L}\left\|\nabla h^{\delta_t}(y_k)\right\|^2 \\ &+ \left(1 - \frac{\delta_t \tau_z}{\tau_x}\right)\frac{\alpha}{2}\left(\left\|z_k - x^\star_{\delta_t}\right\|^2 - \left(1+\frac{\delta_t}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\left\|z_{k+1} - x^\star_{\delta_t}\right\|^2\right]\right) \\ &+ \left(1 - \frac{\delta_t \tau_z}{\tau_x}\right)\frac{1}{2\alpha}\mathbb{E}_{i_k}\left[\left\|\mathcal{H}_k^{\delta_t}\right\|^2\right]. \end{aligned}$$

To bound the shifted moment, we use the interpolation condition (2.2) of $h_{i_k}^{\delta_t}$ at $(\tilde{x}_k, y_k)$, that is

$$\begin{aligned} &\mathbb{E}_{i_k}\left[\left\|\mathcal{H}_k^{\delta_t}\right\|^2\right] \\ =& \mathbb{E}_{i_k}\left[\left\|\nabla h_{i_k}^{\delta_t}(y_k) - \nabla h_{i_k}^{\delta_t}(\tilde{x}_k)\right\|^2\right] + 2\left\langle \nabla h^{\delta_t}(y_k), \nabla h^{\delta_t}(\tilde{x}_k)\right\rangle - \left\|\nabla h^{\delta_t}(\tilde{x}_k)\right\|^2 \\ \le& 2L\big(h^{\delta_t}(\tilde{x}_k) - h^{\delta_t}(y_k) - \left\langle \nabla h^{\delta_t}(y_k), \tilde{x}_k - y_k\right\rangle\big) \\ &+ 2\left\langle \nabla h^{\delta_t}(y_k), \nabla h^{\delta_t}(\tilde{x}_k)\right\rangle - \left\|\nabla h^{\delta_t}(\tilde{x}_k)\right\|^2. \end{aligned}$$

Re-arrange the terms.

$$\begin{aligned} h^{\delta_t}(y_k) \le& \left(1 - \frac{\delta_t \tau_z}{\tau_x}\right)\frac{L}{\alpha}\big(h^{\delta_t}(\tilde{x}_k) - h^{\delta_t}(y_k)\big) \\ &+ \left(\frac{1-\tau_x}{\tau_x} - \left(1 - \frac{\delta_t \tau_z}{\tau_x}\right)\frac{L}{\alpha}\right)\left\langle \nabla h^{\delta_t}(y_k), \tilde{x}_k - y_k\right\rangle \\ &+ \left(1 - \frac{\delta_t \tau_z}{\tau_x}\right)\frac{\alpha}{2}\left(\left\|z_k - x^\star_{\delta_t}\right\|^2 - \left(1+\frac{\delta_t}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\left\|z_{k+1} - x^\star_{\delta_t}\right\|^2\right]\right) \\ &+ \left(\frac{1}{\alpha} - \frac{\delta_t \tau_z}{\alpha \tau_x} - \frac{\tau_z}{\tau_x}\right)\left\langle \nabla h^{\delta_t}(y_k), \nabla h^{\delta_t}(\tilde{x}_k)\right\rangle - \frac{1}{2L}\left\|\nabla h^{\delta_t}(y_k)\right\|^2 \\ &- \left(\frac{1}{2\alpha} - \frac{\delta_t \tau_z}{2\alpha\tau_x}\right)\left\|\nabla h^{\delta_t}(\tilde{x}_k)\right\|^2. \end{aligned}$$

The choice of τ_z in Proposition 8.1 ensures that $\frac{1-\tau_x}{\tau_x} = \left(1 - \frac{\delta_t\tau_z}{\tau_x}\right)\frac{L}{\alpha}$, which leads to

$$\begin{aligned}
&h^{\delta_t}(y_k) \\
&\leq (1-\tau_x)h^{\delta_t}(\tilde{x}_k) + \frac{\alpha^2(1-\tau_x)}{2L}\left(\left\|z_k - x^\star_{\delta_t}\right\|^2 - \left(1+\frac{\delta_t}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\left\|z_{k+1} - x^\star_{\delta_t}\right\|^2\right]\right) \\
&\quad + \frac{\alpha+\delta_t-(\alpha+L+\delta_t)\tau_x}{L\delta_t}\left\langle \nabla h^{\delta_t}(y_k), \nabla h^{\delta_t}(\tilde{x}_k)\right\rangle - \frac{\tau_x}{2L}\left\|\nabla h^{\delta_t}(y_k)\right\|^2 \\
&\quad - \frac{1-\tau_x}{2L}\left\|\nabla h^{\delta_t}(\tilde{x}_k)\right\|^2.
\end{aligned}$$

Substitute the choice $\tau_x = \frac{\alpha+\delta_t}{\alpha+L+\delta_t}$.

$$\begin{aligned}
h^{\delta_t}(y_k) \leq{}& \frac{L}{\alpha+L+\delta_t}h^{\delta_t}(\tilde{x}_k) \\
&+ \frac{\alpha^2}{2(\alpha+L+\delta_t)}\left(\left\|z_k - x^\star_{\delta_t}\right\|^2 - \left(1+\frac{\delta_t}{\alpha}\right)^2 \mathbb{E}_{i_k}\left[\left\|z_{k+1} - x^\star_{\delta_t}\right\|^2\right]\right).
\end{aligned}$$

Note that by construction, $\mathbb{E}\big[h^{\delta_t}(\tilde{x}_{k+1})\big] = p\mathbb{E}\big[h^{\delta_t}(y_k)\big] + (1-p)\mathbb{E}\big[h^{\delta_t}(\tilde{x}_k)\big]$, and thus

$$\begin{aligned}
\mathbb{E}\big[h^{\delta_t}(\tilde{x}_{k+1})\big] \leq{}& \left(1 - \frac{p(\alpha+\delta_t)}{\alpha+L+\delta_t}\right)\mathbb{E}\big[h^{\delta_t}(\tilde{x}_k)\big] \\
&+ \frac{\alpha^2 p}{2(\alpha+L+\delta_t)}\left(\mathbb{E}\big[\left\|z_k - x^\star_{\delta_t}\right\|^2\big] - \left(1+\frac{\delta_t}{\alpha}\right)^2 \mathbb{E}\big[\left\|z_{k+1} - x^\star_{\delta_t}\right\|^2\big]\right).
\end{aligned}$$

Since α is chosen as the positive root of

$$\left(1 - \frac{p(\alpha+\delta_t)}{\alpha+L+\delta_t}\right)\left(1+\frac{\delta_t}{\alpha}\right)^2 = 1,$$

defining the potential function

$$T_k \triangleq \mathbb{E}\big[h^{\delta_t}(\tilde{x}_k)\big] + \frac{\alpha^2 p}{2\big(L + (1-p)(\alpha+\delta_t)\big)}\mathbb{E}\big[\left\|z_k - x^\star_{\delta_t}\right\|^2\big], \tag{B.19}$$

we have $T_{k+1} \leq \left(1+\frac{\delta_t}{\alpha}\right)^{-2} T_k$.

Thus, at iteration k, the following holds,

$$\begin{aligned}
&\mathbb{E}\big[h^{\delta_t}(\tilde{x}_k)\big] \\
&\leq \left(1+\frac{\delta_t}{\alpha}\right)^{-2k}\left(h^{\delta_t}(x_0)+\frac{\alpha^2 p}{2\big(L+(1-p)(\alpha+\delta_t)\big)}\left\|x_0-x^\star_{\delta_t}\right\|^2\right) \\
&\leq \left(1+\frac{\delta_t}{\alpha}\right)^{-2k}\left(f^{\delta_t}(x_0)-f^{\delta_t}(x^\star_{\delta_t})+\frac{\alpha^2 p}{2\big(L+(1-p)(\alpha+\delta_t)\big)}\left\|x_0-x^\star_{\delta_t}\right\|^2\right) \\
&\overset{(\star)}{\leq} \left(1+\frac{\delta_t}{\alpha}\right)^{-2k}\left(f(x_0)-f(x^\star)+\frac{\alpha^2 p}{2\big(L+(1-p)(\alpha+\delta_t)\big)}\left\|x_0-x^\star_{\delta_t}\right\|^2\right),
\end{aligned}$$

where $(\star)$ uses Lemma 5 *(ii)*.

Note that using the interpolation condition (2.2), we have

$$\begin{aligned}
\mathbb{E}\big[h^{\delta_t}(\tilde{x}_k)\big] &\geq \frac{1}{2L}\mathbb{E}\big[\left\|\nabla h^{\delta_t}(\tilde{x}_k)\right\|^2\big] \\
&= \frac{1}{2L}\mathbb{E}\big[\left\|\nabla f^{\delta_t}(\tilde{x}_k)-\delta_t(\tilde{x}_k-x^\star_{\delta_t})\right\|^2\big] \\
&= \frac{1}{2L}\mathbb{E}\big[\left\|\nabla f(\tilde{x}_k)+\delta_t(\tilde{x}_k-x_0)-\delta_t(\tilde{x}_k-x^\star_{\delta_t})\right\|^2\big] \\
&= \frac{1}{2L}\mathbb{E}\big[\left\|\nabla f(\tilde{x}_k)-\delta_t(x_0-x^\star_{\delta_t})\right\|^2\big] \\
&\geq \frac{1}{2L}\mathbb{E}\big[\left\|\nabla f(\tilde{x}_k)-\delta_t(x_0-x^\star_{\delta_t})\right\|\big]^2.
\end{aligned}$$

Finally, we conclude that

$$\begin{aligned}
&\mathbb{E}\big[\left\|\nabla f(\tilde{x}_k)\right\|\big] \\
&\leq \delta_t\left\|x_0-x^\star_{\delta_t}\right\| \\
&\quad+\left(1+\frac{\delta_t}{\alpha}\right)^{-k}\sqrt{2L\big(f(x_0)-f(x^\star)\big)+\frac{L\alpha^2 p}{L+(1-p)(\alpha+\delta_t)}\left\|x_0-x^\star_{\delta_t}\right\|^2}.
\end{aligned} \tag{B.20}$$

Under IDC: Invoking Lemma 5 *(iii)* to upper bound (B.20), we obtain that for any $x^\star \in \mathcal{X}^\star$,

$$\mathbb{E}\big[\left\|\nabla f(\tilde{x}_k)\right\|\big] \leq \left(\delta_t+\left(1+\frac{\delta_t}{\alpha}\right)^{-k}\sqrt{L^2+\frac{L\alpha^2 p}{L+(1-p)(\alpha+\delta_t)}}\right)\left\|x_0-x^\star\right\|.$$

Under IFC: Invoking Lemma 5 *(iv)* to upper bound (B.20), we can conclude that

$$\begin{aligned}&\mathbb{E}\big[\,\|\nabla f(\tilde{x}_k)\|\,\big]\\ &\le \left(\sqrt{2\delta_t} + \left(1+\frac{\delta_t}{\alpha}\right)^{-k}\sqrt{2L + \frac{2L\alpha^2 p}{\big(L+(1-p)(\alpha+\delta_t)\big)\delta_t}}\right)\sqrt{f(x_0)-f(x^\star)}.\end{aligned}$$

B.4.4 Proof of Theorem 9

(i) At outer iteration ℓ, if Algorithm 7 breaks the inner loop (Step 9) at iteration k, by construction, we have $(1+\frac{\delta_\ell}{\alpha})^{-k}\sqrt{C_{\text{IDC}}} \le \delta_\ell$. Then, from Proposition 8.2 *(i)*,

$$\mathbb{E}\big[\,\|\nabla f(\tilde{x}_k)\|\,\big] \le 2\delta_\ell R_0 \overset{(\star)}{\le} \epsilon q,$$

where $(\star)$ uses $\delta_\ell \le \delta^\star_{\text{IDC}}$. By Markov's inequality, it holds that

$$\text{Prob}\left\{\|\nabla f(\tilde{x}_k)\| \ge \epsilon\right\} \le \frac{\mathbb{E}\big[\,\|\nabla f(\tilde{x}_k)\|\,\big]}{\epsilon} \le q,$$

which means that with probability at least $1-q$, Algorithm 7 terminates at iteration k (Step 8) before reaching Step 9.

(ii) Note that the expected gradient complexity of each inner iteration is $p(n+2)+(1-p)2 = np+2$. Then, for an inner loop that breaks at Step 9, its expected complexity is

$$\mathbb{E}\big[\#\text{grad}_t\big] \le (np+2)\left(\frac{\alpha}{\delta_t}+1\right)\log\frac{\sqrt{C_{\text{IDC}}}}{\delta_t}.$$

Substituting the choices in Proposition 8.1, we obtain

$$\mathbb{E}\big[\#\text{grad}_t\big] = O\left(\left(n+\sqrt{\frac{nL}{\delta_t}}\right)\log\frac{L+\delta_t}{\delta_t}\right).$$

Thus, the total expected complexity before Algorithm 7 terminates with high probability at outer iteration ℓ is at most (note that $\delta_t = \delta_0/\beta^t$)

$$\sum_{t=0}^{\ell}\mathbb{E}\big[\#\text{grad}_t\big] = O\left(\left(\ell n + \frac{1}{\sqrt{\beta}-1}\sqrt{\frac{nL\beta}{\delta_\ell}}\right)\log\frac{L+\delta_\ell}{\delta_\ell}\right).$$

Since outer iteration $\ell > 0$ is the first time $\delta_\ell \leq \delta^\star_{\mathrm{IDC}}$, we have $\delta_\ell \leq \delta^\star_{\mathrm{IDC}} \leq \delta_\ell \beta$. Moreover, noting that $\ell = O(\log \frac{\delta_0}{\delta_\ell})$ and $\delta_0 = L$, we can conclude that (omitting β)

$$\begin{aligned}\sum_{t=0}^{\ell} \mathbb{E}\left[\#\mathrm{grad}_t\right] &= O\left(\left(n \log \frac{\delta_0}{\delta_\ell} + \sqrt{\frac{nL}{\delta_\ell}}\right) \log \frac{L + \delta_\ell}{\delta_\ell}\right) \\ &= O\left(\left(n \log \frac{LR_0}{\epsilon q} + \sqrt{\frac{nLR_0}{\epsilon q}}\right) \log \frac{LR_0}{\epsilon q}\right).\end{aligned}$$

B.4.5 Proof of Theorem 10

(i) At outer iteration ℓ, if Algorithm 7 breaks the inner loop (Step 10) at iteration k, by construction, we have $(1 + \frac{\delta_\ell}{\alpha})^{-k}\sqrt{C_{\mathrm{IFC}}} \leq \sqrt{2\delta_\ell}$. Then, from Proposition 8.2 *(ii)*,

$$\mathbb{E}\left[\,\|\nabla f(\tilde{x}_k)\|\,\right] \leq \sqrt{8\delta_\ell \Delta_0} \overset{(\star)}{\leq} \epsilon q,$$

where $(\star)$ uses $\delta_\ell \leq \delta^\star_{\mathrm{IFC}}$. By Markov's inequality, it holds that

$$\mathrm{Prob}\left\{\|\nabla f(\tilde{x}_k)\| \geq \epsilon\right\} \leq \frac{\mathbb{E}\left[\,\|\nabla f(\tilde{x}_k)\|\,\right]}{\epsilon} \leq q,$$

which means that with probability at least $1-q$, Algorithm 7 terminates at iteration k (Step 8) before reaching Step 10.

(ii) Note that the expected gradient complexity of each inner iteration is $p(n+2) + (1-p)2 = np + 2$. Then, for an inner loop that breaks at Step 10, its expected complexity is

$$\mathbb{E}\left[\#\mathrm{grad}_t\right] \leq (np+2)\left(\frac{\alpha}{\delta_t} + 1\right) \log \sqrt{\frac{C_{\mathrm{IFC}}}{2\delta_t}}.$$

Substituting the choices in Proposition 8.1, we obtain

$$\mathbb{E}\left[\#\mathrm{grad}_t\right] = O\left(\left(n + \sqrt{\frac{nL}{\delta_t}}\right) \log \frac{L}{\delta_t}\right).$$

Thus, the total expected complexity before Algorithm 7 terminates with high probability at outer iteration ℓ is at most (note that $\delta_t = \delta_0/\beta^t$)

$$\sum_{t=0}^{\ell} \mathbb{E}\left[\#\mathrm{grad}_t\right] = O\left(\left(\ell n + \frac{1}{\sqrt{\beta} - 1}\sqrt{\frac{nL\beta}{\delta_\ell}}\right) \log \frac{L}{\delta_\ell}\right).$$

Since outer iteration $\ell > 0$ is the first time $\delta_\ell \le \delta^\star_{\text{IFC}}$, we have $\delta_\ell \le \delta^\star_{\text{IFC}} \le \delta_\ell \beta$. Moreover, noting that $\ell = O(\log \frac{\delta_0}{\delta_\ell})$ and $\delta_0 = L$, we can conclude that (omitting β)

$$\begin{aligned}\sum_{t=0}^{\ell} \mathbb{E}\big[\#\text{grad}_t\big] &= O\left(\left(n \log \frac{\delta_0}{\delta_\ell} + \sqrt{\frac{nL}{\delta_\ell}}\right) \log \frac{L}{\delta_\ell}\right) \\ &= O\left(\left(n \log \frac{\sqrt{L\Delta_0}}{\epsilon q} + \frac{\sqrt{nL\Delta_0}}{\epsilon q}\right) \log \frac{\sqrt{L\Delta_0}}{\epsilon q}\right).\end{aligned}$$

B.5 Katyusha + L2S

By applying AdaptReg on Katyusha, Allen-Zhu [4] showed that the scheme outputs a point x_{s_1} satisfying $\mathbb{E}\big[f(x_{s_1})\big] - f(x^\star) \le \epsilon_1$ in

$$O\left(n \log \frac{LR_0^2}{\epsilon_1} + \frac{\sqrt{nL}R_0}{\sqrt{\epsilon_1}}\right),$$

oracle calls for any $\epsilon_1 > 0$ (cf. Corollary 3.5 in [4]).

For L2S, Li et al. [88] proved that when using an n-dependent step size, its output x_a satisfies (cf. Corollary 3 in [88])

$$\mathbb{E}\big[\,\|\nabla f(x_a)\|\,\big]^2 \le \mathbb{E}\big[\,\|\nabla f(x_a)\|^2\,\big] = O\left(\frac{\sqrt{n}L\big(f(x_0) - f(x^\star)\big)}{T}\right),$$

after running T iterations.

We can combine these two rates following the ideas in [106]. Set $\epsilon_1 = O\big(\frac{T\epsilon^2}{\sqrt{nL}}\big)$ for some $\epsilon > 0$ and let the input x_0 of L2S be the output x_{s_1} of Katyusha. By chaining the above two results, we obtain the guarantee $\mathbb{E}\big[\,\|\nabla f(x_a)\|\,\big] = O(\epsilon)$ in oracle complexity

$$O\left(n + T + n \log \frac{n^{1/4}LR_0}{\sqrt{T}\epsilon} + \frac{n^{3/4}LR_0}{\sqrt{T}\epsilon}\right).$$

Minimizing the complexity by choosing $T = O\big(\frac{\sqrt{n}(LR_0)^{2/3}}{\epsilon^{2/3}}\big)$, we get the total oracle complexity

$$O\left(n \log \frac{LR_0}{\epsilon} + \frac{\sqrt{n}(LR_0)^{2/3}}{\epsilon^{2/3}}\right).$$

Appendix C

Appendix for Chapter 4

C.1 Proof of Lemma 3

Upper-bounding the variance term.

$$\begin{aligned}
&\mathbb{E}_i\left[\|\mathcal{G}_y - \nabla f(y)\|^2\right] \\
&\overset{(a)}{=} \mathbb{E}_i\left[\|\nabla f_i(y) - \nabla f_i(\tilde{x}) + D_i \nabla f(\tilde{x})\|^2\right] - \|\nabla f(y)\|^2 \\
&= \mathbb{E}_i\left[\|\nabla f_i(y) - \nabla f_i(\tilde{x})\|^2\right] + 2\mathbb{E}_i\left[\langle \nabla f_i(y) - \nabla f_i(\tilde{x}), D_i \nabla f(\tilde{x})\rangle\right] \\
&\quad + \mathbb{E}_i\left[\|D_i \nabla f(\tilde{x})\|^2\right] - \|\nabla f(y)\|^2 \\
&\overset{(b)}{=} \mathbb{E}_i\left[\|\nabla f_i(y) - \nabla f_i(\tilde{x})\|^2\right] + 2\mathbb{E}_i\left[\langle \nabla f_i(y) - \nabla f_i(\tilde{x}), D \nabla f(\tilde{x})\rangle\right] \\
&\quad + \mathbb{E}_i\left[\langle D \nabla f(\tilde{x}), D_i \nabla f(\tilde{x})\rangle\right] - \|\nabla f(y)\|^2 \\
&= \mathbb{E}_i\left[\|\nabla f_i(y) - \nabla f_i(\tilde{x})\|^2\right] + 2\langle \nabla f(y), D\nabla f(\tilde{x})\rangle - \langle \nabla f(\tilde{x}), D \nabla f(\tilde{x})\rangle - \|\nabla f(y)\|^2 \\
&\overset{(c)}{\leq} 2L\big(f(\tilde{x}) - f(y) - \langle \nabla f(y), \tilde{x} - y\rangle\big) + 2\langle \nabla f(y), D\nabla f(\tilde{x})\rangle - \langle \nabla f(\tilde{x}), D\nabla f(\tilde{x})\rangle \\
&\quad - \|\nabla f(y)\|^2,
\end{aligned}$$

where (a) uses the unbiasedness $\mathbb{E}_i[\mathcal{G}_y] = \nabla f(y)$, (b) follows from that $\nabla f_i(y) - \nabla f_i(\tilde{x})$ is supported on T_i and $P_i^2 = P_i$, and (c) uses the interpolation condition.

C.2 Proof of Theorem 11

We omit the superscript s in the one-epoch analysis for clarity. Using convexity, we have

$$\begin{aligned}
&f(y_k) - f(x^\star) \\
\leq\ & \langle \nabla f(y_k), y_k - x^\star \rangle \\
=\ & \langle \nabla f(y_k), y_k - z_k \rangle + \langle \nabla f(y_k), z_k - x^\star \rangle \\
\overset{(\star)}{\leq}\ & \frac{1-\vartheta}{\vartheta} \langle \nabla f(y_k), \tilde{x}_s - y_k \rangle - \frac{\varphi}{\vartheta} \langle \nabla f(y_k), D \nabla f(\tilde{x}_s) \rangle + \langle \nabla f(y_k), z_k - x^\star \rangle,
\end{aligned} \tag{C.1}$$

where $(\star)$ follows from the construction $y_k = \vartheta z_k + (1-\vartheta)\tilde{x}_s - \varphi D \nabla f(\tilde{x}_s)$.

Denote $\mathcal{G}_{y_k} = \nabla f_{i_k}(y_k) - \nabla f_{i_k}(\tilde{x}_s) + D_{i_k} \nabla f(\tilde{x}_s)$. Based on the updating rule: $z_{k+1} = z_k - \eta \cdot \mathcal{G}_{y_k}$, it holds that

$$\begin{aligned}
&\|z_{k+1} - x^\star\|^2 = \|z_k - x^\star - \eta \cdot \mathcal{G}_{y_k}\|^2 \\
\Rightarrow\ & \langle \mathcal{G}_{y_k}, z_k - x^\star \rangle = \frac{\eta}{2} \|\mathcal{G}_{y_k}\|^2 + \frac{1}{2\eta} \left(\|z_k - x^\star\|^2 - \|z_{k+1} - x^\star\|^2 \right) \\
\overset{(\star)}{\Rightarrow}\ & \langle \nabla f(y_k), z_k - x^\star \rangle = \frac{\eta}{2} \mathbb{E}_{i_k} \left[\|\mathcal{G}_{y_k}\|^2 \right] + \frac{1}{2\eta} \left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k} \left[\|z_{k+1} - x^\star\|^2 \right] \right) \\
\Rightarrow\ & \langle \nabla f(y_k), z_k - x^\star \rangle = \frac{\eta}{2} \mathbb{E}_{i_k} \left[\|\mathcal{G}_{y_k} - \nabla f(y_k)\|^2 \right] + \frac{\eta}{2} \|\nabla f(y_k)\|^2 \\
& \qquad + \frac{1}{2\eta} \left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k} \left[\|z_{k+1} - x^\star\|^2 \right] \right),
\end{aligned} \tag{C.2}$$

where $(\star)$ follows from taking the expectation with respect to sample i_k.

Combine (C.1), (C.2) and use the variance bound in Lemma 3.

$$\begin{aligned}
f(y_k) - f(x^\star) \leq\ & \eta L \big(f(\tilde{x}_s) - f(y_k) \big) + \left(\frac{1-\vartheta}{\vartheta} - \eta L \right) \langle \nabla f(y_k), \tilde{x}_s - y_k \rangle \\
& + \left(\eta - \frac{\varphi}{\vartheta} \right) \langle \nabla f(y_k), D \nabla f(\tilde{x}_s) \rangle - \frac{\eta}{2} \langle \nabla f(\tilde{x}_s), D \nabla f(\tilde{x}_s) \rangle \\
& + \frac{1}{2\eta} \left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k} \left[\|z_{k+1} - x^\star\|^2 \right] \right).
\end{aligned}$$

Substituting the choices $\eta = \frac{1-\vartheta}{L\vartheta}$ and $\varphi = \eta\vartheta = \frac{1-\vartheta}{L}$ and noting that $D \succ 0$, we obtain

$$\begin{aligned}
f(y_k) - f(x^\star) \leq\ & (1-\vartheta) \big(f(\tilde{x}_s) - f(x^\star) \big) \\
& + \frac{L\vartheta^2}{2(1-\vartheta)} \left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k} \left[\|z_{k+1} - x^\star\|^2 \right] \right).
\end{aligned}$$

Summing the above inequality from $k = 0$ to $m - 1$ and noting that $z_0^{s+1} = z_m^s$, we obtain

$$\begin{aligned}\mathbb{E}\big[f(\tilde{x}_{s+1}) - f(x^\star)\big] &= \frac{1}{m}\sum_{k=0}^{m-1}\mathbb{E}\big[f(y_k) - f(x^\star)\big] \\ &\leq (1-\vartheta)\mathbb{E}\big[f(\tilde{x}_s) - f(x^\star)\big] \\ &\quad + \frac{L\vartheta^2}{2m(1-\vartheta)}\Big(\mathbb{E}\big[\,\|z_0^s - x^\star\|^2\,\big] - \mathbb{E}\big[\,\|z_0^{s+1} - x^\star\|^2\,\big]\Big).\end{aligned}$$

Denoting $Q_s \triangleq \mathbb{E}\big[f(\tilde{x}_s) - f(x^\star)\big]$ and $P_s \triangleq \mathbb{E}\big[\,\|z_0^s - x^\star\|^2\,\big]$, we can write the above as

$$Q_{s+1} \leq \frac{1-\vartheta}{\vartheta}(Q_s - Q_{s+1}) + \frac{L\vartheta}{2m(1-\vartheta)}(P_s - P_{s+1}).$$

Summing this inequality from $s = 0$ to $S - 1$ and using Jensen's inequality, we have

$$\begin{aligned}\mathbb{E}\big[f(x_{r+1}) - f(x^\star)\big] &\leq \frac{1}{S}\sum_{s=0}^{S-1} Q_{s+1} \\ &\leq \frac{1-\vartheta}{\vartheta S}(Q_0 - Q_S) + \frac{L\vartheta}{2m(1-\vartheta)S}(P_0 - P_S) \\ &\overset{(\star)}{\leq} \frac{1-\vartheta}{\vartheta S}\mathbb{E}\big[f(x_r) - f(x^\star)\big] + \frac{L\vartheta}{2m(1-\vartheta)S}\mathbb{E}\big[\,\|x_r - x^\star\|^2\,\big],\end{aligned}$$

where $(\star)$ follows from $\tilde{x}_0 = z_0^0 = x_r$.

Using μ-strong convexity at $(x_r, x^\star)$ (or quadratic growth), $f(x_r) - f(x^\star) \geq \frac{\mu}{2}\|x_r - x^\star\|^2$, we arrive at

$$\mathbb{E}\big[f(x_{r+1}) - f(x^\star)\big] \leq \left(\frac{1-\vartheta}{\vartheta S} + \frac{\kappa\vartheta}{m(1-\vartheta)S}\right)\mathbb{E}\big[f(x_r) - f(x^\star)\big].$$

Choosing $S = \left\lceil \omega \cdot \left(\frac{1-\vartheta}{\vartheta} + \frac{\kappa\vartheta}{m(1-\vartheta)}\right)\right\rceil$, we have $\mathbb{E}\big[f(x_{r+1}) - f(x^\star)\big] \leq \frac{1}{\omega}\cdot\mathbb{E}\big[f(x_r) - f(x^\star)\big]$. Thus, for any accuracy $\epsilon > 0$ and any constant $\omega > 1$, to guarantee that the output satisfies $\mathbb{E}\big[f(x_R)\big] - f(x^\star) \leq \epsilon$, we need to perform totally $R = O\left(\log\frac{f(x_0)-f(x^\star)}{\epsilon}\right)$ restarts. Note that the total oracle complexity of Algorithm 8 is #grad $= R \cdot S \cdot (n + 2m)$. Setting $m = \Theta(n)$ and minimizing S with respect to ϑ, we obtain the optimal choice $\vartheta = \frac{\sqrt{m}}{\sqrt{\kappa}+\sqrt{m}}$. In this case, we have $S = \lceil 2\omega\sqrt{\frac{\kappa}{m}}\rceil = O\left(\max\left\{1, \sqrt{\frac{\kappa}{n}}\right\}\right)$. Finally, the total oracle complexity is

$$\#\text{grad} = O\left(\max\left\{n, \sqrt{\kappa n}\right\}\log\frac{f(x_0) - f(x^\star)}{\epsilon}\right).$$

C.3 Proof of Theorem 12

We use the following lemma from [83] to bound the term in asynchrony. Although the gradient is evaluated at a different point, the same proof works in our case, and thus is omitted here.

Lemma 6 (Inequality (63) in [83])**.** *We have the following bound for the iterates z_k (defined at* (4.4)*) and $\hat{z}_k$ (defined at* (4.6)*) in one epoch of Algorithm 9:*

$$\mathbb{E}\left[\, \langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - z_k \rangle \,\right] \le \frac{\sqrt{\Delta}\eta}{2} \left(\sum_{j=(k-\tau)_+}^{k-1} \mathbb{E}\left[\, \left\| \mathcal{G}_{\hat{y}_j} \right\|^2 \right] + \tau \mathbb{E}\left[\, \| \mathcal{G}_{\hat{y}_k} \|^2 \,\right] \right).$$

We start with analyzing one epoch of virtual iterates defined at (4.4): $z_{k+1} = z_k - \eta \cdot \mathcal{G}_{\hat{y}_k}$. The expectation is first taken conditioned on the previous epochs.

$$\begin{aligned}
&\|z_{k+1} - x^\star\|^2 = \|z_k - x^\star - \eta \cdot \mathcal{G}_{\hat{y}_k}\|^2 \\
\Rightarrow\ & \langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - x^\star \rangle + \langle \mathcal{G}_{\hat{y}_k}, z_k - \hat{z}_k \rangle = \frac{\eta}{2} \|\mathcal{G}_{\hat{y}_k}\|^2 + \frac{1}{2\eta} \left(\|z_k - x^\star\|^2 - \|z_{k+1} - x^\star\|^2 \right) \\
\overset{(\star)}{\Rightarrow}\ & \mathbb{E}\left[\, \langle \nabla f(\hat{y}_k), \hat{z}_k - x^\star \rangle \,\right] = \frac{\eta}{2} \mathbb{E}\left[\, \|\mathcal{G}_{\hat{y}_k}\|^2 \,\right] + \mathbb{E}\left[\, \langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - z_k \rangle \,\right] \\
& \qquad + \frac{1}{2\eta} \left(\mathbb{E}\left[\, \|z_k - x^\star\|^2 \,\right] - \mathbb{E}\left[\, \|z_{k+1} - x^\star\|^2 \,\right] \right), \qquad \text{(C.3)}
\end{aligned}$$

where $(\star)$ follows from taking the expectation and the unbiasedness assumption $\mathbb{E}\big[\mathcal{G}_{\hat{y}_k} | \hat{z}_k\big] = \nabla f(\hat{y}_k)$.

Using the convexity at $x^\star$ and the inconsistent $\hat{y}_k$ (ordered at (4.5)), we have

$$\begin{aligned}
& f(\hat{y}_k) - f(x^\star) \\
\le\ & \langle \nabla f(\hat{y}_k), \hat{y}_k - x^\star \rangle \\
=\ & \langle \nabla f(\hat{y}_k), \hat{y}_k - \hat{z}_k \rangle + \langle \nabla f(\hat{y}_k), \hat{z}_k - x^\star \rangle \\
\overset{(\star)}{\le}\ & \frac{1-\vartheta}{\vartheta} \langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k \rangle - \frac{\varphi}{\vartheta} \langle \nabla f(\hat{y}_k), D \nabla f(\tilde{x}_s) \rangle + \langle \nabla f(\hat{y}_k), \hat{z}_k - x^\star \rangle .
\end{aligned}$$

where $(\star)$ follows from the construction $\hat{y}_k = \vartheta \hat{z}_k + (1-\vartheta)\, \tilde{x}_s - \varphi D \nabla f(\tilde{x}_s)$.

After taking the expectation and combining with (C.3), we obtain

$$\begin{aligned}
\mathbb{E}\big[f(\hat{y}_k)\big] - f(x^\star) \le\ & \frac{1-\vartheta}{\vartheta} \mathbb{E}\left[\, \langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k \rangle \,\right] - \frac{\varphi}{\vartheta} \mathbb{E}\left[\, \langle \nabla f(\hat{y}_k), D \nabla f(\tilde{x}_s) \rangle \,\right] \\
& + \frac{\eta}{2} \mathbb{E}\left[\, \|\mathcal{G}_{\hat{y}_k}\|^2 \,\right] + \mathbb{E}\left[\, \langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - z_k \rangle \,\right] \\
& + \frac{1}{2\eta} \left(\mathbb{E}\left[\, \|z_k - x^\star\|^2 \,\right] - \mathbb{E}\left[\, \|z_{k+1} - x^\star\|^2 \,\right] \right).
\end{aligned}$$

Summing this inequality from $k = 0$ to $m - 1$, we have

$$\begin{aligned}
&\sum_{k=0}^{m-1} \mathbb{E}\big[f(\hat{y}_k) - f(x^\star)\big] \\
\leq& \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\,\big] - \frac{\varphi}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\,\big] + \frac{\eta}{2}\mathbb{E}\big[\,\|\mathcal{G}_{\hat{y}_k}\|^2\,\big]\right) \\
&+ \sum_{k=0}^{m-1} \mathbb{E}\big[\,\langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - z_k\rangle\,\big] + \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\big[\,\|z_m - x^\star\|^2\,\big]\right).
\end{aligned}$$

Invoke Lemma 6 to bound the asynchronous perturbation.

$$\begin{aligned}
&\sum_{k=0}^{m-1} \mathbb{E}\big[f(\hat{y}_k) - f(x^\star)\big] \\
\leq& \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\,\big] - \frac{\varphi}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\,\big] + \frac{\eta}{2}\mathbb{E}\big[\,\|\mathcal{G}_{\hat{y}_k}\|^2\,\big]\right) \\
&+ \frac{\sqrt{\Delta}\eta}{2}\sum_{k=0}^{m-1}\left(\sum_{j=(k-\tau)_+}^{k-1} \mathbb{E}\big[\,\big\|\mathcal{G}_{\hat{y}_j}\big\|^2\,\big] + \tau\mathbb{E}\big[\,\|\mathcal{G}_{\hat{y}_k}\|^2\,\big]\right) + \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\big[\,\|z_m - x^\star\|^2\,\big]\right) \\
=& \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\,\big] - \frac{\varphi}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\,\big]\right. \\
&\left.+ \frac{\eta(1+\sqrt{\Delta}\tau)}{2}\mathbb{E}\big[\,\|\mathcal{G}_{\hat{y}_k}\|^2\,\big]\right) + \frac{\sqrt{\Delta}\eta}{2}\sum_{k=0}^{m-1}\sum_{j=(k-\tau)_+}^{k-1} \mathbb{E}\big[\,\big\|\mathcal{G}_{\hat{y}_j}\big\|^2\,\big] \\
&+ \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\big[\,\|z_m - x^\star\|^2\,\big]\right) \\
\leq& \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\,\big] - \frac{\varphi}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\,\big]\right. \\
&\left.+ \frac{\eta(1+\sqrt{\Delta}\tau)}{2}\mathbb{E}\big[\|\mathcal{G}_{\hat{y}_k}\|^2\big]\right) + \frac{\sqrt{\Delta}\eta\tau}{2}\sum_{k=0}^{m-1}\mathbb{E}\big[\|\mathcal{G}_{\hat{y}_k}\|^2\big] + \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\big[\|z_m - x^\star\|^2\big]\right) \\
=& \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\,\big] - \frac{\varphi}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\,\big]\right. \\
&\left.+ \frac{\eta(1+2\sqrt{\Delta}\tau)}{2}\mathbb{E}\big[\,\|\mathcal{G}_{\hat{y}_k}\|^2\,\big]\right) + \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\big[\,\|z_m - x^\star\|^2\,\big]\right).
\end{aligned}$$

For any $m \geq \widetilde{\tau} \geq \tau$, we choose $\eta = \frac{(1-\vartheta)}{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}$, and then

$$\begin{aligned}
&\sum_{k=0}^{m-1} \mathbb{E}\big[f(\hat{y}_k) - f(x^\star)\big] \\
&\leq \sum_{k=0}^{m-1}\left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\,\big] - \frac{\varphi}{\vartheta}\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\,\big] + \frac{1-\vartheta}{2L\vartheta}\mathbb{E}\big[\,\|\mathcal{G}_{\hat{y}_k}\|^2\,\big]\right) \\
&\quad + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2(1-\vartheta)}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\big[\,\|z_m - x^\star\|^2\,\big]\right).
\end{aligned}$$

Note that $\mathbb{E}\big[\,\|\mathcal{G}_{\hat{y}_k}\|^2\,\big] = \mathbb{E}\big[\,\|\mathcal{G}_{\hat{y}_k} - \nabla f(\hat{y}_k)\|^2\,\big] + \mathbb{E}\big[\,\|\nabla f(\hat{y}_k)\|^2\,\big]$ due to the unbiasedness assumption. Using Lemma 3, we can conclude that

$$\begin{aligned}
&\sum_{k=0}^{m-1} \mathbb{E}\big[f(\hat{y}_k) - f(x^\star)\big] \\
&\leq \sum_{k=0}^{m-1}\left(\left(\frac{1-\vartheta}{L\vartheta} - \frac{\varphi}{\vartheta}\right)\mathbb{E}\big[\,\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\,\big] + \frac{1-\vartheta}{\vartheta}\big(f(\tilde{x}_s) - \mathbb{E}\big[f(\hat{y}_k)\big]\big)\right) \\
&\quad + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2(1-\vartheta)}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\big[\,\|z_m - x^\star\|^2\,\big]\right).
\end{aligned}$$

Choosing $\varphi = \frac{1-\vartheta}{L}$ and dividing both sides by m, we can arrange this inequality as

$$\begin{aligned}
\frac{1}{m}\sum_{k=0}^{m-1} \mathbb{E}\big[f(\hat{y}_k) - f(x^\star)\big] &\leq (1-\vartheta)\big(f(\tilde{x}_s) - f(x^\star)\big) \\
&\quad + \frac{L\vartheta^2(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\big[\,\|z_m - x^\star\|^2\,\big]\right).
\end{aligned}$$

Since $\tilde{x}_{s+1}$ is chosen uniformly at random from $\{\hat{y}_0, \ldots, \hat{y}_{m-1}\}$ and that $z_0^{s+1} = z_m^s$ (the first and the last virtual iterates exist in the shared memory, and z is unchanged after each epoch in Algorithm 9), the following holds

$$\begin{aligned}
\mathbb{E}\big[f(\tilde{x}_{s+1}) - f(x^\star)\big] &\leq (1-\vartheta)\big(f(\tilde{x}_s) - f(x^\star)\big) \\
&\quad + \frac{L\vartheta^2(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)}\left(\|z_0^s - x^\star\|^2 - \mathbb{E}\big[\,\|z_0^{s+1} - x^\star\|^2\,\big]\right).
\end{aligned}$$

For the sake of clarity, we denote $Q_s \triangleq \mathbb{E}\big[f(\tilde{x}_s)-f(x^\star)\big]$ and $P_s \triangleq \mathbb{E}\big[\,\|z_0^s - x^\star\|^2\,\big]$. Then, it holds that

$$Q_{s+1} \le \frac{1-\vartheta}{\vartheta}(Q_s - Q_{s+1}) + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)}\left(P_s - P_{s+1}\right).$$

Summing this inequality from $s = 0$ to $S-1$ and using Jensen's inequality, we have

$$\begin{aligned}\mathbb{E}\big[f(x_{r+1})\big] - f(x^\star) &\le \frac{1}{S}\sum_{s=0}^{S-1} Q_{s+1}\\ &\le \frac{1-\vartheta}{\vartheta S}(Q_0 - Q_S) + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)S}\left(P_0 - P_S\right)\\ &\le \frac{1-\vartheta}{\vartheta S}Q_0 + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)S}P_0.\end{aligned}$$

Using μ-strong convexity at $(x_r, x^\star)$ (or quadratic growth), we have $\|x_r - x^\star\|^2 \le \frac{2}{\mu}\big(f(x_r) - f(x^\star)\big) \Leftrightarrow P_0 \le \frac{2}{\mu}Q_0$ and thus

$$\mathbb{E}\big[f(x_{r+1})\big] - f(x^\star) \le \left(\frac{1-\vartheta}{\vartheta S} + \frac{\kappa\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{m(1-\vartheta)S}\right)\mathbb{E}\big[f(x_r) - f(x^\star)\big],$$

Letting $S = \left\lceil \omega\cdot\left(\frac{1-\vartheta}{\vartheta} + \frac{\kappa\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{m(1-\vartheta)}\right)\right\rceil$, we have $\mathbb{E}\big[f(x_{r+1})\big]-f(x^\star) \le \frac{1}{\omega}\cdot\mathbb{E}\big[f(x_r)-f(x^\star)\big]$. Then, since $\omega > 1$ is a constant, to achieve an ϵ-additive error, we need to restart totally $R = O\left(\log\frac{f(x_0)-f(x^\star)}{\epsilon}\right)$ times. Note that the total oracle complexity of Algorithm 9 is $\#\text{grad} = R\cdot S\cdot(n+2m)$. Setting $m = \Theta(n)$ and choosing ϑ that minimizes S, we obtain the optimal choice $\vartheta = \frac{\sqrt{m}}{\sqrt{\kappa(1+2\sqrt{\Delta}\widetilde{\tau})}+\sqrt{m}}$, which leads to

$$S = \left\lceil 2\omega\sqrt{\frac{\kappa}{m}(1+2\sqrt{\Delta}\widetilde{\tau})}\right\rceil = O\left(\max\left\{1, \sqrt{\frac{\kappa}{n}(1+2\sqrt{\Delta}\widetilde{\tau})}\right\}\right).$$

Finally, the total oracle complexity is

$$\#\text{grad} = O\left(\max\left\{n, \sqrt{\kappa n(1+2\sqrt{\Delta}\widetilde{\tau})}\right\}\log\frac{f(x_0)-f(x^\star)}{\epsilon}\right).$$

C.4 The Effect of the Constant ω

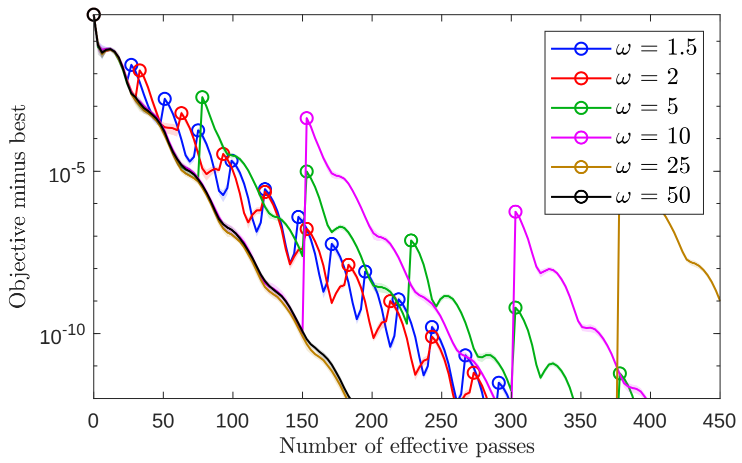


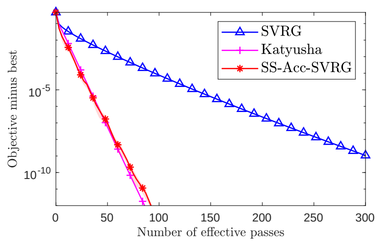
Figure C.1: The practical effect of ω. RCV1.train, run 10 seeds. The circle marks the restarting points, i.e., $\{x_r\}$. Shaded bands indicate ±1 standard deviation.

We numerically evaluate the effect of the constant ω in Figure C.1. We choose ω from 1.5 (frequent restart) to 50 (not restart in this task). The results on the News20 and KDD2010.S datasets are basically identical, and thus are omitted here. As we can see, unfortunately, the restarts only deteriorate the performance. An intuitive explanation is that the restart strategy is more conservative as it periodically retracts the point. In theory, the explicit dependence on ω (in the serial case) is

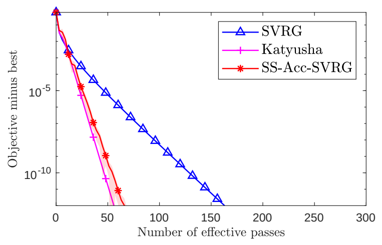
$$\frac{1}{\log \omega}\left\lceil 2\omega\sqrt{\frac{\kappa}{m}}\right\rceil,$$

which suggests that if ω is large, the convergence on the restarting points $\{x_r\}$ will be slower. This is observed in Figure C.1. However, what our current theory cannot explain is the superior performance of not performing restart, and we are not aware of a situation where the "aggressiveness" of no restart would hurt the convergence. Optimally tuning ω in the complexity will only lead to a small ω that is close to 1. Further investigation is needed for the convergence of Algorithms 8 and 9 without restart (we have strong numerical evidence in Appendix C.5 that without restart, they are still optimal).

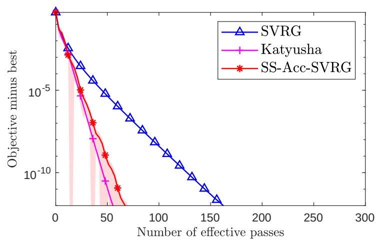
C.5 Justifying the $\sqrt{\kappa}$ Dependence



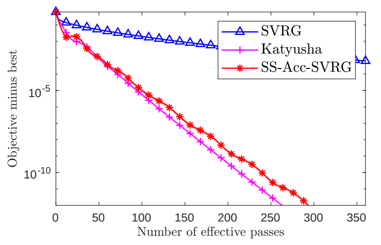
(a) KDD2010.S, $\mu = 10^{-6}$.



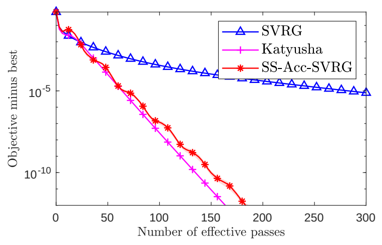
(b) RCV1.train, $\mu = 10^{-5}$.



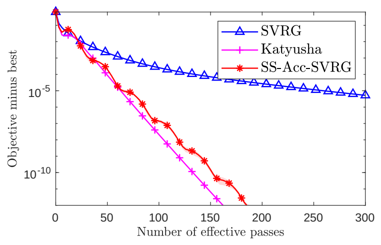
(c) News20, $\mu = 10^{-5}$.



(d) KDD2010.S, $\mu = 10^{-7}$.



(e) RCV1.train, $\mu = 10^{-6}$.



(f) News20, $\mu = 10^{-6}$.

Figure C.2: Justifying the $\sqrt{\kappa}$ dependence. Run 10 seeds. Shaded bands indicate ± 1 standard deviation.

We choose μ to be 10-times larger (Figures C.2a to C.2c) than the ones specified in Table 4.1 (Figures C.2d to C.2f) to verify the $\sqrt{\kappa}$ dependence. In this case, κ is 10-times smaller and we expect the accelerated methods to be around 3-times faster in terms of the number of data passes. This has been observed across all the datasets for Katyusha and SS-Acc-SVRG, which verifies the accelerated rate.

C.6 Sanity Check for Our Implementation

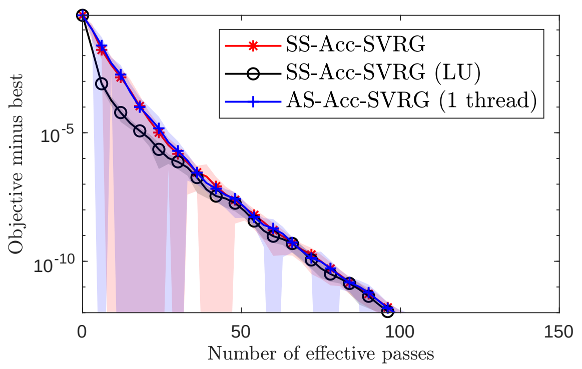


Figure C.3: Sanity check. a9a.dense, $\mu = 10^{-6}$. Run 20 seeds. Shaded bands indicate ± 1 standard deviation.

If the dataset is fully dense, then the following three schemes should be equivalent: SS-Acc-SVRG, SS-Acc-SVRG with lagged update implementation and AS-Acc-SVRG with a single thread. We construct a fully dense dataset by adding a small positive number to each elements in the a9a dataset [20] ($n = 32\,561, d = 123$), and we name this dataset as a9a.dense. The sanity check is then provided in Figure C.3. SS-Acc-SVRG (LU) shows slightly different convergence because we used an averaged snapshot instead of a random one in its implementation. Implementing lagged update technique with a random snapshot is quite tricky, since the last seen iteration might be in the previous epochs in this case. We tested SS-Acc-SVRG with an averaged snapshot and it shows almost identical convergence as SS-Acc-SVRG (LU).

C.7 Experimental Setup

All the methods are implemented in C++.

Serial setup We ran serial experiments on an HP Z440 machine with a single Intel Xeon E5-1630v4 with 3.70GHz cores, 16GB RAM, Ubuntu 18.04 LTS with GCC 9.4.0, MATLAB R2021a. We fixed the epoch length $m = 2n$ and chose the default parameters: SS-Acc-SVRG follows Theorem 8; Katyusha uses $\tau_2 = \frac{1}{2}, \tau_1 = \sqrt{\frac{m}{3\kappa}}, \alpha = \frac{1}{3\tau_1 L}$ [4]; SVRG uses $\eta = \frac{1}{4L}$.

Asynchronous setup We ran asynchronous experiments on a Dell PowerEdge R840 HPC with four Intel Xeon Platinum 8160 CPU @ 2.10GHz each with 24 cores, 768GB RAM, CentOS 7.9.2009 with GCC 9.3.1, MATLAB R2021a. We implemented the original version of KroMagnon in [98]. We fixed the epoch length $m = 2n$ for KroMagnon and AS-Acc-SVRG. AS-Acc-SVRG used the same parameters as in the serial case for all datasets. The step sizes of KroMagnon and ASAGA were chosen as follows: On RCV1.full, KroMagnon uses $\frac{1}{L}$, ASAGA uses $\frac{1}{1.3L}$; on Avazu-site, KroMagnon uses $\frac{1}{2L}$, ASAGA uses $\frac{1}{2L}$; on KDD2010, KroMagnon uses $\frac{1}{2L}$, ASAGA uses $\frac{1}{3L}$. Due to the scale of the tasks, we cannot do fine-grained grid search for the step sizes of KroMagnon and ASAGA. They were chosen to ensure a stable and consistent performance in the 20-thread experiments.

Appendix D

Appendix for Chapter 5

D.1 Extra Experimental Results

In this appendix, we provide more experimental results to further evaluate the Amortized Nesterov's Momentum.

- Table D.1 shows the detailed data of the parameter sweep experiments, where the convergence curves of these results are given in Appendix D.1.2.
- In Appendix D.1.1, we discuss our implementations of AM1-SGD and AM2-SGD .
- In Appendix D.1.3, we report the results of a full-batch loss experiment using ResNet18.
- In Appendix D.1.4, we compare the robustness of AM1-SGD and M-SGD on large momentum parameters.
- In Appendix D.1.5, we empirically compare the Amortized Nesterov's Momentum with classical momentum [123], aggregated momentum [95] and quasi-hyperbolic momentum [96].
- In Appendix D.1.6, we discuss the issues with learning rate schedulers.
- In Appendix D.1.7, we provide a CIFAR-100 experiment.
- In Appendix D.1.8, we add some convex experiments.
- In Appendix D.1.9, we provide a sanity check for our implementation.

Table D.1: Final test accuracy and average accuracy STD of training ResNet34 on CIFAR-10 over 5 runs (including the detailed data of the curves in Figure 5.2 and Figure 5.3a). For all the methods, $\eta_0 = 0.1, \beta = 0.9$. Multiple runs start with the same x_0.

METHOD	**DESCRIPTION**	**FINAL ACCURACY**	**Avg. STD**
SGD	Standard Pytorch	$93.41\% \pm 0.15\%$	0.99%
M-SGD	Standard Pytorch	$94.61\% \pm 0.15\%$	1.04%
AM1-SGD	Option I, $m = 1$, sanity check	$94.67\% \pm 0.14\%$	0.91%
AM1-SGD	Option I, $m = 3$	$94.63\% \pm 0.03\%$	0.64%
AM1-SGD	Option I, $m = 5$	$94.60\% \pm 0.10\%$	0.50%
AM1-SGD	Option I, $m = 7$	$94.64\% \pm 0.13\%$	0.44%
AM1-SGD	Option I, $m = 10$	$94.54\% \pm 0.13\%$	0.44%
AM1-SGD	Option I, $m = 20$	$94.38\% \pm 0.22\%$	0.40%
AM1-SGD	Option I, $m = 30$	$94.30\% \pm 0.15\%$	0.43%
OM-SGD	AM1-SGD (Opt. II, $m = 1$)	$94.73\% \pm 0.11\%$	0.63%
AM1-SGD	Option II, $m = 3$	$94.66\% \pm 0.14\%$	0.41%
AM1-SGD	Option II, $m = 5$	$94.60\% \pm 0.08\%$	0.27%
AM1-SGD	Option II, $m = 7$	$94.51\% \pm 0.10\%$	0.28%
AM1-SGD	Option II, $m = 10$	$94.42\% \pm 0.12\%$	0.29%
AM1-SGD	Option II, $m = 20$	$94.36\% \pm 0.18\%$	0.31%
AM1-SGD	Option II, $m = 30$	$94.27\% \pm 0.13\%$	0.34%
AM2-SGD	Option I, $m = 1$, sanity check	$94.68\% \pm 0.21\%$	0.82%
AM2-SGD	Option I, $m = 5$	$94.57\% \pm 0.19\%$	0.59%
AM2-SGD	Option I, $m = 10$	$94.44\% \pm 0.14\%$	0.74%
AM2-SGD	Option I, $m = 20$	$94.31\% \pm 0.15\%$	0.74%
AM2-SGD	Option II, $m = 5$	$94.66\% \pm 0.11\%$	0.26%
AM2-SGD	Option II, $m = 10$	$94.50\% \pm 0.21\%$	0.28%
AM2-SGD	Option II, $m = 20$	$94.41\% \pm 0.14\%$	0.25%

D.1.1 Implementing AM1-SGD and AM2-SGD

Similar to M-SGD, it is easier to implement Option I in existing deep learning frameworks. To implement Option II, we can either maintain another identical network for the shifted point ($\tilde{x}$ or $\bar{\phi}$) or temporarily change the network parameters in the evaluation phase. In our implementations of AM1-SGD and AM2-SGD, we simply adopt the latter solution.

Algorithm 17 AM1-SGD (Algorithm 11 with improved efficiency)

```
Input: Initial guess $x_0$, learning rate $\eta$, momentum $\beta$, amortization length $m$, iteration number $K$.
Initialize: $x \leftarrow x_0, \tilde{x} \leftarrow x_0, \tilde{v}^+ \leftarrow -m \cdot x_0$.
1: for $k = 0, \ldots, K-1$ do
2:     $x \leftarrow x - \eta \cdot \nabla f_{i_k}(x)$.
3:     $\tilde{v}^+ \leftarrow \tilde{v}^+ + x$.
4:     if $(k+1) \mod m = 0$ then
5:         $x \leftarrow x + (\beta/m) \cdot \tilde{v}^+$.
6:         $\tilde{x} \leftarrow \tilde{x} + (1/m) \cdot \tilde{v}^+, \tilde{v}^+ \leftarrow -m \cdot \tilde{x}$.
7:     end if
8: end for
Output: Option I: $x$, Option II: $\tilde{x}$.
```

For AM1-SGD, we can improve the efficiency of Algorithm 11 by maintaining a running scaled momentum $\tilde{v}^+ \triangleq m \cdot (\tilde{x}^+ - \tilde{x})$ instead of the running average $\tilde{x}^+$ as shown in Algorithm 17. Then, in one m-iterations loop, for each of the first $m-1$ iterations, AM1-SGD requires 1 vector addition and 1 scaled vector addition. At the m-th iteration, it requires 1 vector addition, 1 scalar-vector multiplication and 3 scaled vector additions. In comparison, M-SGD (standard PyTorch) requires 1 vector addition, 1 (in-place) scalar-vector multiplication and 2 scaled vector additions per iteration. Thus, as long as $m > 2$, AM1-SGD has lower amortized cost than M-SGD. In terms of memory complexity, AM1-SGD requires one more auxiliary buffer than M-SGD. For AM2-SGD, we implement Algorithm 13. Note that at each iteration, we sample an index in $[m]$ as j_{k+1} and obtain the stored index j_k.

D.1.2 The Effect of m on Convergence

We show in Figure D.1 how m affects the convergence of test accuracy. The results show that increasing m speeds up the convergence in the early stage. While for AM1-SGD the convergences of Option I and Option II are similar, AM2-SGD with Option II is consistently better than with Option I in this experiment. It seems that AM2-SGD with Option I does not benefit from increasing m and the algorithm is not robust. Thus, we do not recommend using Option I for AM2-SGD.

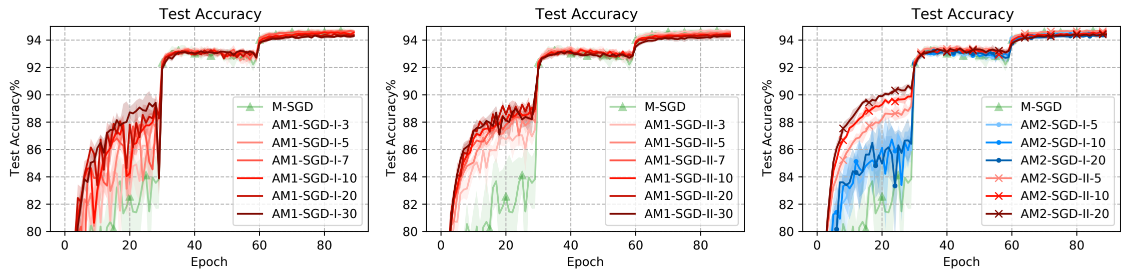
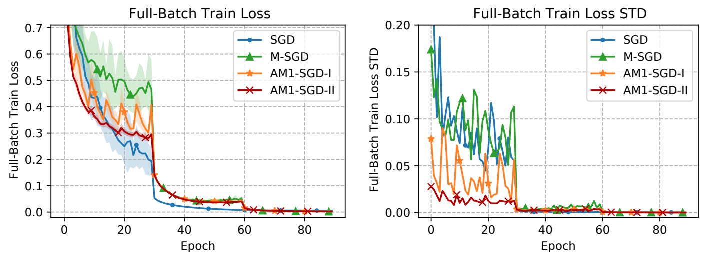


Figure D.1: Convergence of test accuracy from the parameter sweep experiments in Table D.1. Labels are formatted as 'AM1/2-SGD-{*Option*}-{m}'.

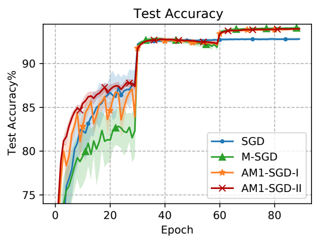
D.1.3 Full-batch Loss Experiment

We did a full-batch loss experiment of training ResNet18 with pre-activation [54] on CIFAR-10 in Figure D.2 & Table D.2. The accuracy results are given in Figure D.3 & Table D.3. These results are reminiscent of the ResNet34 experiments (Figure 5.3b and Table 5.1).



METHOD	Avg. STD
SGD	0.033
M-SGD	0.034
AM1-SGD-I	**0.014**
AM1-SGD-II	**0.006**

Figure D.2 & Table D.2: ResNet18 with pre-activation on CIFAR-10. For all methods, $\eta_0 = 0.1, \beta = 0.9$, run 20 seeds. For AM1-SGD, $m = 5$ and its labels are formatted as 'AM1-SGD-{*Option*}'. Shaded bands indicate ± 1 standard deviation.

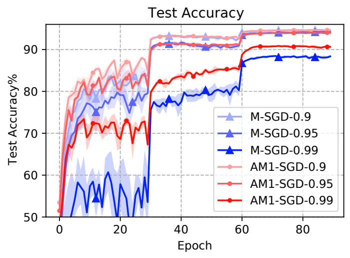


METHOD	FINAL ACCURACY	Avg. STD
SGD	$92.81\% \pm 0.15\%$	1.01%
M-SGD	$94.06\% \pm 0.17\%$	1.10%
AM1-SGD-I	$93.97\% \pm 0.15\%$	**0.54%**
AM1-SGD-II	$93.95\% \pm 0.19\%$	**0.32%**

Figure D.3 & Table D.3: ResNet18 with pre-activation on CIFAR-10. For all methods, $\eta_0 = 0.1, \beta = 0.9$, run 20 seeds. For AM1-SGD, $m = 5$. Shaded bands indicate ± 1 standard deviation.

D.1.4 Robustness on Large Momentum Parameters

We compare the robustness of M-SGD and AM1-SGD when β is large in Figure D.4 & Table D.4. AM1-SGD uses Option I, which omits the tail averaging effect at the output point. As we can see, the STD error of M-SGD scales up significantly when β is larger and the performance is more affected by a large β compared with AM1-SGD.

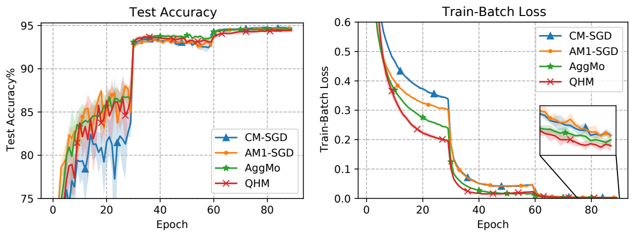


METHOD	FINAL ACCURACY	Avg. STD
M-SGD-0.9	$94.61\% \pm 0.15\%$	1.04%
M-SGD-0.95	$94.20\% \pm 0.12\%$	1.20%
M-SGD-0.99	$88.37\% \pm 0.36\%$	2.56%
AM1-SGD-0.9	$94.60\% \pm 0.10\%$	0.50%
AM1-SGD-0.95	$93.94\% \pm 0.07\%$	0.58%
AM1-SGD-0.99	$90.64\% \pm 0.38\%$	0.90%

Figure D.4 & Table D.4: ResNet34 on CIFAR-10. $\eta_0 = 0.1, \beta \in \{0.9, 0.95, 0.99\}$, run 5 seeds (the $\beta = 0.9$ results are copied from Table D.1). Labels are formatted as "$\{Algorithm\}$-$\{\beta\}$".

D.1.5 Comparison with Other Momentum

In this section, we compare AM1-SGD (Option I) with classical momentum [123], AggMo [95] and QHM [96] in our basic case study (training ResNet34 on CIFAR-10). Since we are not aware of what makes a fair comparison with these methods (e.g., it is not clear what is the effective learning rate for AM1-SGD), we compare them based on the default hyper-parameter settings suggested by their papers.



METHOD	FINAL ACCURACY	Avg. STD
CM-SGD	$94.69\% \pm 0.21\%$	1.11%
AM1-SGD	$94.60\% \pm 0.10\%$	0.50%
AggMo	$94.56\% \pm 0.14\%$	0.78%
QHM	$94.43\% \pm 0.23\%$	1.07%
M-SGD	$94.61\% \pm 0.15\%$	1.04%

Figure D.5 & Table D.5: ResNet34 on CIFAR-10. Run 5 seeds. The results of AM1-SGD and M-SGD are copied from Table D.1.

Classical Momentum The SGD with classical momentum (CM-SGD) that is widely used in deep learning has the following scheme (standard PyTorch) ($v^{cm} \in \mathbb{R}^d, v_0^{cm} = \vec{0}$):

$$\begin{aligned} v_{k+1}^{cm} &= \beta \cdot v_k^{cm} + \nabla f_{i_k}(x_k), \\ x_{k+1} &= x_k - \eta \cdot v_{k+1}^{cm}, \text{ for } k \geq 0. \end{aligned}$$

CM-SGD with its typical hyper-parameter settings ($\eta_0 = 0.1, \beta = 0.9$) is observed to achieve similar generalization performance as M-SGD. However, CM-SGD is more unstable and prone to oscillations [95], which makes it less robust than M-SGD as shown in Table D.5.

Aggregated Momentum (AggMo) AggMo uses multiple momentum buffers, which is inspired by the passive damping from physics literature [95]. AggMo uses the following update rules (for $t = 1, \ldots, T$, $v^{(t)} \in \mathbb{R}^d, v_0^{(t)} = \vec{0}$):

$$v_{k+1}^{(t)} = \beta^{(t)} \cdot v_k^{(t)} - \nabla f_{i_k}(x_k), \text{ for } t = 1, \ldots, T,$$
$$x_{k+1} = x_k + \frac{\eta}{T} \cdot \sum_{t=1}^{T} v_{k+1}^{(t)}, \text{ for } k \geq 0.$$

We used the exponential hyper-parameter setting recommended in the original work with the scale-factor $a = 0.1$ fixed, $\beta^{(t)} = 1 - a^{t-1}$, for $t = 1, \ldots, T$ and choosing T in $\{2, 3, 4\}$. We found that $T = 2$ gave the best performance in this experiment. As shown in Figure D.5 & Table D.5, with the help of passive damping, AggMo is more stable and robust compared with CM-SGD.

Quasi-hyperbolic Momentum (QHM) Ma and Yarats [96] introduce the immediate discount factor $\nu \in \mathbb{R}$ for the momentum scheme, which results in the QHM update rules ($\alpha \in \mathbb{R}, v^{qh} \in \mathbb{R}^d, v_0^{qh} = \vec{0}$):

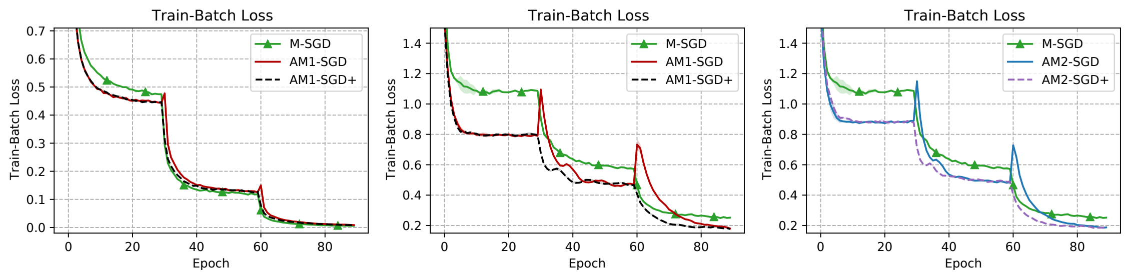
$$v_{k+1}^{qh} = \beta \cdot v_k^{qh} + (1 - \beta) \cdot \nabla f_{i_k}(x_k),$$
$$x_{k+1} = x_k - \alpha \cdot (\nu \cdot v_{k+1}^{qh} + (1 - \nu) \cdot \nabla f_{i_k}(x_k)), \text{ for } k \geq 0.$$

Here we used the recommended hyper-parameter setting for QHM ($\alpha_0 = 1.0, \beta = 0.999, \nu = 0.7$).

Figure D.5 shows that AM1-SGD, AggMo and QHM achieve faster convergence in the early stage while CM-SGD has the highest final accuracy. In terms of robustness, huge gaps are observed when comparing AM1-SGD with the remaining methods in Table D.5. Note that AM1-SGD is more efficient than both QHM and AggMo, and is as efficient as CM-SGD.

We also plot the convergence of train-batch loss for all the methods in Figure D.5. Despite of showing worse generalization performance, both QHM and AggMo perform better on reducing the train-batch loss in this experiment, which is consistent with the results reported in Ma and Yarats [96], Lucas et al. [95].

D.1.6 Issues with Learning Rate Schedulers



(a) $\beta = 0.95$ (b) $\beta = 0.995$ (c) $\beta = 0.995$

Figure D.6: ResNet18 on CIFAR-10. $\eta_0 = 0.1, \beta \in \{0.95, 0.995\}$. '+' represents performing a restart after each learning rate reduction.

We show in Figure D.6 that when β is large for the task, using step learning rate scheduler with decay factor 10, a performance drop is observed after each reduction. We fix this issue by performing a restart after each learning rate reduction (labeled with '+'). We plot the train-batch loss here because we find that the phenomenon is clearer in this way. Note that output options do not affect the convergence of train-batch loss, and thus this phenomenon exists for both options. If $\beta = 0.9$, there is no observable performance drop in this experiment.

For smooth-changing schedulers such as the cosine annealing scheduler [94], the amortized momentum works well as shown in Figure D.7.

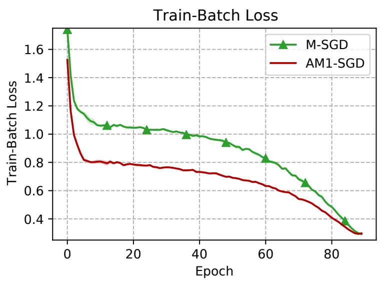


Figure D.7: ResNet18 on CIFAR-10. Using cosine annealing scheduler (without restarts), $\eta_0 = 0.1, \beta = 0.995$.

D.1.7 CIFAR-100 Experiment

We report the results of training DenseNet121 [60] on CIFAR-100 in Figure D.8, which shows that both AM1-SGD and AM2-SGD perform well before the final learning rate reduction. However, the final accuracies are lowered around 0.6% compared with M-SGD. We also notice that SGD reduces the train-batch loss at an incredibly fast rate and the losses it reaches are consistently lower than other methods in the entire 300 epochs. However, this performance is not reflected in the convergence of test accuracy. We believe that this phenomenon suggests that the DenseNet model is actually "overfitting" M-SGD (since in the ResNet experiments, M-SGD always achieves a lower train loss than SGD after the final learning rate reduction).

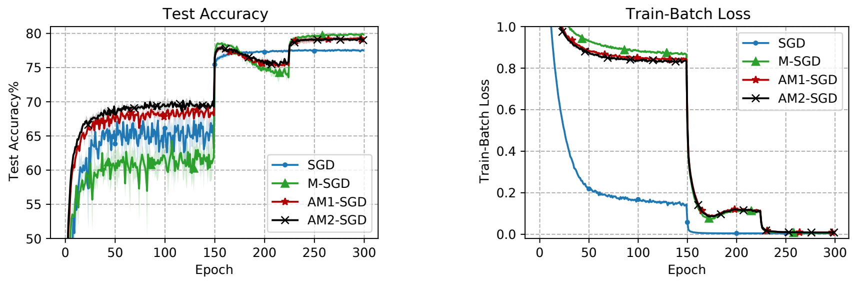


Figure D.8: DenseNet121 on CIFAR-100. For all methods, $\eta_0 = 0.1, \beta = 0.9$, run 3 seeds. AM1-SGD and AM2-SGD use Option II and $m = 5$. Shaded bands indicate ± 1 standard deviation.

D.1.8 Convex Experiments

We provide empirical results for AM1-SGD (Algorithm 10) to justify its convergence guarantee (Theorem 13). We consider the following simple convex empirical risk minimization task ($x \in \mathbb{R}^n, a_i \in \mathbb{R}^n, b_i \in \{-1, +1\}, \forall i \in [|D|]$):

$$\text{Logistic Regression: } f(x) = \frac{1}{|D|} \sum_{i=1}^{|D|} \log\left(1 + \exp\left(-b_i \langle a_i, x \rangle\right)\right).$$

We used the `a5a` dataset ($|D| = 6414$, $n = 123$) from LIBSVM [20]. By normalizing the dataset, we have $L = 0.25, M = 0$ in Assumption (a). The stochastic gradient oracle is defined as $\nabla f_i(x) = \nabla f(x) + \delta$, where δ is sampled uniformly from the sphere $\mathbb{B}(\mathbf{0}, \sigma)$ (which is centered at $\mathbf{0}$ and its radius is σ). This oracle satisfies Assumptions (b) and (c) (and also the "light tail" assumption). This type of noise is frequently used to escape saddle points in non-convex optimization or to ensure differential privacy. We fixed $x_0 = \mathbf{0}$ and estimated $V_d(x^\star, x_0) = \frac{1}{2}\|x_0 - x^\star\|_2^2$ as 350 by running a small amount of iterates. Then we can run AM1-SGD with the parameter choices specified in Theorem 13.

We compared AM1-SGD with M-SGD (or AC-SA, which corresponds to choosing $m = 1$). In Theorem 13, when σ is large, AM1-SGD is expected to converge faster than M-SGD and it also has a smaller deviation predicted by Theorem 13b. We justify these predictions in Figure D.9.

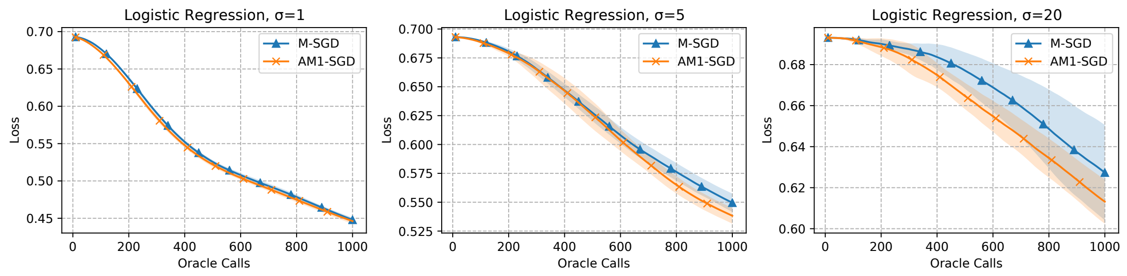


Figure D.9: Comparisons between AM1-SGD and M-SGD (AC-SA) in different noise levels. $m = 10$, run 10 seeds. Shaded bands indicate ± 1 standard deviation.

We also studied the effect of choosing different m in Figure D.10. The results are similar to the deep learning experiment in Figure 5.3a (right): m needs to be sufficiently large to show the effect, and after some point, not much benefit can be obtained by further increasing m.

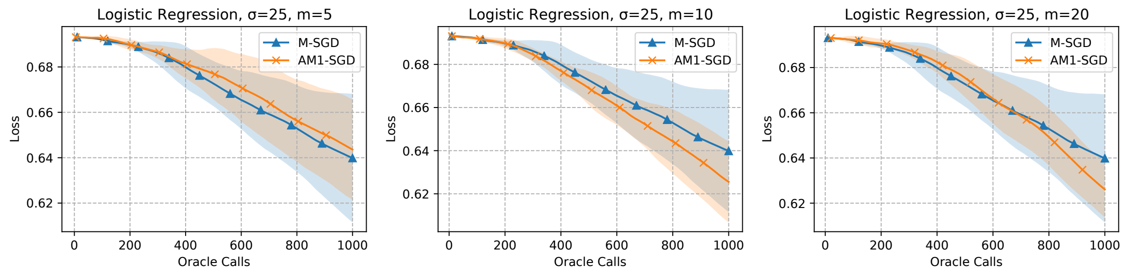


Figure D.10: Effect of choosing different m. Run 10 seeds. Shaded bands indicate ± 1 standard deviation.

D.1.9 Sanity Check

When $m = 1$, both AM1-SGD and AM2-SGD (Option I) are equivalent to M-SGD, we plot their convergence in Figure D.11 as a sanity check (the detailed data is given in Table D.1).

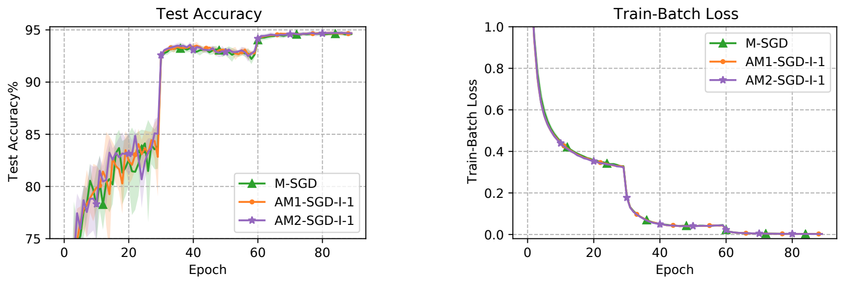


Figure D.11: Sanity check. Labels are formatted as 'AM{1/2}-SGD-{*Option*}-{m}'.

We observed that when $m = 1$, both AM1-SGD and AM2-SGD have a lower STD error than M-SGD. We believe that it is because they both maintain the iterates without scaling, which is numerically more stable than M-SGD (M-SGD in standard PyTorch maintains a scaled buffer, i.e., $v_k^{pt} = \eta^{-1}\beta^{-1} \cdot (y_k - x_k)$).

D.2 Technical Lemma

The following lemma is a known result for the martingale difference (cf. Lemma 2 in Lan et al. [80]):

Lemma 7. *With $N > 0$, let $\xi_0, \xi_1, \ldots, \xi_{N-1}$ be a sequence of i.i.d. random variables, for $t = 0, \ldots, N-1$, $\sigma_t > 0$ be a deterministic number and $\psi_t = \psi_t(\xi_0, \ldots, \xi_t)$ be a deterministic measurable function such that $\mathbb{E}_{\xi_t}\big[\psi_t\big] = 0$ a.s. and $\mathbb{E}_{\xi_t}\big[\exp\{\psi_t^2/\sigma_t^2\}\big] \leq \exp\{1\}$ a.s.. Then for any $\Lambda \geq 0$,*

$$\operatorname{Prob}\left\{\left\{\sum_{t=0}^{N-1} \psi_t \geq \Lambda \sqrt{\sum_{t=0}^{N-1} \sigma_t^2}\right\}\right\} \leq \exp\{-\Lambda^2/3\}.$$

D.3 Proof of Lemma 4

This Lemma is provided in a similar way as in Lan [78], Ghadimi and Lan [46], we give a proof here for completeness.

Based on the convexity (Assumption (a)), we have

$$\begin{aligned} f(x) - f(x^\star) \leq\ & \underbrace{\langle \nabla f(x), x - z\rangle}_{R_0} + \underbrace{\langle \nabla f(x) - \nabla f_i(x), z - x^\star\rangle}_{R_1} + \underbrace{\langle \nabla f_i(x), z - z^+\rangle}_{R_2} \\ & + \underbrace{\langle \nabla f_i(x), z^+ - x^\star\rangle}_{R_3}. \end{aligned} \tag{D.1}$$

We upper bound the terms on the right side one-by-one.

For R_0,

$$R_0 \overset{(\star)}{=} \frac{\beta}{1-\beta} \langle \nabla f(x), y - x\rangle \leq \frac{\beta}{1-\beta}\big(f(y) - f(x)\big), \tag{D.2}$$

where $(\star)$ uses the relation between x and z, i.e., $(1-\beta)\cdot(x-z) = \beta\cdot(y-x)$.

For R_2, based on Assumption (a), we have

$$f(y^+) - f(x) + \langle \nabla f(x), x - y^+\rangle \leq \frac{L}{2}\left\|x - y^+\right\|^2 + M\left\|x - y^+\right\|.$$

Then, noting that $x - y^+ = (1-\beta)\cdot(z - z^+)$, we can arrange the above inequality as

$$\begin{aligned}R_2 &\le \frac{L(1-\beta)}{2}\left\|z-z^+\right\|^2 + \frac{1}{1-\beta}\big(f(x)-f(y^+)\big) + \left\langle \nabla f(x) - \nabla f_i(x), z^+ - z\right\rangle \\ &\quad + M\left\|z-z^+\right\| \\ &\le \frac{L(1-\beta)}{2}\left\|z-z^+\right\|^2 + \frac{1}{1-\beta}\big(f(x)-f(y^+)\big) \\ &\quad + \big(\left\|\nabla f(x)-\nabla f_i(x)\right\|_* + M\big)\left\|z-z^+\right\|.\end{aligned}$$

Using Young's inequality with $\zeta > 0$, we obtain

$$\begin{aligned}R_2 \le \frac{L(1-\beta)+\zeta}{2}\left\|z-z^+\right\|^2 + \frac{1}{1-\beta}\big(f(x)-f(y^+)\big) \\ + \frac{(\left\|\nabla f(x)-\nabla f_i(x)\right\|_* + M)^2}{2\zeta}.\end{aligned} \tag{D.3}$$

For R_3, based on the optimality condition of

$$\operatorname{prox}_{\alpha h}\big(z, \alpha\cdot\nabla f_i(x)\big) \triangleq \arg\min_{u\in X}\big\{V_d(u,z) + \alpha\left\langle \nabla f_i(x), u\right\rangle + \alpha h(u)\big\},$$

and denoting $\partial h(z^+) \in E^*$ as a subgradient of h at z^+, we have for any $w \in X$,

$$\left\langle \nabla d(z^+) - \nabla d(z) + \alpha\cdot\nabla f_i(x) + \alpha\cdot\partial h(z^+), w - z^+\right\rangle \ge 0,$$

$$\begin{aligned}\left\langle \nabla f_i(x), z^+ - w\right\rangle &\le \left\langle \partial h(z^+), w - z^+\right\rangle + \frac{1}{\alpha}\left\langle \nabla d(z^+) - \nabla d(z), w - z^+\right\rangle \\ &\le h(w) - h(z^+) + \frac{1}{\alpha}\left\langle \nabla d(z^+) - \nabla d(z), w - z^+\right\rangle.\end{aligned}$$

Choosing $w = x^\star$ and applying the triangle equality of Bregman divergence, we obtain

$$\begin{aligned}R_3 &\le h(x^\star) - h(z^+) + \frac{1}{\alpha}\left\langle \nabla d(z^+) - \nabla d(z), x^\star - z^+\right\rangle \\ &= h(x^\star) - h(z^+) + \frac{1}{\alpha}\big(V_d(x^\star, z) - V_d(x^\star, z^+) - V_d(z^+, z)\big) \\ &\overset{(\star)}{\le} h(x^\star) - h(z^+) + \frac{1}{\alpha}\big(V_d(x^\star, z) - V_d(x^\star, z^+)\big) - \frac{1}{2\alpha}\left\|z^+ - z\right\|^2,\end{aligned} \tag{D.4}$$

where $(\star)$ follows from the 1-strong convexity of d, which implies that $V_d(x,y) \ge \frac{1}{2}\left\|x-y\right\|^2, \forall x, y \in X$.

Finally, by upper bounding (D.1) using (D.2), (D.3), (D.4), we conclude that

$$\begin{aligned} f(x) - f(x^\star) \le R_1 &+ \frac{\beta}{1-\beta}\big(f(y) - f(x)\big) + \frac{L(1-\beta) + \zeta - \alpha^{-1}}{2}\left\|z - z^+\right\|^2 \\ &+ \frac{1}{1-\beta}\big(f(x) - f(y^+)\big) + h(x^\star) - h(z^+) \\ &+ \frac{(\|\nabla f(x) - \nabla f_i(x)\|_* + M)^2}{2\zeta} + \frac{1}{\alpha}\left(V_d(x^\star, z) - V_d(x^\star, z^+)\right), \end{aligned}$$

After simplification,

$$\begin{aligned} &\frac{1}{1-\beta}\big(f(y^+) - f(x^\star)\big) \\ &\le \frac{\beta}{1-\beta}\big(f(y) - f(x^\star)\big) + \frac{L(1-\beta) + \zeta - \alpha^{-1}}{2}\left\|z - z^+\right\|^2 \\ &\quad + h(x^\star) - h(z^+) + \frac{(\|\nabla f(x) - \nabla f_i(x)\|_* + M)^2}{2\zeta} + R_1 \\ &\quad + \frac{1}{\alpha}\left(V_d(x^\star, z) - V_d(x^\star, z^+)\right). \end{aligned} \tag{D.5}$$

Note that with the convexity of h and $y^+ = (1-\beta)\cdot z^+ + \beta \cdot y$, we have

$$\begin{aligned} h(y^+) &\le (1-\beta)h(z^+) + \beta h(y), \\ h(z^+) &\ge \frac{1}{1-\beta}h(y^+) - \frac{\beta}{1-\beta}h(y). \end{aligned}$$

Using the above inequality and choosing $\zeta = \alpha^{-1} - L(1-\beta) > 0 \Rightarrow \alpha(1-\beta) < 1/L$, we can arrange (D.5) as

$$\begin{aligned} \frac{1}{1-\beta}\big(F(y^+) - F(x^\star)\big) \le \frac{\beta}{1-\beta}\big(F(y) - F(x^\star)\big) &+ \frac{1}{\alpha}\left(V_d(x^\star, z) - V_d(x^\star, z^+)\right) \\ &+ \frac{(\|\nabla f(x) - \nabla f_i(x)\|_* + M)^2}{2(\alpha^{-1} - L(1-\beta))} + R_1. \end{aligned}$$

D.4 Proof of Theorem 13a

Using Assumption (c), Lemma 4 with

$$\begin{cases} x = x_k \\ z = z_k \\ z^+ = z_{k+1} \\ y = \tilde{x}_s \\ y^+ = x_{k+1} \\ \alpha = \alpha_s \\ \beta = \beta_s \end{cases}, \tag{D.6}$$

and taking expectation, if $\alpha_s(1-\beta_s) < 1/L$, we have

$$\frac{1}{1-\beta_s}\big(\mathbb{E}_{i_k}\left[F(x_{k+1})\right] - F(x^\star)\big) + \frac{1}{\alpha_s}\mathbb{E}_{i_k}\left[V_d(x^\star, z_{k+1})\right]$$
$$\leq \frac{\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{\alpha_s}V_d(x^\star, z_k) + \frac{(\sigma+M)^2}{2(\alpha_s^{-1} - L(1-\beta_s))}.$$

Summing the above inequality from $k = sm, \dots, sm+m-1$, we obtain

$$\frac{1}{(1-\beta_s)m}\sum_{j=1}^{m}\big(\mathbb{E}\left[F(x_{sm+j})\right] - F(x^\star)\big) + \frac{1}{\alpha_s m}\mathbb{E}\left[V_d(x^\star, z_{(s+1)m})\right]$$
$$\leq \frac{\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{\alpha_s m}V_d(x^\star, z_{sm}) + \frac{(\sigma+M)^2}{2(\alpha_s^{-1} - L(1-\beta_s))},$$

Using the definition of $\tilde{x}_{s+1}$ and convexity,

$$\frac{\alpha_s}{1-\beta_s}\big(\mathbb{E}\left[F(\tilde{x}_{s+1})\right] - F(x^\star)\big) + \frac{1}{m}\mathbb{E}\left[V_d(x^\star, z_{(s+1)m})\right]$$
$$\leq \frac{\alpha_s\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{m}V_d(x^\star, z_{sm}) + \frac{\alpha_s(\sigma^2+M^2)}{\alpha_s^{-1} - L(1-\beta_s)}. \tag{D.7}$$

It can be verified that with the choices $\beta_s = \frac{s}{s+2}$ and $\alpha_s = \frac{\lambda_1}{L(1-\beta_s)}$, the following holds for $s \geq 0$,

$$\frac{\alpha_{s+1}\beta_{s+1}}{1-\beta_{s+1}} \leq \frac{\alpha_s}{1-\beta_s} \text{ and } \beta_0 = 0. \tag{D.8}$$

Thus, by telescoping (D.7) from $s = S-1, \dots, 0$, we obtain

$$\frac{\alpha_{S-1}}{1-\beta_{S-1}}\big(\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star)\big) + \frac{1}{m}\mathbb{E}\left[V_d(x^\star, z_{Sm})\right] \leq \frac{1}{m}V_d(x^\star, x_0) + \sum_{s=0}^{S-1}\frac{\alpha_s(\sigma^2+M^2)}{\alpha_s^{-1} - L(1-\beta_s)},$$

and thus,

$$\begin{aligned}\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) &\leq \frac{4L}{\lambda_1 m(S+1)^2}V_d(x^\star, x_0) + \frac{4L(\sigma^2+M^2)}{\lambda_1(S+1)^2}\sum_{s=0}^{S-1}\frac{\alpha_s^2}{1-\alpha_s(1-\beta_s)L} \\ &\overset{(a)}{\leq} \frac{4L}{\lambda_1 m(S+1)^2}V_d(x^\star, x_0) + \frac{3\lambda_1(\sigma^2+M^2)}{L(S+1)^2}\sum_{s=0}^{S-1}(s+2)^2 \\ &\overset{(b)}{\leq} \frac{4L}{\lambda_1 m(S+1)^2}V_d(x^\star, x_0) + \frac{8\lambda_1(\sigma^2+M^2)(S+1)}{L},\end{aligned}$$

where (a) follows from $\lambda_1 \leq \frac{2}{3}$ and (b) holds because $0 \leq x \mapsto (x+2)^2$ is non-decreasing and thus

$$\sum_{s=0}^{S-1}(s+2)^2 \leq \int_0^S (x+2)^2 dx \leq \frac{(S+2)^3}{3} \leq \frac{8(S+1)^3}{3}.$$

Denoting

$$\lambda_1^\star \triangleq \frac{L\sqrt{V_d(x^\star, x_0)}}{\sqrt{2m}\sqrt{\sigma^2+M^2}(S+1)^{\frac{3}{2}}},$$

and based on the choice of $\lambda_1 = \min\left\{\frac{2}{3}, \lambda_1^*\right\}$, if $\lambda_1^* \leq \frac{2}{3}$, we have

$$\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) \leq \frac{8\sqrt{2V_d(x^\star, x_0)}\sqrt{\sigma^2+M^2}}{m^{\frac{1}{2}}(S+1)^{\frac{1}{2}}}.$$

If $\lambda_1^* > \frac{2}{3}$,

$$\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) \leq \frac{6LV_d(x^\star, x_0)}{m(S+1)^2} + \frac{4\sqrt{2V_d(x^\star, x_0)}\sqrt{\sigma^2+M^2}}{m^{\frac{1}{2}}(S+1)^{\frac{1}{2}}}.$$

Thus, we conclude that

$$\mathbb{E}\left[F(\tilde{x}_S)\right] - F(x^\star) \leq \frac{6LV_d(x^\star, x_0)}{m(S+1)^2} + \frac{8\sqrt{2V_d(x^\star, x_0)}\sqrt{\sigma^2+M^2}}{m^{\frac{1}{2}}(S+1)^{\frac{1}{2}}}.$$

Substituting $S = K/m$ completes the proof.

D.5 Proof of Theorem 13b

To start with, using Lemma 4 with the parameter mapping (D.6), we have

$$
\begin{aligned}
&\frac{1}{1-\beta_s}\big(F(x_{k+1}) - F(x^\star)\big) + \frac{1}{\alpha_s} V_d(x^\star, z_{k+1}) \\
\le\ & \frac{\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{\alpha_s} V_d(x^\star, z_k) \\
&+ \frac{(\|\nabla f(x_k) - \nabla f_{i_k}(x_k)\|_* + M)^2}{2(\alpha_s^{-1} - L(1-\beta_s))} + \langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star \rangle \\
\le\ & \frac{\beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{\alpha_s} V_d(x^\star, z_k) + \frac{M^2}{\alpha_s^{-1} - L(1-\beta_s)} \\
&+ \frac{\|\nabla f(x_k) - \nabla f_{i_k}(x_k)\|_*^2}{\alpha_s^{-1} - L(1-\beta_s)} + \langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star \rangle .
\end{aligned}
$$

Summing the above inequality from $k = sm, \dots, sm+m-1$ and using the choice $\alpha_s = \frac{\lambda_1}{L(1-\beta_s)}$ with $\lambda_1 \le \frac{2}{3}$, we obtain

$$
\begin{aligned}
&\frac{\alpha_s}{1-\beta_s}\big(F(\tilde{x}_{s+1}) - F(x^\star)\big) + \frac{1}{m} V_d(x^\star, z_{(s+1)m}) \\
\le\ & \frac{\alpha_s \beta_s}{1-\beta_s}\big(F(\tilde{x}_s) - F(x^\star)\big) + \frac{1}{m} V_d(x^\star, z_{sm}) + 3\alpha_s^2 M^2 \\
&+ \frac{3\alpha_s^2}{m} \sum_{k=sm}^{sm+m-1} \|\nabla f(x_k) - \nabla f_{i_k}(x_k)\|_*^2 + \frac{\alpha_s}{m} \sum_{k=sm}^{sm+m-1} \langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star \rangle.
\end{aligned}
$$

With our parameter choices, the relations in (D.8) hold and thus we can telescope the above inequality from $s = S-1, \dots, 0$,

$$
\begin{aligned}
\frac{\alpha_{S-1}}{1-\beta_{S-1}}\big(F(\tilde{x}_S) - F(x^\star)\big) \le\ & \frac{1}{m} V_d(x^\star, x_0) + 3M^2 \sum_{s=0}^{S-1} \alpha_s^2 \\
&+ \frac{3}{m} \underbrace{\sum_{k=0}^{K-1} \alpha_{\lfloor k/m \rfloor}^2 \|\nabla f(x_k) - \nabla f_{i_k}(x_k)\|_*^2}_{R_4} \\
&+ \frac{1}{m} \underbrace{\sum_{k=0}^{K-1} \alpha_{\lfloor k/m \rfloor} \langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star \rangle}_{R_5} .
\end{aligned}
\tag{D.9}
$$

Denoting $\mathcal{V}_k^2 \triangleq \|\nabla f(x_k) - \nabla f_{i_k}(x_k)\|_*^2$, $\bar{\alpha} = \sum_{k=0}^{K-1} \alpha_{\lfloor k/m \rfloor}^2 = m \sum_{s=0}^{S-1} \alpha_s^2$, for R_4, by Jensen's inequality, we have

$$\mathbb{E}\left[\exp\left\{\frac{1}{\bar{\alpha}}\sum_{k=0}^{K-1} \alpha_{\lfloor k/m \rfloor}^2 \mathcal{V}_k^2/\sigma^2\right\}\right] \leq \frac{1}{\bar{\alpha}}\sum_{k=0}^{K-1} \alpha_{\lfloor k/m \rfloor}^2 \mathbb{E}\left[\exp\left\{\mathcal{V}_k^2/\sigma^2\right\}\right] \overset{(\star)}{\leq} \exp\{1\},$$

where $(\star)$ uses the additional assumption $\mathbb{E}_{i_k}\left[\exp\left\{\mathcal{V}_k^2/\sigma^2\right\}\right] \leq \exp\{1\}$.

Then, based on Markov's inequality, we have for any $\Lambda \geq 0$,

$$\text{Prob}\left\{\left\{\exp\left\{\frac{1}{\bar{\alpha}}\sum_{k=0}^{K-1} \alpha_{\lfloor k/m \rfloor}^2 \mathcal{V}_k^2/\sigma^2\right\} \geq \exp\{\Lambda + 1\}\right\}\right\} \leq \exp\{-\Lambda\},$$

$$\text{Prob}\left\{\left\{R_4 \geq (\Lambda + 1)\sigma^2 m \sum_{s=0}^{S-1} \alpha_s^2\right\}\right\} \leq \exp\{-\Lambda\}. \tag{D.10}$$

For R_5, since we have $\mathbb{E}_{i_k}\left[\alpha_{\lfloor k/m \rfloor}\left\langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star\right\rangle\right] = 0$ and

$$\mathbb{E}_{i_k}\left[\exp\left\{\frac{\alpha_{\lfloor k/m \rfloor}^2 \left\langle \nabla f(x_k) - \nabla f_{i_k}(x_k), z_k - x^\star\right\rangle^2}{\alpha_{\lfloor k/m \rfloor}^2 \sigma^2 D_X^2}\right\}\right] \leq \mathbb{E}_{i_k}\left[\exp\left\{\mathcal{V}_k^2/\sigma^2\right\}\right] \leq \exp\{1\},$$

which is based on the "light tail" assumption, using Lemma 7, we obtain

$$\text{Prob}\left\{\left\{R_5 \geq \Lambda\sigma D_X \sqrt{m\sum_{s=0}^{S-1} \alpha_s^2}\right\}\right\} \leq \exp\{-\Lambda^2/3\}. \tag{D.11}$$

Combining (D.9), (D.10) and (D.11), based on the parameter setting and using the notation

$$\mathcal{K}_0(m) \triangleq \frac{6LmV_d(x^\star, x_0)}{(K+m)^2} + \frac{8\sqrt{2V_d(x^\star, x_0)}\sqrt{\sigma^2 + M^2}}{\sqrt{K+m}},$$

$$R_6 \triangleq \frac{12L\sigma^2}{\lambda_1(S+1)^2}\sum_{s=0}^{S-1} \alpha_s^2 + \frac{4L\sigma D_X}{\lambda_1(S+1)^2\sqrt{m}}\sqrt{\sum_{s=0}^{S-1} \alpha_s^2},$$

we conclude that

$$\text{Prob}\left\{\left\{F(\tilde{x}_S) - F(x^\star) \leq \mathcal{K}_0(m) + \Lambda R_6\right\}\right\} \geq 1 - (\exp\{-\Lambda^2/3\} + \exp\{-\Lambda\}).$$

For R_6, using the choice of α_s and λ_1, we obtain

$$\begin{aligned} R_6 &\le \frac{4\sqrt{6}\sigma D_X}{3\sqrt{K+m}} + \frac{8\lambda_1\sigma^2(S+1)}{L} \\ &\le \frac{4\sqrt{6}\sigma D_X}{3\sqrt{K+m}} + \frac{4\sqrt{2V_d(x^\star, x_0)}\sigma^2}{\sqrt{K+m}\sqrt{\sigma^2+M^2}} \\ &\le \frac{4\sqrt{6}\sigma\big(\sqrt{3V_d(x^\star, x_0)} + D_X\big)}{3\sqrt{K+m}}, \end{aligned}$$

which completes the proof.

D.6 Proof of Theorem 14

Using Assumption (c), Lemma 4 with

$$\begin{cases} x = x_k^{j_k} \\ z = z_k \\ z^+ = z_{k+1} \\ y = \phi_{j_k}^k \\ y^+ = \phi_{j_k}^{k+1} \\ \alpha = \alpha_k \\ \beta = \beta_k \end{cases},$$

and taking expectation, if $\alpha_k(1-\beta_k) < 1/L$, we have

$$\begin{aligned} &\frac{1}{1-\beta_k}\mathbb{E}_{i_k,j_k}\big[F(\phi_{j_k}^{k+1}) - F(x^\star)\big] + \frac{1}{\alpha_k}\mathbb{E}_{i_k,j_k}\big[V_d(x^\star, z_{k+1})\big] \\ &\le \frac{\beta_k}{1-\beta_k}\mathbb{E}_{j_k}\big[F(\phi_{j_k}^k) - F(x^\star)\big] + \frac{1}{\alpha_k}V_d(x^\star, z_k) + \frac{(\sigma+M)^2}{2(\alpha_k^{-1} - L(1-\beta_k))}. \end{aligned} \tag{D.12}$$

Note that

$$\begin{aligned} &\mathbb{E}_{i_k,j_k}\big[F(\phi_{j_k}^{k+1}) - F(x^\star)\big] \\ &= \mathbb{E}_{i_k,j_k}\Big[\sum_{j=1}^m \big(F(\phi_j^{k+1}) - F(x^\star)\big) - \sum_{j\ne j_k}^m \big(F(\phi_j^k) - F(x^\star)\big)\Big] \\ &= \mathbb{E}_{i_k,j_k}\Big[\sum_{j=1}^m \big(F(\phi_j^{k+1}) - F(x^\star)\big)\Big] - \mathbb{E}_{j_k}\Big[\sum_{j\ne j_k}^m \big(F(\phi_j^k) - F(x^\star)\big)\Big]. \end{aligned}$$

Dividing (D.12) by m and then adding $\frac{1}{(1-\beta_k)m}\mathbb{E}_{j_k}\big[\sum_{j\neq j_k}^{m}\big(F(\phi_j^k)-F(x^\star)\big)\big]$ to both sides, we obtain

$$\begin{aligned}
&\frac{1}{1-\beta_k}\mathbb{E}_{i_k,j_k}\big[\frac{1}{m}\sum_{j=1}^{m}F(\phi_j^{k+1})-F(x^\star)\big]+\frac{1}{\alpha_k m}\mathbb{E}_{i_k,j_k}\big[V_d(x^\star,z_{k+1})\big]\\
\leq\ &-\frac{1}{m}\mathbb{E}_{j_k}\big[F(\phi_{j_k}^k)-F(x^\star)\big]+\frac{1}{1-\beta_k}\left(\frac{1}{m}\sum_{j=1}^{m}F(\phi_j^k)-F(x^\star)\right)+\frac{1}{\alpha_k m}V_d(x^\star,z_k)\\
&+\frac{(\sigma+M)^2}{2m(\alpha_k^{-1}-L(1-\beta_k))}\\
=\ &\frac{1-\frac{1-\beta_k}{m}}{1-\beta_k}\left(\frac{1}{m}\sum_{j=1}^{m}F(\phi_j^k)-F(x^\star)\right)+\frac{1}{\alpha_k m}V_d(x^\star,z_k)\\
&+\frac{(\sigma+M)^2}{2m(\alpha_k^{-1}-L(1-\beta_k))}.
\end{aligned}\tag{D.13}$$

It can be verified that with our parameters choice: $\beta_k=\frac{k/m}{k/m+2}$ and $\alpha_k=\frac{\lambda_2}{L(1-\beta_k)}$, the following holds for $k\geq 0$,

$$\alpha_{k+1}\frac{1-\frac{1-\beta_{k+1}}{m}}{1-\beta_{k+1}}\leq\frac{\alpha_k}{1-\beta_k}\text{ and }\beta_0=0.$$

Then, we can telescope (D.13) from $k=K-1,\ldots,0$, which results in

$$\begin{aligned}
&\frac{\alpha_{K-1}}{1-\beta_{K-1}}\mathbb{E}\left[\frac{1}{m}\sum_{j=1}^{m}F(\phi_j^K)-F(x^\star)\right]+\frac{1}{m}\mathbb{E}\left[V_d(x^\star,z_K)\right]\\
\leq\ &\frac{\lambda_2(m-1)}{Lm}\big(F(x_0)-F(x^\star)\big)+\frac{1}{m}V_d(x^\star,x_0)+\sum_{k=0}^{K-1}\frac{\alpha_k(\sigma+M)^2}{2m(\alpha_k^{-1}-L(1-\beta_k))}.
\end{aligned}$$

Using the definition of $\bar{\phi}^K$ and convexity, we obtain

$$
\begin{aligned}
&\mathbb{E}\left[F(\bar{\phi}^K) - F(x^\star)\right] \\
\le\ & \frac{1-\beta_{K-1}}{\alpha_{K-1}}\left(\frac{\lambda_2(m-1)}{Lm}\big(F(x_0)-F(x^\star)\big) + \frac{1}{m}V_d(x^\star, x_0)\right) \\
&+ \frac{1-\beta_{K-1}}{\alpha_{K-1}}\sum_{k=0}^{K-1}\frac{\alpha_k(\sigma+M)^2}{2m(\alpha_k^{-1} - L(1-\beta_k))} \\
\overset{(a)}{=}\ & \frac{4(m-1)\big(F(x_0)-F(x^\star)\big)}{m\left(\frac{K-1}{m}+2\right)^2} + \frac{4LV_d(x^\star, x_0)}{\lambda_2 m\left(\frac{K-1}{m}+2\right)^2} \\
&+ \frac{3\lambda_2(\sigma+M)^2}{2Lm\left(\frac{K-1}{m}+2\right)^2}\sum_{k=0}^{K-1}\left(\frac{k}{m}+2\right)^2 \\
\overset{(b)}{\le}\ & \frac{4(m-1)\big(F(x_0)-F(x^\star)\big)}{m\left(\frac{K-1}{m}+2\right)^2} + \frac{4LV_d(x^\star, x_0)}{\lambda_2 m\left(\frac{K-1}{m}+2\right)^2} + \frac{4\lambda_2(\sigma+M)^2\left(\frac{K-1}{m}+2\right)}{L},
\end{aligned}
\tag{D.14}
$$

where (a) uses $\lambda_2 \le \frac{2}{3}$, (b) follows from simple integration arguments and that $\frac{K}{m}+2 \le 2\left(\frac{K-1}{m}+2\right)$ since $K \ge 1, m \ge 1$.

Based on the choice of

$$
\lambda_2 = \min\left\{\frac{2}{3}, \frac{L\sqrt{V_d(x^\star, x_0)}}{\sqrt{m}(\sigma+M)\left(\frac{K-1}{m}+2\right)^{\frac{3}{2}}}\right\},
$$

(D.14) can be further upper bounded as

$$
\begin{aligned}
&\mathbb{E}\left[F(\bar{\phi}^K) - F(x^\star)\right] \\
\le\ & \frac{4(m-1)\big(F(x_0)-F(x^\star)\big)}{m\left(\frac{K-1}{m}+2\right)^2} + \frac{6LV_d(x^\star, x_0)}{m\left(\frac{K-1}{m}+2\right)^2} + \frac{8\sqrt{V_d(x^\star, x_0)}(\sigma+M)}{m^{\frac{1}{2}}\left(\frac{K-1}{m}+2\right)^{\frac{1}{2}}}.
\end{aligned}
$$

D.7 Experimental Setup

All of our experiments were conducted using PyTorch [118] library.

D.7.1 Classification Setup

CIFAR-10 & CIFAR-100 Our implementation (e.g., ResNet and DenseNet implementations, data pre-processing) generally follows the repository `https://github.`

`com/kuangliu/pytorch-cifar`. All the CIFAR experiments used a single GPU in a mix of RTX2080Ti, TITAN Xp and TITAN V. The batch size is fixed to 128. We used cross-entropy loss with 0.0005 weight decay and used batch normalization [61]. Data augmentation includes random 32-pixel crops with a padding of 4-pixel and random horizontal flips with 0.5 probability. We used step (or multi-step) learning rate scheduler with a decay factor 10. For the CIFAR-10 experiments, we trained 90 epochs and decayed the learning rate every 30 epochs. For the CIFAR-100 experiments, we trained 300 epochs and decayed the learning rate at 150 epoch and 225 epoch following the settings in DenseNet [60].

ImageNet In the ImageNet experiments, we tried both ResNet50 and ResNet152 [55]. The training strategy is the same as the PyTorch's official repository `https://github.com/pytorch/examples/tree/master/imagenet`, which uses a batch size of 256. The learning rate starts at 0.1 and decays by a factor of 10 every 30 epochs. Also, we applied weight decay with 0.0001 decay rate to the model during the training. For the data augmentation, we applied random 224-pixel crops and random horizontal flips with 0.5 probability. Here, we ran all experiments across 8 NVIDIA P100 GPUs for 90 epochs.

D.7.2 Language Model Setup

We followed the implementation in the repository `https://github.com/salesforce/awd-lstm-lm` and trained word level Penn Treebank with LSTM without fine-tuning or continuous cache pointer augmentation for 750 epochs. The experiments were conducted on a single RTX2080Ti. We used the default hyper-parameter tuning except for learning rate and momentum: The LSTM has 3 layers containing 1150 hidden units each, embedding size is 400, gradient clipping has a maximum norm 0.25, batch size is 80, using variable sequence length, dropout for the layers has probability 0.4, dropout for the RNN layers has probability 0.3, dropout for the input embedding layer has probability 0.65, dropout to remove words from embedding layer has probability 0.1, weight drop [100] has probability 0.5, the amount of $\ell 2$-regularization on the RNN activation is 2.0, the amount of slowness regularization applied on the RNN activation is 1.0 and all weights receive a weight decay of 0.0000012.

Appendix E

Appendix for Chapter 6

E.1 Proof of Theorem 15

Let $\alpha_i \triangleq \beta^{t-i}(1-\beta)$ and denote $x'_{t+1} \triangleq x_{t+1} + \zeta b_{t+1}$ and $\mathcal{G}_t \triangleq \sigma(g_1, \ldots, g_t), \forall t$. Clearly, the random variables $m_t, x_t, x_{t+1}, \eta_t$ are $\mathcal{G}_t$-measurable. Note that

$$m_t = \beta^K m_{t-K} + \sum_{i=t-K+1}^{t} \alpha_i g_i.$$

Conceptually, if we choose K to be sufficiently large, the term $\beta^K m_{t-K}$ is negligible. Then, if all the points $y_{t-K+1}, \ldots, y_t$ are inside $x_{t-K} + \delta\mathbb{B}$, we have that m_t approximately belongs to $\partial_\delta f(x_{t-K})$ in expectation.

Note that for all $i = t-K+1, \ldots, t$,

$$\begin{aligned}
\|y_i - x_{t-K}\| &\leq \|y_i - x_{i-1}\| + \|x_{i-1} - x_{t-K}\| \\
&\overset{(a)}{\leq} \|x'_i - x_{i-1}\| + \|x_{i-1} - x_{t-K}\| \\
&= \|\zeta b_i - \eta_{i-1} m_{i-1}\| + \left\| \sum_{j=t-K}^{i-2} \eta_j m_j \right\| \\
&\overset{(b)}{\leq} \zeta + \sum_{j=t-K}^{i-1} \eta_j \|m_j\| \\
&\overset{(c)}{\leq} \frac{\omega}{p} + \frac{i-t+K}{p} \\
&\leq \frac{K+\omega}{p},
\end{aligned}$$

where (a) holds since y_i is sampled from the line segment $[x_{i-1}, x_i']$, (b) uses $\|b_i\| \le 1$ and (c) follows from $\zeta \le \frac{\omega}{p}$ and $\eta_t \|m_t\| \le \frac{1}{p}, \forall t$. We verify that the choices of K, ω and p satisfy $\frac{K+\omega}{p} \le \delta$:

$$\frac{K+\omega}{p} = \frac{\frac{1}{1-\beta} \ln \frac{16G}{\epsilon}}{\frac{64G^2}{\delta\epsilon^2} \ln \frac{16G}{\epsilon}} = \delta.$$

Then, conditioned on $\mathcal{G}_{t-K}$, since for all $i = t-K+1, \ldots, t$,

$$\mathbb{E}\big[g_i \mid \mathcal{G}_{t-K}\big] = \mathbb{E}\big[\nabla f(y_i) \mid \mathcal{G}_{t-K}\big] \in \partial_\delta f(x_{t-K}),$$

we have (note that $\sum_{i=t-K+1}^{t} \alpha_i = 1 - \beta^K$)

$$\begin{aligned}
&\frac{1}{1-\beta^K} \sum_{i=t-K+1}^{t} \alpha_i \mathbb{E}\big[g_i \mid \mathcal{G}_{t-K}\big] \in \partial_\delta f(x_{t-K}) \\
\Rightarrow\ &\frac{1}{1-\beta^K} \left(\mathbb{E}\big[m_t \mid \mathcal{G}_{t-K}\big] - \beta^K m_{t-K}\right) \in \partial_\delta f(x_{t-K}) \\
\Rightarrow\ &\mathrm{dist}(0, \partial_\delta f(x_{t-K})) \le \frac{1}{1-\beta^K} \left(\left\|\mathbb{E}\big[m_t \mid \mathcal{G}_{t-K}\big]\right\| + \beta^K \|m_{t-K}\|\right) \\
&\qquad\qquad\qquad\qquad\quad \le \frac{1}{1-\beta^K} \left(\mathbb{E}\big[\, \|m_t\| \mid \mathcal{G}_{t-K}\big] + \beta^K \|m_{t-K}\|\right).
\end{aligned}$$

Take expectation.

$$\mathbb{E}\big[\mathrm{dist}(0, \partial_\delta f(x_{t-K}))\big] \le \frac{1}{1-\beta^K} \mathbb{E}\big[\, \|m_t\|\,\big] + \frac{\beta^K G}{1-\beta^K},$$

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\big[\mathrm{dist}(0, \partial_\delta f(x_{t-K}))\big] \le \frac{1}{(1-\beta^K)T} \sum_{t=1}^{T} \mathbb{E}\big[\, \|m_t\|\,\big] + \frac{\beta^K G}{1-\beta^K}.$$

We verify that the choices of β and K satisfy $\beta^K G \le \frac{\epsilon}{16}$: $\left(\beta^K \le \frac{\epsilon}{16G}\right) \Leftrightarrow \left(K \ge \frac{1}{\ln \frac{1}{\beta}} \ln \frac{16G}{\epsilon}\right)$. Without loss of generality, we assume that $\epsilon \le G$, and thus $\beta^K \le \frac{1}{16}$. The above inequality can be further bounded as

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\big[\mathrm{dist}(0, \partial_\delta f(x_{t-K}))\big] \le \frac{16}{15T} \sum_{t=1}^{T} \mathbb{E}\big[\, \|m_t\|\,\big] + \frac{\epsilon}{15}. \tag{E.1}$$

The remaining proof is to show Algorithm 14 ensures $\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\big[\, \|m_t\|\,\big] = O(\epsilon)$.

For clarity, we let $\mathcal{Y}_{t+1} \triangleq \sigma(g_1, \ldots, g_t, b_{t+1}, y_{t+1})$ and $\hat{\mathcal{Y}}_{t+1} \triangleq \sigma(g_1, \ldots, g_t, b_{t+1})$. Clearly, we have $\mathcal{G}_t \subset \hat{\mathcal{Y}}_{t+1} \subset \mathcal{Y}_{t+1} \subset \mathcal{G}_{t+1}$. Let $\varphi(\lambda) \triangleq (1-\lambda)x_t + \lambda x'_{t+1}$ for $\lambda \in [0,1]$. Since y_{t+1} is uniformly sampled from the line segment $[x_t, x'_{t+1}]$ and that f is differentiable at y_{t+1} almost surely, it holds that

$$\begin{aligned}\mathbb{E}\big[\,\langle g_{t+1}, x'_{t+1} - x_t\rangle \mid \mathcal{G}_t\big] &= \mathbb{E}\big[\mathbb{E}\big[\mathbb{E}\big[\,\langle g_{t+1}, x'_{t+1} - x_t\rangle \mid \mathcal{Y}_{t+1}\big] \mid \hat{\mathcal{Y}}_{t+1}\big] \mid \mathcal{G}_t\big] \\ &= \mathbb{E}\big[\mathbb{E}\big[\,\langle \nabla f(y_{t+1}), x'_{t+1} - x_t\rangle \mid \hat{\mathcal{Y}}_{t+1}\big] \mid \mathcal{G}_t\big] \\ &= \mathbb{E}\Big[\int_0^1 f'(\varphi(\lambda); x'_{t+1} - x_t)\mathrm{d}\lambda \mid \mathcal{G}_t\Big] \\ &= \mathbb{E}\big[f(x'_{t+1}) - f(x_t) \mid \mathcal{G}_t\big].\end{aligned} \tag{E.2}$$

By $x'_{t+1} - x_t = -\eta_t m_t + \zeta b_{t+1}$, we have

$$\begin{aligned}\mathbb{E}\big[\,\langle g_{t+1}, x'_{t+1} - x_t\rangle \mid \mathcal{G}_t\big] &= -\eta_t \mathbb{E}\big[\,\langle g_{t+1}, m_t\rangle \mid \mathcal{G}_t\big] + \zeta \mathbb{E}\big[\,\langle g_{t+1}, b_{t+1}\rangle \mid \mathcal{G}_t\big] \\ &\le -\eta_t \mathbb{E}\big[\,\langle g_{t+1}, m_t\rangle \mid \mathcal{G}_t\big] + \zeta G,\end{aligned}$$

where we used $\|b_{t+1}\| \le 1$. Thus, combining with (E.2), we obtain

$$\begin{aligned}\mathbb{E}\big[\,\langle g_{t+1}, m_t\rangle \mid \mathcal{G}_t\big] &\le \frac{1}{\eta_t}\mathbb{E}\big[f(x_t) - f(x_{t+1}) + f(x_{t+1}) - f(x'_{t+1}) \mid \mathcal{G}_t\big] + \frac{\zeta}{\eta_t}G \\ &\le \frac{1}{\eta_t}\big(f(x_t) - f(x_{t+1})\big) + \frac{\zeta}{\eta_t}(L_0 + G).\end{aligned} \tag{E.3}$$

Based on the construction $m_{t+1} = \beta m_t + (1-\beta) g_{t+1}$, we can conclude that

$$\begin{aligned}\|m_{t+1}\|^2 &= \beta^2 \|m_t\|^2 + 2\beta(1-\beta)\langle g_{t+1}, m_t\rangle + (1-\beta)^2 \|g_{t+1}\|^2, \\ \mathbb{E}\big[\eta_t\big(\|m_{t+1}\|^2 - \beta^2\|m_t\|^2\big)\big] &= 2\beta(1-\beta)\mathbb{E}\big[\eta_t \langle g_{t+1}, m_t\rangle\big] + (1-\beta)^2\mathbb{E}\big[\eta_t \|g_{t+1}\|^2\big].\end{aligned}$$

From (E.3), it holds that

$$\begin{aligned}\mathbb{E}\big[\eta_t\big(\|m_{t+1}\|^2 - \beta^2\|m_t\|^2\big)\big] &\le 2\beta(1-\beta)\mathbb{E}\big[f(x_t) - f(x_{t+1})\big] \\ &\quad + 2\beta(1-\beta)(L_0 + G)\zeta + (1-\beta)^2\mathbb{E}\big[\eta_t\|g_{t+1}\|^2\big], \\ \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\eta_t\big(\|m_{t+1}\|^2 - \beta^2\|m_t\|^2\big)\big] &\le \frac{2\beta(1-\beta)\Delta}{T} + 2\beta(1-\beta)(L_0 + G)\zeta \\ &\quad + \frac{(1-\beta)^2 G^2}{q},\end{aligned}$$

where we used $\eta_t \le \frac{1}{q}$.

Since $\eta_t = \frac{1}{p\|m_t\|+q}$, using the same telescoping proof in [155], as long as $\frac{pG}{q} \leq \frac{\beta}{2}$, the following holds

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\eta_t\big(\|m_{t+1}\|^2 - \beta^2\|m_t\|^2\big)\right] \geq \frac{\beta(1-\beta)}{2T}\sum_{t=1}^{T+1}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right] - \frac{\beta G^2}{qT}.$$

Thus,

$$\frac{\beta(1-\beta)}{2T}\sum_{t=1}^{T+1}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right] \leq \frac{2\beta(1-\beta)\Delta}{T} + 2\beta(1-\beta)(L_0+G)\zeta + \frac{(1-\beta)^2G^2}{q} + \frac{\beta G^2}{qT},$$

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{q\|m_t\|^2}{p\|m_t\|+q}\right] \leq \frac{4q\Delta}{T} + 4q(L_0+G)\zeta + \frac{2(1-\beta)G^2}{\beta} + \frac{2G^2}{T(1-\beta)}.$$

Comparing the above inequality with (14)[1] in [155], we notice that the only difference is the additional perturbation term $4q(L_0+G)\zeta$. Since we choose the identical β, p, q and T as in [155], using the arguments (15) and (16) in [155] and denoting $m_{avg} \triangleq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|m_t\|\big]$, we obtain

$$\frac{4Gm_{avg}^2}{m_{avg}+4G} \leq \frac{\epsilon^2}{17} + 4q(L_0+G)\zeta \overset{(\star)}{\leq} \frac{\epsilon^2}{15},$$

where $(\star)$ uses $\zeta \leq \frac{\epsilon^2}{510q(L_0+G)}$. The above is a quadratic equation in m_{avg}:

$$4Gm_{avg}^2 - \frac{\epsilon^2}{15}m_{avg} - \frac{4G\epsilon^2}{15} \leq 0.$$

Solving for the positive root of this quadratic equation and using $\epsilon \leq G$, we obtain

$$m_{avg} \leq \frac{\frac{\epsilon^2}{15} + \sqrt{\frac{\epsilon^4}{225} + \frac{64G^2\epsilon^2}{15}}}{8G} \leq \frac{4\epsilon}{15} \leq \frac{5\epsilon}{16}.$$

Finally, using (E.1), we conclude that

$$\mathbb{E}\big[\mathrm{dist}(0, \partial_\delta f(x_{out}))\big] = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\mathrm{dist}(0, \partial_\delta f(x_{t-K}))\big] \leq \frac{2\epsilon}{5}.$$

Thus, with probability at least $\frac{3}{5}$, we have $\mathrm{dist}(0, \partial_\delta f(x_{out})) \leq \epsilon$.

[1]There is a typo in the telescoping proof of Theorem 14 in [155]: The term $\frac{\beta^2G^2}{q}$ above Equation (14) should be $\frac{\beta G^2}{q}$. This typo does not affect the final convergence result.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016.

[2] Z. Allen-Zhu. How to make the gradients small stochastically: Even faster convex and nonconvex sgd. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 1157–1167, 2018.

[3] Z. Allen-Zhu. Katyusha X: Simple Momentum Method for Stochastic Sum-of-Nonconvex Optimization. In *International Conference on Machine Learning (ICML)*, pages 179–185, 2018.

[4] Z. Allen-Zhu. Katyusha: The First Direct Acceleration of Stochastic Gradient Methods. *J. Mach. Learn. Res.*, 18(221):1–51, 2018.

[5] Z. Allen-Zhu and L. Orecchia. Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent. In *Innovations in Theoretical Computer Science (ITCS)*, 2017.

[6] Z. Allen-Zhu and Y. Yuan. Improved SVRG for Non-Strongly-Convex or Sum-of-Non-Convex Objectives. In *International Conference on Machine Learning (ICML)*, pages 1080–1089, 2016.

[7] Z. Allen-Zhu, Y. Li, R. M. de Oliveira, and A. Wigderson. Much Faster Algorithms for Matrix Scaling. In C. Umans, editor, *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 890–901, 2017.

[8] H. Attouch and J. Peypouquet. The rate of convergence of nesterov's accelerated forward-backward method is actually faster than 1/kˆ2. *SIAM J. Optim.*, 26(3):1824–1834, 2016.

[9] A. Auslender and M. Teboulle. Interior gradient and proximal methods for convex and conic optimization. *SIAM J. Optim.*, 16(3):697–725, 2006.

[10] N. Bansal and A. Gupta. Potential-Function Proofs for Gradient Methods. *Theory Comput.*, 15(4):1–32, 2019.

[11] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces.* Springer, New York, 2017. Second edition.

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

[13] M. Belkin, S. Ma, and S. Mandal. To Understand Deep Learning We Need to Understand Kernel Learning. In *International Conference on Machine Learning (ICML)*, pages 541–549, 2018.

[14] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization.* Society for Industrial and Applied Mathematics, 2013.

[15] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *SIAM Review*, 60(2):223–311, 2018.

[16] S. Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[17] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Lower bounds for finding stationary points i. *Math. Program.*, pages 1–50, 2019.

[18] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Lower bounds for finding stationary points ii: first-order methods. *Math. Program.*, 185(1-2), 2021.

[19] A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[20] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[21] F. H. Clarke. *Optimization and Nonsmooth Analysis.* SIAM, 1990.

[22] M. B. Cohen, A. Madry, D. Tsipras, and A. Vladu. Matrix Scaling and Balancing via Box Constrained Newton's Method and Interior Point Methods. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 902–913. IEEE, 2017.

[23] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995.

[24] S. Cyrus, B. Hu, B. Van Scoy, and L. Lessard. A robust accelerated optimization algorithm for strongly convex functions. In *American Control Conference (ACC)*, pages 1376–1381. IEEE, 2018.

[25] A. d'Aspremont, D. Scieur, and A. Taylor. Acceleration methods. *arXiv preprint arXiv:2101.09545*, 2021.

[26] D. Davis and D. Drusvyatskiy. Complexity of finding near-stationary points of convex functions stochastically. *arXiv preprint arXiv:1802.08556*, 2018.

[27] A. Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 676–684, 2016.

[28] A. Defazio. On the Curved Geometry of Accelerated Optimization. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, volume 32, pages 1764–1773, 2019.

[29] A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 1646–1654, 2014.

[30] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Math. Program.*, 146(1):37–75, 2014.

[31] J. Diakonikolas and C. Guzmán. Complementary Composite Minimization, Small Gradients in General Norms, and Applications to Regression Problems. *arXiv preprint arXiv:2101.11041*, 2021.

[32] J. Diakonikolas and P. Wang. Potential Function-based Framework for Making the Gradients Small in Convex and Min-Max Optimization. *arXiv preprint arXiv:2101.12101*, 2021.

[33] D. Driggs, M. J. Ehrhardt, and C. Schönlieb. Accelerating variance-reduced stochastic gradient methods. *Math. Program.*, 191(2):671–715, 2022.

[34] Y. Drori. The exact information-based complexity of smooth convex minimization. *J. Complex.*, 39:1–16, 2017.

[35] Y. Drori. On the properties of convex functions over open sets. *arXiv preprint arXiv:1812.02419*, 2018.

[36] Y. Drori and A. Taylor. On the oracle complexity of smooth strongly convex minimization. *arXiv preprint arXiv:2101.09740*, 2021.

[37] Y. Drori and M. Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Math. Program.*, 145(1-2):451–482, 2014.

[38] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[39] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12(Jul):2121–2159, 2011.

[40] J. C. Duchi, M. I. Jordan, and H. B. McMahan. Estimation, Optimization, and Parallelism when Data is Sparse. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 2832–2840, 2013.

[41] C. Fang, Y. Huang, and Z. Lin. Accelerating asynchronous algorithms for convex optimization by momentum compensation. *arXiv preprint arXiv:1802.09747*, 2018.

[42] C. Fang, C. J. Li, Z. Lin, and T. Zhang. SPIDER: Near-Optimal Non-Convex Optimization via Stochastic Path-Integrated Differential Estimator. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 687–697, 2018.

[43] D. J. Foster, A. Sekhari, O. Shamir, N. Srebro, K. Sridharan, and B. Woodworth. The Complexity of Making the Gradient Small in Stochastic Convex Optimization. In *Annual Conference on Learning Theory (COLT)*, pages 1319–1345, 2019.

[44] R. Frostig, R. Ge, S. Kakade, and A. Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning (ICML)*, pages 2540–2548, 2015.

[45] R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping From Saddle Points — Online Stochastic Gradient for Tensor Decomposition. In *Annual Conference on Learning Theory (COLT)*, pages 797–842, 2015.

[46] S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. *SIAM J. Optim.*, 22(4):1469–1492, 2012.

[47] S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.*, 23(4):2341–2368, 2013.

[48] S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Math. Program.*, 156(1-2):59–99, 2016.

[49] A. Goldstein. Optimization of Lipschitz continuous functions. *Math. Program.*, 13(1):14–22, 1977.

[50] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[51] B. Gu, W. Xian, Z. Huo, C. Deng, and H. Huang. A Unified q-Memorization Framework for Asynchronous Stochastic Optimization. *J. Mach. Learn. Res.*, 21:190–1, 2020.

[52] R. Hannah, Y. Liu, D. O'Connor, and W. Yin. Breaking the span assumption yields fast finite-sum minimization. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 2312–2321, 2018.

[53] R. Hannah, F. Feng, and W. Yin. A2BCD: Asynchronous Acceleration with Optimal Complexity. In *International Conference on Learning Representations (ICLR)*, 2019.

[54] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016.

[55] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition Conference (CVPR)*, pages 770–778, 2016.

[56] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[57] B. Hu and L. Lessard. Dissipativity Theory for Nesterov's Accelerated Method. In *International Conference on Machine Learning (ICML)*, pages 1549–1557, 2017.

[58] B. Hu, S. Wright, and L. Lessard. Dissipativity Theory for Accelerating Stochastic Variance Reduction: A Unified Analysis of SVRG and Katyusha Using Semidefinite Programs. In *International Conference on Machine Learning (ICML)*, pages 2038–2047, 2018.

[59] C. Hu, W. Pan, and J. T. Kwok. Accelerated gradient methods for stochastic optimization and online learning. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 781–789, 2009.

[60] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition Conference (CVPR)*, pages 4700–4708, 2017.

[61] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[62] M. Ito and M. Fukuda. Nearly optimal first-order methods for convex optimization under gradient norm measure: An adaptive regularization approach. *J. Optim. Theory Appl.*, 188(3):770–804, 2021.

[63] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan. How to Escape Saddle Points Efficiently. In *International Conference on Machine Learning (ICML)*, pages 1724–1732, 2017.

[64] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 315–323, 2013.

[65] P. Joulani, A. György, and C. Szepesvári. Think out of the "Box": Generically-Constrained Asynchronous Composite Optimization and Hedging. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 12225–12235, 2019.

[66] Y. Juan, Y. Zhuang, W. Chin, and C. Lin. Field-aware Factorization Machines for CTR Prediction. In *ACM Conference on Recommender Systems (RecSys)*, pages 43–50. ACM, 2016.

[67] S. S. Keerthi, D. DeCoste, and T. Joachims. A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. *J. Mach. Learn. Res.*, 6:341–361, 2005.

[68] D. Kim and J. A. Fessler. Optimized first-order methods for smooth convex minimization. *Math. Program.*, 159(1-2):81–107, 2016.

[69] D. Kim and J. A. Fessler. Generalizing the optimized gradient method for smooth convex minimization. *SIAM J. Optim.*, 28(2):1920–1950, 2018.

[70] D. Kim and J. A. Fessler. Another look at the fast iterative shrinkage/thresholding algorithm (FISTA). *SIAM J. Optim.*, 28(1):223–250, 2018.

[71] D. Kim and J. A. Fessler. Optimizing the Efficiency of First-Order Methods for Decreasing the Gradient of Smooth Convex Functions. *J. Optim. Theory Appl.*, 188(1):192–219, 2021.

[72] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[73] J. Konečnỳ and P. Richtárik. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

[74] G. Kornowski and O. Shamir. Oracle Complexity in Nonsmooth Nonconvex Optimization. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 324–334, 2021.

[75] D. Kovalev, S. Horváth, and P. Richtárik. Don't jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop. In *Algorithmic Learning Theory (ALT)*, pages 451–467. PMLR, 2020.

[76] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[77] A. Kulunchakov and J. Mairal. Estimate Sequences for Variance-Reduced Stochastic Composite Optimization. In *International Conference on Machine Learning (ICML)*, pages 3541–3550, 2019.

[78] G. Lan. An optimal method for stochastic composite optimization. *Math. Program.*, 133(1-2):365–397, 2012.

[79] G. Lan and Y. Zhou. An optimal randomized incremental gradient method. *Math. Program.*, 171(1-2):167–215, 2018.

[80] G. Lan, A. Nemirovski, and A. Shapiro. Validation analysis of mirror descent stochastic approximation method. *Math. Program.*, 134(2):425–458, 2012.

[81] G. Lan, Z. Li, and Y. Zhou. A unified variance-reduced accelerated gradient method for convex optimization. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 10462–10472, 2019.

[82] R. Leblond, F. Pedregosa, and S. Lacoste-Julien. ASAGA: Asynchronous Parallel SAGA. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54, pages 46–54, 2017.

[83] R. Leblond, F. Pedregosa, and S. Lacoste-Julien. Improved Asynchronous Parallel Optimization Analysis for Stochastic Incremental Methods. *J. Mach. Learn. Res.*, 19:81:1–81:68, 2018.

[84] J. Lee, C. Park, and E. K. Ryu. A Geometric Structure of Acceleration and Its Role in Making Gradients Small Fast. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 11999–12012, 2021.

[85] L. Lessard and P. J. Seiler. Direct Synthesis of Iterative Algorithms With Bounds on Achievable Worst-Case Convergence Rate. In *American Control Conference (ACC)*, pages 119–125, 2020.

[86] L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM J. Optim.*, 26(1):57–95, 2016.

[87] D. D. Lewis, Y. Yang, T. Russell-Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *J. Mach. Learn. Res.*, 5:361–397, 2004.

[88] B. Li, M. Ma, and G. B. Giannakis. On the Convergence of SARAH and Beyond. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 223–233, 2020.

[89] Z. Li. ANITA: An Optimal Loopless Accelerated Variance-Reduced Gradient Method. *arXiv preprint arXiv:2103.11333*, 2021.

[90] X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous Parallel Stochastic Gradient for Nonconvex Optimization. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 2737–2745, 2015.

[91] H. Lin, J. Mairal, and Z. Harchaoui. A Universal Catalyst for First-Order Optimization. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 3366–3374, 2015.

[92] Q. Lin and L. Xiao. An Adaptive Accelerated Proximal Gradient Method and its Homotopy Continuation for Sparse Optimization. In *International Conference on Machine Learning (ICML)*, pages 73–81, 2014.

[93] Q. Lin, Z. Lu, and L. Xiao. An accelerated proximal coordinate gradient method. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 3059–3067, 2014.

[94] I. Loshchilov and F. Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[95] J. Lucas, S. Sun, R. Zemel, and R. Grosse. Aggregated Momentum: Stability Through Passive Damping. In *International Conference on Learning Representations (ICLR)*, 2019. URL `https://openreview.net/forum?id=Syxt5oC5YQ`.

[96] J. Ma and D. Yarats. Quasi-hyperbolic momentum and Adam for deep learning. In *International Conference on Learning Representations (ICLR)*, 2019. URL `https://openreview.net/forum?id=S1fUpoR5FQ`.

[97] S. Ma, R. Bassily, and M. Belkin. The Power of Interpolation: Understanding the Effectiveness of SGD in Modern Over-parametrized Learning. In *International Conference on Machine Learning (ICML)*, pages 3325–3334, 2018.

[98] H. Mania, X. Pan, D. S. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan. Perturbed Iterate Analysis for Asynchronous Stochastic Optimization. *SIAM J. Optim.*, 27(4):2202–2229, 2017.

[99] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. 1993.

[100] S. Merity, N. S. Keskar, and R. Socher. Regularizing and Optimizing LSTM Language Models. In *International Conference on Learning Representations (ICLR)*, 2018.

[101] A. Nazin, A. Nemirovsky, A. Tsybakov, and A. Juditsky. Algorithms of robust stochastic optimization based on mirror descent method. *Autom. Remote. Control.*, 80(9):1607–1627, 2019.

[102] A. Nemirovski and D. Yudin. *Problem complexity and method efficiency in optimization.* John Wiley, New York, 1983.

[103] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. Optim.*, 19(4): 1574–1609, 2009.

[104] Y. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547, 1983.

[105] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005.

[106] Y. Nesterov. How to make the gradients small. *Optima. Mathematical Optimization Society Newsletter*, (88):10–11, 2012.

[107] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[108] Y. Nesterov. Gradient methods for minimizing composite functions. *Math. Program.*, 140(1):125–161, 2013.

[109] Y. Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

[110] Y. Nesterov, A. Gasnikov, S. Guminov, and P. Dvurechensky. Primal–dual accelerated gradient methods with small-dimensional relaxation oracle. *Optim. Methods Softw.*, pages 1–38, 2020.

[111] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A Novel Method for Machine Learning Problems Using Stochastic Recursive Gradient. In *International Conference on Machine Learning (ICML)*, pages 2613–2621, 2017.

[112] L. M. Nguyen, P. H. Nguyen, M. van Dijk, P. Richtárik, K. Scheinberg, and M. Takác. SGD and Hogwild! Convergence Without the Bounded Gradients Assumption. In *International Conference on Machine Learning (ICML)*, volume 80, pages 3747–3755, 2018.

[113] A. Nitanda. Stochastic Proximal Gradient Descent with Acceleration Techniques. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 1574–1582, 2014.

[114] J. Nocedal, A. Sartenaer, and C. Zhu. On the Behavior of the Gradient Norm in the Steepest Descent Method. *Comput. Optim. Appl.*, 22(1):5–35, 2002.

[115] B. O'donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. *Found. Comput. Math.*, 15(3):715–732, 2015.

[116] C. Paquette and S. Vavasis. Potential-based analyses of first-order methods for constrained and composite optimization. *arXiv preprint arXiv:1903.08497*, 2019.

[117] N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[118] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[119] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[120] F. Pedregosa, R. Leblond, and S. Lacoste-Julien. Breaking the Nonsmooth Barrier: A Scalable Parallel Method for Composite Optimization. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 56–65, 2017.

[121] N. H. Pham, L. M. Nguyen, D. T. Phan, and Q. Tran-Dinh. ProxSARAH: An efficient algorithmic framework for stochastic composite nonconvex optimization. *J. Mach. Learn. Res.*, 21(110):1–48, 2020.

[122] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[123] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. & Math. Phys.*, 4(5):1–17, 1964.

[124] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, 1992.

[125] B. Recht, C. Ré, S. J. Wright, and F. Niu. Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 693–701, 2011.

[126] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat.*, pages 400–407, 1951.

[127] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control. Optim.*, 14(5):877–898, 1976.

[128] R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*, volume 317. Springer Science & Business Media, 2009.

[129] U. G. Rothblum and H. Schneider. Scalings of matrices which have prespecified row sums and column sums via optimization. *Linear Algebra Appl.*, 114:737–764, 1989.

[130] N. L. Roux, M. Schmidt, and F. R. Bach. A Stochastic Gradient Method with an Exponential Convergence Rate for Finite Training Sets. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 2663–2671, 2012.

[131] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[132] C. D. Sa, C. Zhang, K. Olukotun, and C. Ré. Taming the Wild: A Unified Analysis of Hogwild-Style Algorithms. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 2674–2682, 2015.

[133] M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162(1-2):83–112, 2017.

[134] B. V. Scoy, R. A. Freeman, and K. M. Lynch. The Fastest Known Globally Convergent First-Order Method for Minimizing Strongly Convex Functions. *IEEE Contr. Syst. Lett.*, 2(1):49–54, 2017.

[135] S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. 2007. Technical report, The Hebrew University.

[136] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *J. Mach. Learn. Res.*, 14(Feb):567–599, 2013.

[137] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *International Conference on Machine Learning (ICML)*, pages 64–72, 2014.

[138] C. Song, Y. Jiang, and Y. Ma. Variance Reduction via Accelerated Dual Averaging for Finite-Sum Optimization. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, volume 33, pages 833–844, 2020.

[139] S. U. Stich, A. Mohtashami, and M. Jaggi. Critical Parameters for Scalable Distributed Learning with Large Batches and Asynchronous Updates. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130, pages 4042–4050, 2021.

[140] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1139–1147, 2013.

[141] A. Taylor and F. Bach. Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions. In *Annual Conference on Learning Theory (COLT)*, pages 2934–2992, 2019.

[142] A. Taylor and Y. Drori. An optimal gradient method for smooth strongly convex minimization. *Math. Program.*, pages 1–38, 2022.

[143] A. Taylor, B. Van Scoy, and L. Lessard. Lyapunov Functions for First-Order Methods: Tight Automated Convergence Guarantees. In *International Conference on Machine Learning (ICML)*, pages 4897–4906, 2018.

[144] A. B. Taylor, J. M. Hendrickx, and F. Glineur. Exact worst-case performance of first-order methods for composite convex optimization. *SIAM J. Optim.*, 27 (3):1283–1313, 2017.

[145] A. B. Taylor, J. M. Hendrickx, and F. Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Math. Program.*, 161(1-2):307–345, 2017.

[146] L. Tian, K. Zhou, and A. M.-C. So. On the Finite-Time Complexity and Practical Computation of Approximate Stationarity Concepts of Lipschitz Functions. In *International Conference on Machine Learning (ICML)*, pages 21360–21379, 2022.

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

[148] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. `https://www.mit.edu/~dimitrib/PTseng/papers/apgm.pdf`, 2008. Accessed May 1, 2020.

[149] A. C. Wilson, B. Recht, and M. I. Jordan. A lyapunov analysis of momentum methods in optimization. *arXiv preprint arXiv:1611.02635*, 2016.

[150] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 4148–4158, 2017.

[151] B. E. Woodworth and N. Srebro. Tight Complexity Bounds for Optimizing Composite Objectives. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 3639–3647, 2016.

[152] L. Xiao and T. Zhang. A Proximal Stochastic Gradient Method with Progressive Variance Reduction. *SIAM J. Optim.*, 24(4):2057–2075, 2014.

[153] L. Xiao, A. W. Yu, Q. Lin, and W. Chen. DSCOVR: Randomized Primal-Dual Block Coordinate Algorithms for Asynchronous Distributed Optimization. *J. Mach. Learn. Res.*, 20(1):1634–1691, 2019.

[154] H.-F. Yu, H.-Y. Lo, H.-P. Hsieh, J.-K. Lou, T. G. McKenzie, J.-W. Chou, P.-H. Chung, C.-H. Ho, C.-F. Chang, Y.-H. Wei, et al. Feature engineering and classifier ensemble for KDD cup 2010. In *KDD cup*, 2010.

[155] J. Zhang, H. Lin, S. Jegelka, S. Sra, and A. Jadbabaie. Complexity of Finding Stationary Points of Nonconvex Nonsmooth Functions. In *International Conference on Machine Learning (ICML)*, pages 11173–11182, 2020.

[156] Y. Zhang and L. Xiao. Stochastic Primal-Dual Coordinate Method for Regularized Empirical Risk Minimization. In *International Conference on Machine Learning (ICML)*, pages 353–361, 2015.

[157] D. Zhou, P. Xu, and Q. Gu. Stochastic Nested Variance Reduction for Nonconvex Optimization. *J. Mach. Learn. Res.*, 21:103:1–103:63, 2020.

[158] K. Zhou, F. Shang, and J. Cheng. A Simple Stochastic Variance Reduced Algorithm with Fast Convergence Rates. In *International Conference on Machine Learning (ICML)*, pages 5980–5989, 2018.

[159] K. Zhou, Q. Ding, F. Shang, J. Cheng, D. Li, and Z.-Q. Luo. Direct Acceleration of SAGA using Sampled Negative Momentum. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1602–1610, 2019.

[160] K. Zhou, Y. Jin, Q. Ding, and J. Cheng. Amortized Nesterov's Momentum: A Robust Momentum and Its Application to Deep Learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211–220, 2020.

[161] K. Zhou, A. M.-C. So, and J. Cheng. Boosting First-Order Methods by Shifting Objective: New Schemes with Faster Worst-Case Rates. In *Advances in Neural Information Processing Systems (NIPS/NeurIPS)*, pages 15405–15416, 2020.

[162] K. Zhou, A. M.-C. So, and J. Cheng. Accelerating Perturbed Stochastic Iterates in Asynchronous Lock-Free Optimization. In *NeurIPS Workshop on Optimization for Machine Learning (NeurIPS OPT)*, 2022.

[163] K. Zhou, L. Tian, A. M.-C. So, and J. Cheng. Practical Schemes for Finding Near-Stationary Points of Convex Finite-Sums. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3684–3708, 2022.