THE UTILIZATION OF SECOND-ORDER INFORMATION FOR LARGE-SCALE UNCONSTRAINED OPTIMIZATION PROBLEMS

HARDIK TANKARIA

THE UTILIZATION OF SECOND-ORDER INFORMATION FOR LARGE-SCALE UNCONSTRAINED OPTIMIZATION PROBLEMS

by

HARDIK TANKARIA

Submitted in partial fulfillment of
the requirement for the degree of
DOCTOR OF INFORMATICS
(Applied Mathematics and Physics)

KYOTO UNIVERSITY
KYOTO 606–8501, JAPAN
FEBRUARY 2024

Preface

Second-order optimization methods, which utilize information from the Hessian matrix, employ various real-world applications, including in statistics, machine learning, computer vision, control systems, and scientific research. In this thesis, we introduce the utilization of three specific types of second-order information: 1) a regularized limited memory BFGS method with simultaneous use of line search, 2) reduction of variance of stochastic gradient via the Barzilai-Borwein method and Barzilai-Borwein step size, and 3) a regularized Nyström method.

Since the 1980s, second-order methods have gained popularity due to their ability to achieve faster convergence compared to first-order methods. Nonetheless, the large data sets involved in machine learning pose a challenge for large-scale optimization problems, particularly in the computation and inversion of Hessian approximations. Despite extensive research on Hessian approximation, there are still several significant issues that continue to demand consideration and attention. These include the high memory storage, high computational cost, and CPU time associated with many Hessian approximation techniques. This imposes the development of novel or modified Hessian approximation techniques with reduced memory and computational costs. Moreover, the selection of an optimal step size for iterative algorithms remains an open problem in mathematical optimization. Recently, many research methodologies concerning these issues have been studied well. These methodologies tackle them using Hessian approximations with various regularization techniques, such as using gradient information or trust region methods. Unlike conventional regularized limited memory BFGS approaches, our method incorporates Wolfe line search and obtains a regularized parameter. An inherent issue with the conventional method is its demand for a high number of function evaluations. These studies motivated us to propose the regularized limited memory BFGS method with the conditional use of Wolfe line search. As a result, our strategy integrates the conditional employment of a strong Wolfe line search, enabling the iteration to adopt longer step sizes under specific conditions.

Second-order techniques have huge scope in stochastic gradient methods. Given the widespread use of stochastic gradient methods in machine learning and deep learning, extensive research has been conducted on various stochastic gradient methods, revealing issues related to variance induced by noise and stochasticity. Consequently, there is a substantial need to introduce methodologies aimed at reducing variance within stochastic regimes. In response to this challenge, we propose the incorporation of second-order information, specifically the Barzilai-Borwein approximation, into the search direction of the variance-reduced stochastic gradient method for strongly convex objective functions. Then we establish the generalized convergence using a unique bound over the Lipschitz continuity of the second-order information under usual assumptions. Furthermore, we demonstrate numerical experiments, showcasing their outstanding performance compared to existing similar methods, especially for high-dimensional, dense datasets.

Moreover, a majority of Hessian approximations rely on first-order information, a strategy that might be less effective for ill-conditioned problems. Usually, modifications like regularization techniques are often employed with classical Newton's method to tackle ill-conditioned problems. However, computing the actual Hessian is not practical for large-scale problems. One of the ideas is to obtain a low-rank Hessian approximation tailored for datasets with huge feature spaces. In response to this challenge, we propose to use the regularized Nyström method. Nyström method that extracts the second-order information using a random column selection procedure, resulting in a low-rank Hessian approximation. Subsequently, we integrate this approximation with a judiciously chosen regularized parameter to enhance its efficacy. We subsequently establish probabilistic linear convergence and an upper bound on the distance between the proposed approximation and the Hessian for both convex and non-convex functions. Finally, we conduct numerical comparisons with state-of-the-art existing methods, and the proposed approach yields outstanding results.

The author hopes that the outcomes presented in the current thesis will serve as a valuable contribution to ongoing research on second-order approximation techniques for large-scale unconstrained optimization problems.

Hardik Tankaria
February 2024, Kyoto, Japan

Acknowledgments

As I conclude my doctoral course journey, I would like to convey my deepest gratitude to the numerous individuals who have played pivotal roles in supporting me to achieve my research goals.

Going to a foreign land for research, where everything, from language to culture, people to food, is new and unknown, adjusting to life in a foreign country was no small feat. First and foremost, I want to extend my profound gratitude, appreciation, and affection to my parents, Mrs. Rita Tankaria and Mr. Deepak Tankaria, whose love, encouragement, support, and sacrifice have been the pillars of my research journey in Japan.

I would like to convey my heartfelt appreciation and profound thanks to my supervisor, Professor Nobuo Yamashita, for offering me the chance to join his outstanding optimization laboratory and for guidance over the last 6 years, without which this thesis would not be possible. Prof. Yamashita is the kindest professor I have ever met, and I have not encountered such kindness from any other professor in my academic journey. Coming from different majors in masters and changing majors to optimization in a doctoral course with limited basic knowledge, it was not so easy to manage in the initial stage. However, his consistent support, expert guidance, and insightful feedback throughout every stage of this research journey shaped and profoundly influenced my research journey. Each insightful discussion with him has helped me to understand concepts and broaden my horizons to think critically. I really appreciate the trust and freedom he gave me to do my research activities and collaborations outside the laboratory. Furthermore, I would like to extend my sincere thanks to Professor Yamashita for generously providing opportunities to participate in academic conferences, become a research assistant, and attend a summer school. This experience has significantly broadened my research perspective and helped me work on various unconventional research topics, like random sampling methods and federated learning.

I would like to acknowledge the support of my advisors, Prof. Yoshimasa Nakamura and Prof. Kenji Kashima, for their understanding and support. Prof. Kenji Kashima is very kind, and he always encouraged me and discussed various topics in a friendly manner. Also, he noted my track record in the last three years and offered help in case of any difficulties.

I am especially grateful to Prof. Makoto Yamada for extending the opportunity to work as a research assistant in his Machine Learning and Data Mining lab, affiliated with the Department of Intelligence Science and Technology at the Graduate School of Informatics. During challenging periods, his steady support was vital. Also, his invaluable encouragement and perceptive feedback helped me explore the possibilities for real-world applications of optimization. His consistent presence in the office, engaging in regular enthusiastic discussions, tracking daily progress, and coffee/lunch meetings served as a motivating force that charged my eagerness for the research work. Furthermore, I express my sincere appreciation to Prof. Makoto Yamada for providing opportunities to actively participate in seminars and workshops.

I am deeply grateful to Associate Prof. Ellen Hidemi Fukuda for her multifaceted support throughout my academic endeavors. From offering guidance on reference materials, articles, and research topics to providing invaluable advice on career paths, her unwavering support, encouragement, and dedication have created an intellectually stimulating environment that significantly enriched my academic journey.

I want to express my deepest gratitude to my senior, assistant Prof. Dinesh Singh, who actively engaged in discussions on a wide and various range of research topics and whose support in learning new areas has significantly enriched my last 3 years of research journey. I feel truly fortunate to have had such a supportive and dedicated senior, and I extend my heartfelt and sincere thanks for his invaluable suggestions to help me achieve my research endeavors.

I would like to extend my utmost appreciation to associate Prof. Hiroyuki Sato for helping me explore the different themes of Riemannian manifolds. He helped me introduce Riemannian manifolds over online meetings, explained these topics with examples, and suggested many references. Also, I would like to extend my thanks to Assistant Professor Yuya Yamakawa for some discussions during seminars.

Special thanks are due to my colleagues assistant Prof. Gu Yan, Dr. Rafael Pedro Longhi, Dr. Daiana doc Santos Viana, Assistant Prof. Atsushi Hori, my sister Ms. Kruti Tankaria and a friend Dr. Surya Pratap Singh Chauhan whose collaborative attitude and diverse perspectives have resulted in robustness to my research. Their solidarity, mutual support, and bond made my research enlightening, memorable, and enjoyable.

Moreover, the staff, Mrs. Yuiko Igura, Mrs. Aya Kiriyama, and Ms. Rei Takimoto, and administrators at the student affairs division at the graduate school of informatics, whose dedication to facilitating academic pursuits has created a conducive atmosphere for research and learning.

I want to mention my sincere appreciation to Aasociate Prof. Yogita Madhkant Parmar and associate Prof. Narasimha Kumar for their invaluable guidance and mentorship during my undergraduate and master's studies, which laid a strong foundation for my research journey.

Lastly, I acknowledge the financial support provided by the Japan International Cooperation Agency (JICA) for the first 3 years (from April, 2018 to March, 2021) without which this research would not have been possible.

I express sincere gratitude to everyone who has contributed to my doctoral research journey at Kyoto University.

Contents

List of Figures

List of Tables

List of Notations

$\mathbb{R}$	the set of real numbers
$\mathbb{R}^n$	the set of n-dimensional real space
$\|x\|$	the absolute value of number x
x_i	the i-th coordinate of vector x
$\\|x\\|_1$	the ℓ_1-norm of vector x
$\\|x\\|_2$	the ℓ_2-norm of vector x
$\langle\cdot,\cdot\rangle$	the Euclidean inner product
$\mathrm{dom} f$	the effective domain of function f
$\nabla f(x)$	the gradient of f at x
$\nabla^2 f(x)$	the Hessian matrix of f at x
I	the identity matrix
I_n	the $n \times n$ identity matrix
$A^\top$	the transpose of matrix A
A_j	the j-th column matrix A
A_{ij}	the element at the i-th row and j-th column of matrix A
$A \succ 0$	matrix A is positive definite
$A \succeq 0$	matrix A is positive semidefinite
$\lambda_i(A)$	the eigenvalue of a real symmetric matrix A
$\lambda_{\min}(A)$	the minimum eigenvalue of a real symmetric matrix A
$\lambda_{\max}(A)$	the maximum eigenvalue of a real symmetric matrix A
$\mathrm{Tr}(A)$	the trace of matrix A, *i.e.*, if A is $d \times d$, then $\mathrm{Tr}(A) = \sum_{i=1}^d A_{ii}$
$A^\dagger$	the Moore-Penrose pseudoinverse of matrix A
$\\|A\\|_F$	the Frobenius norm of matrix A, $\\|A\\|_F = \sqrt{\mathrm{Tr}(A^\top A)}$
$\\|A\\|_2$	the spectral norm of matrix A, $\\|A\\| = \max_{\\|x\\|=1} \\|Ax\\|_2 = \sqrt{\lambda_{\min}(A^\top A)}$

Chapter 1

Introduction

Optimization is a process to optimize something, or make the best. In applied mathematics, the *Mathematical optimization* is an important branch that optimizes an objective function under the given constraints. That is, it minimizes or maximizes a given objective function subject to constraints on its variables. One of the subfields of mathematical optimization is the *Convex optimization* which optimizes convex functions and is useful in many different fields such as machine learning, signal processing, control systems, engineering, statistics, finance, etc.

In the contemporary era marked by an exponential growth in data volume, there is a compelling need to enhance the solving capacity of optimization algorithms. This is particularly relevant, as various applications give rise to optimization problems characterized by a large number of variables. A promising approach to addressing this challenge involves developing efficient optimization algorithms tailored for large-scale problems.

This thesis is dedicated to exploring large-scale optimization problems. This chapter will provide an introduction to different types of large-scale optimization problems, specific solution methods, and outline the details of the thesis.

1.1 Large-scale optimization

Generally a *mathematical optimization* problem can be written as follows:

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) = 0, \quad i \in \mathcal{E} \\ & g_i(x) \le b_i, \quad i \in \mathcal{I} \end{aligned} \tag{1.1.1}$$

where the vector $x = (x_1, x_2, \ldots, x_n)$ is $n-$dimensional and it is called the *decision variables* or *parameters* of the problem, f is called the *objective function* of x that we want to optimize (minimize or maximize), and g_i's are the *constraints functions* with the set $\mathcal{E}$ of indices for equality constraints and the set $\mathcal{I}$ of indices for the inequality constraints, and constants b_i are the bounds of the constraints.

There are essentially two categories of optimization problems that primarily differ based on the nature of the *decision variables*. Certain optimization problems necessitate their *decision variables* to be exclusively integers or generally discrete variables, leading to what is termed *discrete optimization*, where *decision variables* strive to make the optimal selection from a finite set. On the other hand, *continuous optimization* permits its *decision variables* to take real number values. In this thesis, our exclusive focus is on *continuous optimization.*

1.1.1 Constrained and unconstrained optimization

The general form of the *objective function* (1.1.1) can be classified into various optimization problems according to the nature of constraints, such as linear, nonlinear, and convex, and the smoothness of the functions.

Constrained optimization problems usually arise in applications where there are some restrictions on the *decision variables* that play a crucial role in serving the goal of an objective function.

On the other hand, there are many applications in which there are no such restrictions on the *decision variables*, which is called the *unconstrained optimization* problem. *Unconstrained optimization* problems usually have $\mathcal{E} = \mathcal{I} = \emptyset$ in the *objective function* (1.1.1), which can be written as follows:

$$\min_{x \in \mathbb{R}^n} f(x). \tag{1.1.2}$$

These issues can be cast as a reformulation of constrained optimization problems, wherein the constraints are commonly substituted by the inclusion of penalizing functions to account for constraint violations. These functions are commonly referred to as *penalty* functions. Furthermore, numerous *constrained optimization* problems exist where constraints can be disregarded, as their presence neither impedes the algorithm nor exerts any influence on the solution.

In this thesis, we consider the *unconstrained optimization* problems, especially where the *objective function* is nonlinear.

Next, we give some applications of the *objective function* that are useful in machine learning, statistics, control systems, science, and engineering.

1.1.2 Applications

The optimization problem (1.1.1) has been extensively used in many applications such as machine learning [70, 56], compressed sensing [19, 3, 20, 12], science [57, 90], statistics [57], etc. First, we see some basic optimization problems, such as least-squares and quadratic programming problems.

(1) **Least-squares:** One of the applications of problem (1.1.1) is the unconstrained least-squares problem. The problem of least squares can be defined as the sum of squared terms:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\|F(x)\|^2, \tag{1.1.3}$$

where each $F : \mathbb{R}^n \to \mathbb{R}^m$, Euclidean norm of F at x is $\|F(x)\|$ and $m \geq n$. The problem (1.1.3) can be seen in [10, 9, 82, 89, 7], where it is represented as the curve fitting problem. Considering the curve fitting problem, let $a_i \in \mathbb{R}^d$ be input data for $i = 1, \dots, m$, and $b_i \in \mathbb{R}$ be the observation of data a_i. If the fitting function is given by $h(x, a_i)$ parametrized by x. Moreover, when $n \leq m$, there may be no solution.

If we define,

$$F(x) := \begin{pmatrix} h(x,a_1) - b_1 \\ \vdots \\ h(x,a_m) - b_m \end{pmatrix},$$

then problem (1.1.3) can be given as,

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} [h(x,a_i) - b_i]^2. \tag{1.1.4}$$

Unconstrained least-squares convex problems are also used in the modern robotics [32] which can be written as:

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|,$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$.

(2) **Quadratic programming(QP):** The QP usually occurs in engineering and in computer vision [72, 123]. The QP problem can be given as follows:

$$\text{minimize } \frac{1}{2}x^\top Ax + b^\top x,$$

subject to $Qx = y, x \geq 0$, $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$ is a positive definite matrix. In the case where $A = 0$, resulting problem becomes the standard linear programming problem. Also, with the conic constraint $x \in \mathcal{K}$, it reduces to a quadratic conic programming problem.

The primary objective of data mining and machine learning is to forecast the likelihood of a particular outcome. Within this domain, two pivotal prediction problems emerge:

regression and classification. Large-scale optimization problems manifest in scenarios like curve fitting, notably in classification and regression tasks. Let us contemplate a large-scale optimization problem formulated as a finite sum, defined as follows:

$$\min_{x\in\mathbb{R}^d} f(x) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1.1.5}$$

where $x \in \mathbb{R}^d$ is a d-dimensional decision variable, n number of functions, and f_i is the component function. Also, f_i represents the data point (a_i, b_i) that is,

$$f_i = \ell(h(x, a_i), b_i), \tag{1.1.6}$$

where ℓ is the loss function and is parametrized by the activation function $h(x, a_i)$. The prediction can be obtained as $\ell(h(x, a_i), b_i) = \ell(\widehat{b_i}, b_i)$, where $\widehat{b_i} = h(x, a_i)$. One such standard example is a linear function $\widehat{b_i} = x^\top a_i + c$ with $c > 0$.

Classification: The classification [5] is a procedure for obtaining a model that helps to divide data into multiple classes. Let $a_i \in \mathbb{R}^d$ be a d-dimensional feature vector, and $b_i \in \{-1, 1\}$(binary) is the target (output) vector of the ith data sample. The classification problem $f(x)$ can be given by,

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1.1.7}$$

where $f_i(x) = \ell(h(x, a_i), b_i)$, and ℓ is the loss function. The most-used loss function for ith data can be defined as follows:

$$\begin{aligned}
\textit{logistic loss:}\quad & \ell(\widehat{b_i}, b_i) = \log(1 + \exp(-b_i\widehat{b_i})),\\
\textit{hinge loss:}\quad & \ell(\widehat{b_i}, b_i) = (1 - b_i\widehat{b_i})_+,\\
\textit{exponential loss:}\quad & \ell(\widehat{b_i}, b_i) = \exp(-b_i\widehat{b_i}).
\end{aligned}$$

Regression: Regression is a procedure for obtaining a model to determine the data into continuous values instead of classes. That is, it maps the linear relationship between dependent and independent variables. The classical regression loss functions for ith data can be defined as follows:

$$\begin{aligned}
\textit{squared loss:}\quad & \ell(\widehat{b_i}, b_i) = (b_i - \widehat{b_i})^2,\\
\textit{huber loss:}\quad & \ell(\widehat{b_i}, b_i) = \begin{cases} \frac{1}{2}(b_i - \widehat{b_i})^2 & \text{for } |b_i - \widehat{b_i}| \le \delta,\\ \delta(|b_i - \widehat{b_i}| - \frac{1}{2}\delta), & \text{otherwise.}\end{cases}
\end{aligned}$$

Next, we discuss the generalized *unconstrained regularized convex optimization problems*, which have the form of the main objective function (1.1.1) in addition to a regularized function as follows:

$$\min_{x\in\mathbb{R}^n} \ f(x)+g(x), \tag{1.1.8}$$

where f is a usual loss function and g is a regularized function. In the case of ill-conditioned problems, regularization technique [92, 21, 116] is used to improve the conditioning of the problem. Tikhonov regularization, also known as squared ℓ_2-norm regularization, and ℓ_1-norm regularization stand out as widely adopted regularizers. At this moment, we recite the different regularizers and the loss function along with those regularizers.

(1) ℓ_1**-regularizer:** It is often used to shrink the solution so that the coefficient of less important features to zero. *i.e.*, to obtain a sparse solution [21, 114, 115, 124, 12, 126]. ℓ_1-regularization can be written as,

$$g(x)=\lambda\|x\|_1, \tag{1.1.9}$$

where $x\in\mathbb{R}^n$ and $\lambda>0$ is the weight of the regularizer.

(1a) **LASSO:** LASSO stands for least absolute shrinkage selector operator [115, 47] where loss function is the mean squared error along with the ℓ_1-regularization:

$$\begin{aligned}&\underset{x\in\mathbb{R}^d}{\text{minimize}} \quad \frac{1}{n}\|Ax-b\|^2\\ &\text{subject to } \|x\|_1\le c,\end{aligned}$$

where $c>0$. From now onwards, for the simplicity of the notation, we denote ℓ_2-norm $\|x\|_2$ as $\|x\|$. Also, the above constrained problem can be written as an unconstrained optimization problem:

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\|Ax-b\|^2+\lambda\|x\|_1,$$

where $\lambda>0$ and scalar λ usually decided via cross-validation. Considering regularized objective function (1.1.8), the loss function $f(x)=\frac{1}{n}\|Ax-b\|^2$, and $g(x)=\lambda\|x\|_1$.

(2) ℓ_2**-regularizer:** Tikhonov regularization is often used to avoid overfitting [116, 117, 85, 61, 23]. However, larger regularizers may add weight and lead to underfitting. ℓ_2-regularizer can be written as,

$$g(x)=\lambda\|x\|_2^2, \tag{1.1.10}$$

where $x\in\mathbb{R}^n$ and $\lambda>0$ is the weight of the regularizer.

(2a) **Ridge regression**: Ridge regression employs the mean squared error as a loss function and puts in ℓ_2 regularization. The cost function can be given as follows:

$$\begin{aligned} &\underset{x\in\mathbb{R}^d}{\text{minimize}} \quad \frac{1}{n}\|Ax-b\|^2 \\ &\text{subject to } \|x\|_2^2 \leq c, \end{aligned}$$

where $c > 0$. Also, the above constrained problem can be written as an unconstrained optimization problem:

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\|Ax-b\|^2 + \lambda\|x\|_2^2, \tag{1.1.11}$$

where $\lambda > 0$ and scalar λ usually decided via cross-validation. Comparing (1.1.8), which means $f(x) = \frac{1}{n}\|Ax-b\|^2$, and $g(x) = \lambda\|x\|^2$.

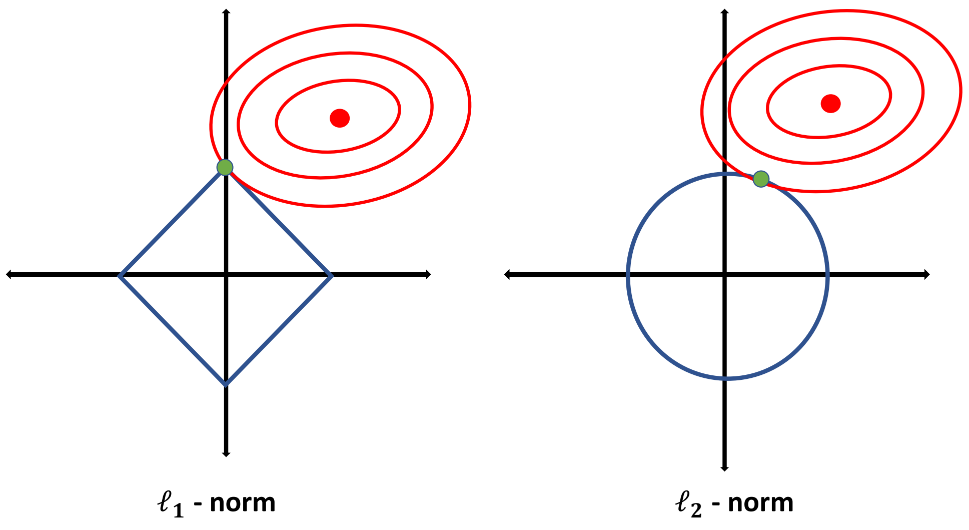


Figure 1.1: Comparison between ℓ_1-norm and ℓ_2-norm regularizer

Figure (1.1) shows the ℓ_1-regularization with sparse solution and ℓ_2-regularization with relatively dense solution.

(3) **The mixed norm penalty** [74, 63] improves the limitation of the ℓ_1-regularization and works well to obtain the best of both ℓ_1 and ℓ_2-regularization, *i.e.*, it obtains the solution that is sparse at the group of parameters.

$$g(x) = \lambda_1\|x\|_1 + \lambda_2\|x\|^2,$$

where $\lambda_1, \lambda_2 > 0$ are the weights of the regularizers.

(3a) **Elastic net:** Elastic net regression [128] combines both penalties ℓ_1 and ℓ_2 regularization linearly. It can be given as:

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\|Ax-b\|^2 + \lambda_1\|x\|_1 + \lambda_2\|x\|^2, \tag{1.1.12}$$

where $\lambda_1, \lambda_2 > 0$. It is clear that $f(x) = \frac{1}{n}\|Ax-b\|^2$ and $g(x) = \lambda_1\|x\|_1 + \lambda_2\|x\|^2$.

1.2 Existing solution methods and their pros and cons

In this section, we discuss the methods for solving nonlinear, large-scale, unconstrained optimization problems. Predominantly, most of the classical solution methods rely on Taylor's series expansion. Such classical methods can be categorized into two categories: first-order methods and second-order methods based on the first-order and second-order Taylor expansions. In optimization, the procedure of minimizing objective function is an iterative scheme that generates a point x_{k+1} incorporating a certain size of step at a particular direction in the previous point such that $f(x_{k+1})$ attains minimal value.

1.2.1 First order methods

We want to minimize the unconstrained objective (loss) function (1.1.1), $f(x)$, which is continuously differentiable. In this subsection, we discuss two classical gradient-based first-order solution methods: the *steepest descent* gradient method and the *conjugate gradient* method.

(1) **Steepest descent method:** It is one of the oldest classical optimization methods. Given a continuously differentiable objective function $f : \mathbb{R}^n \to \mathbb{R}$, *steepest descent* gradient method [22, 8] update a step in the *steepest* direction is $-\nabla f(x)$. The method of *steepest descent* computes a sequence of iterate x_k as,

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \quad k = 0, 1, 2, \dots \tag{1.2.1}$$

where $\alpha_k > 0$ is called a stepsize. The steepest descent method always progresses downhill, avoids saddle points, and works efficiently when the iterate x_k is away from the minimum. However, it becomes slower when iterate x_k is near the minimum. Moreover, the trajectory of linear search may not work well and might take "zigzag" path. The steepest descent method converges with a linear rate [82, 79] which is very slow for nonlinear problems. Poylak [94] proposed an idea to incorporate the momentum term, which involves steps containing past gradients. This acceleration technique is called the *classical momentum.* This technique helps to achieve stability and may accelerate convergence. Nesterov [84] proposed a modification in classical momentum using the *look ahead* feature by projecting a gradient ahead of the current step. This method is also known as Nesterov accelerated gradient (NAG).

(2) **Conjugate direction method:** The main motivation to propose this method was to accelerate the convergence rate of the steepest descent method. The conjugate direction method [59] is an iterative process that was originally developed to solve the quadratic problem as part of optimization problems:

$$\min_{x\in\mathbb{R}^n} \phi(x) \overset{\text{def}}{=} \frac{1}{2}x^\top Ax - b^\top x$$

assuming the matrix A is positive-definite. Then the gradient of ϕ equals

$$\nabla\phi(x) = Ax - b \overset{\text{def}}{=} r(x),$$

with $x = x_k$ implies at the kth iteration. This method involves an iterative process that converges to the solution while minimizing the computation cost using the conjugacy property in its search direction. This property ensures that each iteration is orthogonal to the previous ones. A collection of nonzero vectors $\{p_0, p_1, \ldots, p_\ell\}$ is said to be *conjugate* with respect to the symmetric positive definite matrix A if

$$p_i^\top Ap_j = 0, \text{ for all } i \neq j.$$

One can see that this collection of sets is linearly independent. The significance of this method is that it can minimize $\phi(x)$ in n steps by sequentially minimizing it along with distinct directions within a conjugate set. Let x_0 be an initial point and $\{p_0, p_1, \ldots, p_\ell\}$ be a set of conjugate direction vectors, then the iteration of the sequence $\{x_k\}$ generated by the conjugate direction method can be given as

$$x_{k+1} = x_k + \alpha_k p_k, \tag{1.2.2}$$

where α_k is the minimizer of the quadratic function $\phi(x)$ given by,

$$\alpha_k = -\frac{r_k^\top p_k}{p_k^\top Ap_k}.$$

(2.1) **Conjugate gradient method:** It is a conjugate direction method [45, 58] that generates a vector p_k using the previous vector p_{k-1} by applying the Gram-Schmidt procedure to the gradient vectors. This CG method can be applied to nonquadratic problems:

$$\min_{x\in\mathbb{R}^n} f(x),$$

and its iteration can be given as (1.2.2) where α_k is a step size that can be obtained by line search minimization

$$f(x_k+\alpha_k p_k)=\min_{\alpha} f(x_k+\alpha p_k),$$

and the search direction is written by

$$p_k=-r_k+\beta_k p_{k-1},\quad k=1,\ldots,d-1,\ \text{ and }\ p_0=-r_0,$$

where β_k is given by

$$\beta_k=\frac{r_k^\top r_k}{g_{k-1}^\top g_{k-1}}.$$

Since it is memory-efficient, it is suitable for problems of large-scale optimization. The CG method often exhibits fast convergence, particularly for well-conditioned problems. However, convergence can be influenced by the choice of conjugacy parameter and line search strategy, and it has limited applicability to non-linear problems [41, 79, 8].

1.2.2 Second order methods

In this subsection, we discuss the two most popular second-order methods: Newton's method and the quasi-Newton method. These two methods are powerful optimization techniques that play an important role in solving nonlinear optimization problems.

(1) **Newton's method:** The Newton's method [37, 65] is a classical optimization method that is known for its rapid convergence. The key idea is to approximate the objective function with a quadratic form near the current iteration. Consider the problem of solving an unconstrained optimization problem at $x=\bar{x}$ given as a twice continuous differentiable objective function (1.1.2), and the quadratic Taylor expansion of $f(x)$ can be given by:

$$f(x)\approx p(x):=f(\bar{x})+\nabla f(\bar{x})^\top(x-\bar{x})+\frac{1}{2}(x-\bar{x})^\top\nabla^2 f(\bar{x})(x-\bar{x}),$$

where $\nabla f(x)$ is the gradient and $\nabla^2 f(\bar{x})$ is the Hessian matrix (second-order partial derivatives) of $f(x)$. Minimizing a quadratic function $p(x)$ by solving $\nabla p(x)=0$, we get

$$\nabla p(x)=\nabla f(\bar{x})+\nabla^2 f(\bar{x})(x-\bar{x}),$$

which solves this system of linear equations, implying

$$x = \bar{x} - \left(\nabla^2 f(\bar{x})\right)^{-1} \nabla f(\bar{x}) \tag{1.2.3}$$

which is called the Newton step at $x = \bar{x}$, step size $\alpha_k = 1$ for the iterate $x = \bar{x} - \alpha_k p_k^N$, and $-\left(\nabla^2 f(\bar{x})\right)^{-1} \nabla f(\bar{x})$ called the Newton direction p_k^N. Since it uses the Hessian matrix or second-order information, Newton's method gives an accurate and direct approach to obtaining the minimum, particularly when the initial guess is close to the optimal solution. However, the computational cost of computing the Hessian matrix and inverting the Hessian matrix can be expensive, especially for high-dimensional problems. The computational complexity of Newton's method is $O(n^2+n^3)$ per iteration, and thus it is not acceptable for large-scale realms. Additionally, issues may arise, such as whether the Hessian matrix may be singular or whether the Newton direction may not the descent direction, depending on the properties of the objective function. Consequently, the convergence analysis of Newton's method folds into two important aspects: local and global convergence. Local convergence deals with the pure form (1.2.3) of the method near a nonsingular local minimum. Thus, when the Hessian matrix is positive definite, Newton's method converges quadratically [30, 29], provided that the step size $\alpha_k = 1$ for all k. Whereas global convergence needs some modifications to ensure the nonsingularity of the Hessian matrix, as we will see in the further sections.

(2) **Quasi-Newton method:** Quasi-Newton methods [33, 34, 44, 69] are the milestone in solving the nonlinear optimization problems. Originating with W.C. Davidon's development in the mid-1950s, as documented in the technical report [27]. However, its formal recognition took more than 30 years until it got published in the SIAM journal of Optimization [28]. The primary objective of the quasi-Newton methods is to get rid of computationally intensive and expensive steps in the computation of Hessian and its inverse in Newton's method. Instead, quasi-Newton methods approximate the Hessian matrix by introducing the matrix B using the difference in both gradient and parameters. This strategic methodology enhances the optimization process and marks the significant evolution of nonlinear optimization methodologies. Subsequently, we examine the different variants of the quasi-Newton methods. There are 3 main variants of quasi-Newton methods, 1) DFP, 2) BFGS, and 3) SR1. Consider the objective function (1.1.2) in addressing the unconstrained optimization problem. Similar to Newton's method, quasi-Newton methods iterate as follows:

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k),$$

where α_k denotes step size computed using Armijo's or Strong Wolfe line search, and $B_k = H_k^{-1}$ represents an approximation of the Hessian at x_k. Typically, B_k is computed using B_{k-1}, and the implementation of B_k from B_{k-1} varies across the different quasi-Newton methods. However, the pivot requirement for B_k is to satisfy

$$B_k s_{k-1} = y_{k-1},$$

called the *secant* condition, where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$. The secant condition requires B_k to be symmetric positive definite, which is achievable only if s_k and y_k adhere to the *curvature* condition,

$$s_k^\top y_k > 0.$$

This condition is easily satisfied when the objective function $f(x)$ is strongly convex. However, it requires some modifications for nonconvex functions. Also, to determine the B_{k+1} uniquely that satisfies the secant condition, being closest to the B_k, it requires solving the problem.

$$\begin{aligned} \min_{B} \quad & \|B - B_k\| \\ \text{subject to } & B = B^\top, \quad Bs_k = y_k. \end{aligned} \tag{1.2.4}$$

There are three different solutions for this problem, and it depends on the various techniques as follows:

1) **DFP method:** The Davidon-Fletcher-Powell update is a unique solution of (1.2.4) that is given by

$$B_{k+1} = (I - \rho_k y_k s_k^\top) B_k (I - \rho_k s_k y_k^\top) + \rho_k y_k y_k^\top,$$

with

$$\rho_k = \frac{1}{y_k^\top s_k}. \tag{1.2.5}$$

This formula, known as the DFP updating formula, was originally proposed by Davidon in 1959 and subsequently studied, implemented, and popularized by Fletcher and Powell. The inverse of $B_k^{-1} = H_k$, is calculated using the Sherman-Morrison-Woodbury formula,

$$(A + UV^\top)^{-1} = A^{-1} - A^{-1}U(I + V^\top A^{-1} U)^{-1} V^\top A^{-1}$$

(for $A \in \mathbb{R}^{n\times n}$, and $U, V \in \mathbb{R}^{n\times d}$) to obtain the inverse Hessian approximation H_k

the resulting DFP update is

$$H_{k+1} = H_k - \frac{H_k y_k y_k^\top H_k}{y_k^\top H_k y_k} + \frac{s_k s_k^\top}{y_k^\top s_k}.$$

Note that the final two terms on the right-hand side of the above equation consist of rank-one matrices, which leads to a rank-two modification of H_k.

2) **Broyden-Fletcher-Goldfarb-Shanno (BFGS) method**: The BFGS method [11, 43, 50, 104] discovered by four mathematicians Broyden, Fletcher, Goldfarb, and Shanno. Similar to the DFP formula, the BFGS method imposes similar conditions on H_k, and now the secant equation can be expressed as

$$H_{k+1} y_k = s_k,$$

and the BFGS update formula [34] for H_{k+1} is:

$$H_{k+1} = (I - \rho_k s_k y_k^\top) H_k (I - \rho_k y_k s_k^\top) + \rho_k s_k s_K^\top,$$

with ρ_k defined in (1.2.5). However, this implementation is not computationally efficient as it requires $O(n^2)$. We will see the less expensive variant of the BFGS method.

3) **SR1 method:** As one can notice, both the DFP and BFGS methods update the B_{k+1} matrix with a rank-2 matrix. The *symmetric-rank-1* or SR1 method updates the simple rank-1 matrix, ensuring symmetry and satisfying the secant equation. The usual form of SR1 update is

$$B_{k+1} = B_k + \sigma v v^\top,$$

where σ is ± 1 and v is selected to ensure that B_{k+1} satisfies the secant equation. The unique rank-1 updating formula is given by:

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^\top}{(s_k - H_k y_k)^\top y_k}.$$

4) **The Broyden class:** This class is a sequence of updates, as follows:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k} + \phi_k (s_k^\top B_k s_k) v_k v_k^\top,$$

where ϕ_k is a scalar parameter and

$$v_k = \left[\frac{y_k}{y_k^\top s_k} - \frac{B_k s_k}{s_k^\top B_k s_k} \right].$$

The quasi-Newton methods (specially DFP and BFGS) use Dennis-Moré condition [33]

$$\lim_{k\to\infty} \frac{\|\nabla f(x_k) + \nabla^2 f(x_k) p_k\|}{\|p_k\|} = 0,$$

for super linear convergence [25, 18, 96] of general twice continuously differentiable non linear function, where p_k is descent direction. The convergence of BFGS and DFP is somewhat slower than that of the quadratic convergence of Newton's method. Conn et al [26] studied the convergence for the SR1 in detail.

1.2.3 Limited memory quasi-Newton methods

The limited-memory quasi-Newton methods are one of the important tools to solve large problems whose Hessian can be expensive to compute or dense. As the name implies, "limited" memory, this method does not store the dense $n \times n$ matrices at each iteration and approximates the Hessian using the limited number of past few updates.

1) **Limited memory BFGS (L-BFGS) method**: L-BFGS method [87, 78] offers a vital solution and addresses storage and computational challenges. It computes the approximated Hessian using modified H_k of BFGS using the certain number of vector pairs (s_k, y_k) used in BFGS,

$$H_{k+1} = (I - \rho_k s_k y_k^\top) H_k (I - \rho_k y_k s_k^\top) + \rho_k s_k s_K^\top.$$

L-BFGS algorithm obtains the matrix-vector product $H_k \nabla f(x_k)$ using the inner products involving the last few vector pairs, say m a.k.a. memory vector pairs, and at each iteration the last pair is replaced with the new pair, where $m \ll n$. To be specific, at iteration x_k, the set of vector pairs (s_k, y_k) for $i = k - m, \dots, k-1$ and let $H_k^0 = \gamma_k I$ where

$$\gamma_k = \frac{s_{k-1}^\top y_{k-1}}{y_{k-1}^\top y_{k-1}}, \tag{1.2.6}$$

be an initial matrix with the formula

$$\begin{aligned} H_k &= (V_{k-1}^\top \dots V_{k-m}^\top) H_k^0 \, (V_{k-m} \dots V_{k-1}) \\ &+ \rho_{k-m} \, (V_{k-1}^\top \dots V_{k-m+1}^\top) s_{k-m} s_{k-m}^\top (V_{k-m+1} \dots \, V_{k-1}) \\ &+ \rho_{k-m+1} \, (V_{k-1}^\top \dots V_{k-m+2}^\top) s_{k-m} s_{k-m}^\top (V_{k-m+2} \dots V_{k-1}) \end{aligned}$$

$$\begin{aligned} &+ \dots \\ &+ \rho_{k-1} s_{k-1} s_{k-1}^\top, \end{aligned}$$

where $\rho_k = 1/y_k^\top s_k$, and $V_k = I - \rho_k y_k s_k^\top$. The computational cost of L-BFGS per iteration is $O(mn)$ and storage is $O(mn)$, which is quite less expensive than $O(n^2)$ when $m \ll n$.

1.2.4 Stochastic gradient methods

Stochastic gradient methods revolutionized optimization in the context of machine learning through the efficient training of models on huge datasets. These methods provide a practical and scalable approach to addressing the challenges of dealing with huge datasets with less CPU time and less storage memory. In the past few decades, stochastic methods have undergone significant advancements for real-world applications, finding utility in stochastic gradient methods across diverse fields such as machine learning, healthcare, financial modeling, computer vision, etc. Now, we see the two most popular methods 1) stochastic gradient descent (SGD) method; and 2) stochastic variance reduced gradient (SVRG) method. Consider a large-scale optimization problem (1.1.5),

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

where n is the number of samples in the given dataset and d is the dimension.

1) Stochastic gradient descent (SGD): Instead of computing the full gradient in the steepest descent method, the SGD [98] updates the model parameters using a stochastic estimate of the gradient. It computes one random individual training sample $i \in \{1, 2, \dots, n\}$ uniformly and updates x_k as

$$x_{k+1} = x_k - \alpha_k \nabla f_i(x_k),$$

where $\nabla f_i(x_k)$ is a stochastic gradient computed at x_k. Since it computes the gradient of a single sample, it has a low per-iteration cost. To ensure global convergence, SGD necessitates adopting a diminishing step size α_k due to the variance introduced by the stochastic gradient. However, this requirement often results in slow convergence. It can be anticipated that by reducing the variance of the stochastic gradient, a constant step size can be maintained, potentially resulting in a faster convergence rate. Similar to gradient descent with momentum, a certain amount of research has been done to incorporate the various momentum terms in SGD. However, in the last decade, a few methods focused on huge sparse datasets for machine learning and deep learning used the concept of adaptive gradient or adaptive momentum. Adagrad (Adaptive Gradient Algorithm) [40] is an optimization algorithm designed for machine learning tasks, particularly in the context of training deep neural networks. It adapts the learning rate for each parameter individually by scaling it inversely with the square root of the sum of historical squared gradients. Root mean square propagation (RMSProp) [60] algorithm employs a technique to make use of a moving average of the squared gradients and then divide it by the square root mean, which helps scale the learning rate for each parameter. Adaptive moment estimation, a.k.a. Adam [71] combines the benefits of both RMSProp and Adagrad. Adam computes the moving average of both gradients and squared gradients to adaptively compute the learning rates. These methods are quite useful in various applications due to their low iteration cost and faster behavior.

2) Stochastic variance reduced gradient (SVRG): The stochastic variance reduced gradient [64] builds upon the principles of SGD by reducing the variance of stochastic gradient. The stochastic variance reduced gradient (SVRG) method [64] explicitly decrese the variance via an unbiased estimate of gradient:

$$x_{k+1} = x_k - \alpha_k p_k,$$

where

$$p_k = \nabla f_i(x_k) - \nabla f_i(\tilde{x}) + \nabla F(\tilde{x}),$$

and $\tilde{x}$ is the resultant vector at the end of the previous epoch (outer iteration). In recent years, many researchers have focused on reducing the variance of stochastic gradient methods while using a constant step size [125, 99, 31]. Stochastic average gradient (SAG) [100, 99] employed biased updates, achieving a linear convergence rate. Stochastic average gradient acceleration (SAGA) [31] improves SAG by applying an unbiased estimate. Stochastic recursive gradient algorithm (SARAH) [86, 42] employs an additional term to update the iterate using the full gradient into the outer loop of the SVRG and take a step along to the accumulated direction of the past stochastic gradient as in SAGA.

1.3 Globalization techniques

Globalization techniques in nonlinear optimization refer to strategies employed to ensure that an optimization algorithm converges to a global minimum rather than getting trapped in local minima. Given the intricacies of nonlinear optimization problems characterized by intricate, non-convex objective functions featuring multiple local minima, the quest for the global optimum becomes a formidable challenge. Globalization techniques are designed to enhance the algorithm's ability to explore the solution space more thoroughly. Next, we see some common globalization techniques in nonlinear optimization.

1.3.1 Line-search technique

Line search methods involve adjusting the step size at each iteration to find a balance between exploring the solution space and converging towards the minimum. Usually, the iterative methods take an iterate as

$$x_{k+1} = x_k - \alpha_k p_k,$$

where α_k is the step size that decides how far it needs to move along direction p_k. The most common line search methods require p_k to be the descent direction, *i.e.,*

$$\nabla f(x_k)^\top p_k < 0.$$

To exactly minimize the objective function along the search direction, it is necessary to minimize the problem:

$$\phi(\alpha) = f(x_k + \alpha p_k), \quad \alpha > 0,$$

that requires too many gradients and function evaluations. Therefore, the condition $f(x_{k+1}) < f(x_k)$ is not enough to converge to an optimal point, and α should provide a sufficient decrease in the objective function. Techniques such as backtracking line search and *Wolfe conditions* are commonly used to determine suitable step sizes that ensure global convergence.

1) Armijo condition: In order to obtain a sufficient decrease, *Armijo's condition* is given as

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f(x_k)_k^\top, \tag{1.3.1}$$

where $\alpha \in (0, 1)$. Moreover, this sufficient decrease is not enough, and we need to avoid the short steps. This second requirement is called the *curvature condition*, where α_k satisfy

$$\nabla f(x_k + \alpha_k p_k)^\top p_k \geq c_2 \nabla f(x_k)^\top p_k, \tag{1.3.2}$$

where $c_2 \in (c_1, 1)$ and c_1 is the constant from (1.3.1).

2) Wolfe conditions: Together, the conditions for sufficient decrease (1.3.1) and curvature (1.3.2) are commonly referred to as the Wolfe conditions. However, to obtain the iterate in the neighborhood of the local minimizer of ϕ, the *strong Wolfe conditions*

require α satisfy

$$\begin{aligned} f(x_k + \alpha p_k) &\leq f(x_k) + c_1 \alpha \nabla f(x_k)_k^\top, \\ |\nabla f(x_k + \alpha_k p_k)^\top p_k| &\leq c_2 |\nabla f(x_k)^\top p_k|, \end{aligned}$$

with $0 < c_1 < c_2 < 1$, and it helps to get rid of approaching towards the point far minimum of ϕ.

1.3.2 Trust-region technique

Trust-region (TR) methods control the step size based on the behavior of the objective function within a specified trust-region. The algorithm adjusts the trust region size dynamically to balance local exploration and global convergence. TR methods are particularly effective when dealing with ill-conditioned problems. Consider the objective function (1.1.2). The TR method creates a region surrounding the current iteration where the model has a suitable representation of the objective function. To obtain this region, the quadratic model m_k is used, which is based on the Taylor series expansion of the objective function f around the current iterate x_k,

$$f(x_k + p) = f(x_k) + \nabla f(x_k)^\top p + \frac{1}{2} p^\top \nabla^2 f(x_k + tp) p,$$

where $t \in (0, 1)$ and m_k is given by

$$m_k(p) = f(x_k) + \nabla f(x_k)^\top p + \frac{1}{2} p^\top B_k p,$$

where B_k is any approximation of Hessian. The trust-region method solves a subproblem to obtain the next step. The subproblem can be given as

$$\begin{aligned} &\min_{p \in \mathbb{R}^n} \; m_k(p) \\ &\text{s.t.} \;\; \|p\| \leq \Delta_k, \end{aligned}$$

where Δ_k is the trust-region radius. Solving this subproblem at each iteration can be computationally expensive. Thus, practically, the TR method uses a ratio between the *actual reduction* and *predicted reduction*, that is, the ratio of the difference in the objective function and the quadratic model m_k, respectively. For search direction p_k, the ratio of the TR method can be given as

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}.$$

Whenever model m_k and the function f have almost identical values, the ratio is close to 1. Hence, it is risk-free to increase the trust region for the next step. If ratio is negative, then step is rejected as the updated function value $f(x_k + p_k)$ is greater than the $f(x_k)$. Thus, the step is rejected. There are many variants of the TR method, depending on the approximation B_k and the direction. For example, in the case of $B_k = \nabla^2 f(x_k)$, it is called the TR-Newton method, and for the TR-CG method, the CG direction can be used with the TR method.

1.3.3 Controlling the regularized parameter:

Consider the Newton's method for solving an unconstrained optimization problem (1.1.2). In the case where the Hessian $\nabla^2 f(x_k)$ is not positive definite, the Newton direction is not the descent direction. In order to obtain the nonsingular Hessian matrix, the popular strategy is to modify Hessian by adding a positive diagonal matrix or a full matrix to the $\nabla^2 f(x_k)$,

$$B_k = \nabla^2 f(x_k) + E_k,$$

where E_k can be decided accordingly so that B_k is sufficiently positive-definite. Broadly, this process is called the regularization of Hessian, or more generally, regularized Newton methods [95, 121]. The convergence properties of one of such regularized Newton methods can be seen in [120]. Similar to the Newton's method, there are numerous regularized methods for various Hessian approximations B_k, in which they are regularized by a diagonal matrix as follows:

$$B_k + \lambda_k I,$$

where the λ_k is the regularized parameter and B_k is regularized with a diagonal matrix that is a multiple of the identity matrix. There are interesting techniques to control the regularized parameter λ_k. Li et al. [77] proposed to obtain λ using the square root of the norm of gradients.

1.4 Hessian approximation using randomized matrix:

In the latter years of the preceding decade, a novel and intriguing approach to approximating the Hessian matrix emerged. While certain methods were already in existence, they were originally designed for kernel approximation rather than Hessian approximation. This was particularly applicable to linear kernels, which share similar properties with the Hessian matrix. The overarching strategy involves the utilization of diverse randomized matrices or conducting projections onto lower-dimensional subspaces to derive approximations of the Hessian or its inverse.

1.4.1 Random projection on subspace:

The fundamental concept is to acquire a low-rank Hessian approximation that does not require huge computation like Newton's method. Pilanci et al. [93] and Randomized Subspace Newton [52] advocated for the use of randomized approximation by employing random projections to a lower-dimensional subspace. This approach also incorporates random projections and sub-sampling techniques. Subsequently, this trend led to the development of various randomized methods for tackling large-scale optimization problems. A method employing a similar strategy for obtaining randomized quasi-Newton updates was introduced by Gower et al. [54]. The Nys-Newton method [111] proposed leveraging the Nyström approximation to approximate the Hessian.

1.5 Motivations and contributions

As discussed earlier, second-order approximation methods have demonstrated efficacy in addressing extensive unconstrained optimization challenges. Despite the extensive exploration of these methods, there are numerous issues that need to be considered. Some of these issues can be listed as follows:

1) **Optimal Hessian Approximation:** Determining the most suitable Hessian approximation according to the properties and structure of the objective function for both convex and nonconvex scenarios.

2) **Regularized Hessian Approximation:** Identifying optimal techniques for acquiring the regularized parameter in regularized Hessian approximations.

3) **Variance Reduction in Stochastic Gradient:** Developing strategies to mitigate variance in stochastic gradient methods.

4) **Hyperparameter Optimization:** Defining optimal strategies for hyperparameter optimization, particularly in the context of stochastic optimization methods, focuses on parameters like learning rates.

In this thesis, we carry out our study in three parts. All parts do not have a direct connection. However, each part is conceptually connected as a second-order approximation. The first part focuses on the deterministic **(in the sense that the computation of the full gradient is affordable)** method for regularized L-BFGS with line search. The second part considers the reduction in variance of stochastic gradient **(when the computation of the full gradient is expensive)** via second-order information. The third part focuses on the randomized subspace approach to approximate the Hessian **(when computation of the partial Hessian is affordable)** for both the deterministic and stochastic realms.

1.5.1 When the computation of the full gradient is affordable

addressing large-scale, unconstrained optimization problems. While the conventional L-BFGS method relies on a line search to ensure global convergence, this approach can lead to a substantial number of function evaluations. To mitigate this challenge, Sugimoto and Yamashita [106] introduced the regularized L-BFGS, incorporating a trust-region ratio to determine λ as outlined in Sugimoto's master thesis. We extend this method by integrating various techniques, including the nonmonotone technique and the simultaneous utilization of Wolfe line search with strong conditions. Specifically, we calculate the step size using Wolfe conditions only when the iteration fails to cause a sufficient decrease in the objective function value. This adaptive strategy enables the iteration to take longer steps, helping in the convergence process toward the optimal solution. We establish convergence, and we present numerical results across various test problems, demonstrating the robustness of the proposed method in solving a diverse set of problems.

1.5.2 When the computation of the full gradient is expensive

Motivated by the stochastic variance reduction gradient (SVRG) method, we aim to enhance the incorporation of curvature information from the objective function. Our novel proposal involves further variance reduction of stochastic gradients by integrating the Barzilai-Borwein (BB) method into SVRG. Remarkably, to best of our knowledge, this marks the first instance of leveraging a constant diagonal matrix as a search direction within any stochastic gradient-type method. Notably different from other methods, which typically encompass information from multiple eigenvalues, the BB approximation uniquely progresses in the direction of a single eigenvalue. This distinction is evident when considering the properties of smoothness and strongly convex functions. We introduce a BB step size and its variant, showcasing linear convergence not only for our proposed method but also for other existing SVRG variants that utilize second-order information. Numerical experiments performed on benchmark datasets highlight that our proposed method, especially with a constant step size, outperforms existing variance-reduced methods for specific test problems, especially those linked to dense datasets.

1.5.3 When the computation of the partial Hessian is affordable

Considering the approach of a randomized matrix, we introduce a novel regularized Nyström method for addressing unconstrained optimization challenges within high-dimensional feature spaces. Diverging from conventional second-order approximation methodologies, such as quasi-Newton methods that heavily rely on first-order derivatives, our approach strategically capitalizes on actual Hessian information. We propose a balanced strategy through the incorporation of the regularized Nyström approximation for both strongly convex and non-convex function, which employs partial Hessian information in the form of a thin column to approximate the Hessian, seamlessly integrating it with gradient descent and stochastic gradient descent. A comprehensive convergence analysis is presented, accompanied by an exploration of pertinent theoretical aspects. We conduct numerical experiments, evaluating the proposed method's performance across strongly convex functions and non-convex function. Notably, our results reveal the method's superior performance in comparison to randomized subspace Newton and the approximation of the Newton sketch, signifying significant advancements in optimization within high-dimensional feature spaces. Furthermore, we extend our investigation to the practical application of brain tumor detection, demonstrating the competitiveness of our method against established quasi-Newton methods and underscoring its transformative potential in critical domains.

1.6 Outline of the thesis

The arrangement of this thesis is as follows:

In Chapter 2, we discuss preliminary concepts, encompassing fundamental definitions of differentiability, convexity, and their properties, along with the essential optimality conditions crucial for subsequent chapters.

Subsequently, we provide the Table 1.1 as a comprehensive overview of the thesis, describing individual contributions within next chapters.

Table 1.1: Overview of thesis

Chapters	Problem	Based method	Convergence
Ch. 3	Unconstrained non-convex	Regularized Newton method & L-BFGS method	Global convergence
Ch. 4	Unconstrained sum of strongly convex function	Stochastic variance reduced gradient method & Barzilai-Borwein method	Linear convergence
Ch. 5	Unconstrained convex & non-convex	Nyström method	Linear & Global convergence

In Chapter 3, we propose a regularized limited memory BFGS (L-BFGS) method to solve a large-scale unconstrained optimization problem in which we use the trust-region ratio to obtain the best regularized parameter along with strong Wolfe conditions. We provide the convergence analysis of the proposed method under some mild assumptions. Finally, we show the efficiency and robustness of the proposed method in numerical results on various problems.

In Chapter 4, we consider solving a large-scale optimization problem. We propose to employ second-order information to further decrease the variance of stochastic variance reduced gradient (SVRG). We employ the Barzilai-Borwein (BB) method in the search direction of the SVRG. Moreover, we employ the BB method to obtain a better step size by incorporating it into SVRG with second-order information. We establish the linear convergence of strongly convex functions. The numerical results on large-scale datasets showcase the effectiveness of the proposed algorithms.

In Chapter 5, we consider the approach of a randomized matrix to approximate the Hessian. We use the Nyström approximation to obtain the low-rank Hessian approximation for both strongly-convex and non-convex function. We propose the regularized Nyström method for the unconstrained optimization problem. We show convergence and establish various theoretical properties. We then demonstrate the numerical experiments by comparing the proposed method with existing *state-of-the-art* methods for various optimization problems for strongly convex functions. Moreover, we show an application for detecting a brain tumor in the MRI and demonstrate the numerical results to show the efficacy of proposed algorithm.

Finally, in Chapter 6, we provide concluding remarks, explore potential extensions, and outline future research.

Chapter 2

Preliminaries

In this chapter, we give some basic concepts of convexity, differentiability, Lipschitz continuity, and optimality conditions. In the last few subsections, we introduce the basic properties of expectation and variance. In this chapter, most of the results largely follow from the [7, 83, 89, 10].

2.1 Vectors and matrices

For a set S, an element x in the set S is denoted by $x \in S$. In this thesis, we consider the Euclidean space (set of real numbers), which is denoted by $\mathbb{R}$. For n-dimensional real space, we denote $\mathbb{R}^n$. Vectors of n-dimensional space are usually denoted by lowercase roman characters and matrices by uppercase roman characters. Vector $x \in \mathbb{R}^n$ is column vector x that is

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

where x_i is the ith element of vector x. The transpose of vector x is the row vector denoted by $x^\top$ and it can be written as,

$$x^\top = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^\top = \begin{bmatrix} x_1 \ x_2 \ \dots \ x_n \end{bmatrix}.$$

We denote $x \leq 0$ ($x \geq 0$), to indicate component-wise notation, *i.e.*, $x_i \leq 0$ ($x_i \geq 0$), for all $i \in \{1, 2, \dots, n\}$ and $x < 0$ ($x > 0$) indicates $x_i < 0$ ($x_i > 0$) for all $i \in \{1, 2, \dots, n\}$. Let x and y be two n-dimensional real vectors in $\mathbb{R}^n$, then its Euclidean inner product $\langle x, y \rangle$ is defined by

$$\langle x, y\rangle = x^\top y = \sum_{i=1}^{n} x_i y_i.$$

Euclidean norm (a.k.a. ℓ_2 norm) of vector x is defined as $\|x\| = \sqrt{x^\top x} = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$. It is easy to see that $x^\top x > 0$ and $\|x\| = 0$ if and only if $x = 0$. Vectors x and y are called orthogonal if $x^\top y = 0$. The norm of vector $\|x + y\|$ is upper bound by $\|x\|$ and $\|y\|$, that is, $\|x + y\| \leq \|x\| + \|y\|$ called the triangle inequality. For a vector $x \in \mathbb{R}^n$,

$$(x)_+ = \max(0, x).$$

We denote the ℓ_1 norm of vector $x \in \mathbb{R}^n$ by $\|x\|_1 = |x_1| + \cdots + |x_n|$.

$A \in \mathbb{R}^{m\times n}$ matrix of dimension $m \times n$ is an array of real numbers a_{ij} that is,

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{n1} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix};$$

where elements of the matrix A are denoted by a_{ij} (which ith row and jth column). The transpose of matrix A is $A^\top$ can be written as,

$$A^\top = \begin{bmatrix} a_{11} & a_{21} & \ldots & a_{m1} \\ a_{12} & a_{22} & \ldots & a_{m2} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \ldots & a_{mn} \end{bmatrix};$$

where the elements of $A^\top$ can be denoted by a_{ji}. When $m = n$, the matrix A is called the square matrix. A square matrix A is called symmetric if $A = A^\top$. A square matrix $A \in \mathbb{R}^{n\times n}$ is called positive definite if there exists a positive scalar c such that

$$x^\top A x \geq x^\top x, \text{ for all } x \in \mathbb{R}^n.$$

A matrix A is positive semidefinite if

$$x^\top A x \geq 0.$$

A matrix A is called a nonsingular matrix for a vector $b \in \mathbb{R}^n$, if there is a vector $w \in \mathbb{R}^n$ such that $Aw = b$. A matrix A^{-1} is called an inverse of the nonsingular square matrix A, such that $A^{-1}A = AA^{-1} = I$, where I is an n-dimensional identity matrix. For two square invertible matrices $A, B \in R^{n\times n}$, $(AB)^{-1} = B^{-1}A^{-1}$. The eigenvalues of square matrix A are the scalars λ_i such that for a column vector $x \in \mathbb{R}^n$, the value of matrix-vector multiplication Ax is equal to λx, that is

$$Ax = \lambda_i x, \text{ for all } i \in \{1, 2, \dots, n\},$$

where λ_i is an eigenvalue of A corresponding to an eigenvector vector x. We define the ℓ_2 norm of a matrix $A \in \mathbb{R}^{m\times n}$ as

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^\top A)},$$

where $\lambda_{\max}$ is the largest eigenvalue of matrix A. The Frobenious norm matrix $A \in \mathbb{R}^{m\times n}$ is defined as

$$\|A\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2}.$$

For two matrices A and B, the following inequality holds:

$$\|AB\| \le \|A\|\|B\|$$

where both matrices has consistent dimensions.

2.2 Convexity and differentiability

Definition 2.2.1. *A subset $C \subseteq \mathbb{R}^n$ is said to be convex if*

$$\alpha x + (1-\alpha)y \in C, \quad \textit{for all } x, y \in C, \ \forall \alpha \in [0, 1].$$

Definition 2.2.2. *For a subset $C \subseteq \mathbb{R}^n$, a scalar function $f : C \to \mathbb{R}$ is said to be convex if*

$$f(\alpha x + (1-\alpha)y) \le \alpha f(x) + (1-\alpha)f(y), \quad \forall x, y, \in C, \ \alpha \in [0, 1].$$

Definition 2.2.3. *For a subset $C \subseteq \mathbb{R}^n$, a scalar function $f : C \to \mathbb{R}$ is said to be strictly convex if*

$$f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y), \quad \alpha \in [0, 1],$$

for all $x, y, \in C$, when $x \neq y$.

Definition 2.2.4. *For a subset $C \subseteq \mathbb{R}^n$, a scalar function $f : C \to \mathbb{R}$ is said to be μ-strongly* *convex if*

$$f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y) - \frac{\alpha(1-\alpha)\mu}{2}\|x-y\|^2, \quad \alpha \in [0,1],$$

for all $x, y, \in C$, and $\mu > 0$.

μ is called a strongly convex parameter.

Definition 2.2.5. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function and fix a vector $x \in \mathbb{R}^n$, then*

$$\lim_{\alpha \to 0} \frac{f(x + \alpha e_i) - f(x)}{\alpha},$$

where e_i is the ith unit vector, where the ith element is 1 and all other components are 0. If the above limit exists, it is the partial derivative of f at the vector x. It is denoted by $\partial f(x)/\partial x_i$. The function $f : \mathbb{R}^n \to \mathbb{R}$ is called differentiable if it is differentiable at all $x \in \mathbb{R}^n$.

Assuming a limit exists for all $i \in \{1, 2, \dots, n\}$, we introduce the gradient.

Definition 2.2.6. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. The gradient of f at x is an n-dimensional vector defined by*

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix},$$

where $\frac{\partial f(x)}{\partial x_i}$ is a partial derivative of function f with respect to ith component x_i.

If $\nabla f(x)$ is continuous, then f is said to be continuously differentiable on $\mathbb{R}^n$. Also, such a function is called a smooth function.

Definition 2.2.7. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function, then the Hessian matrix $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$ of f at x is*

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}.$$

Definition 2.2.8. *A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be a L-Lipschitz continuous gradient if for a constant $L > 0$*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \textit{for all } x, y \in \mathbb{R}^n.$$

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function and its gradient is L-Lipschitz, then the following conditions are equivalent to a Lipschitz continuous gradient.

1) $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y,$

2) $f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \dfrac{L}{2}\|y - x\|^2, \quad \forall x, y,$

3) $(\nabla f(x) - \nabla f(y))^\top (x - y) \leq L\|x - y\|^2, \quad \forall x, y.$

Proposition 2.2.1 ([10]). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function; then the function f is convex if and only if*

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x), \forall x, y \in \mathbb{R}^n.$$

The above inequality is strict if and only if f is strictly convex.
Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function that is strongly convex with $\mu > 0$, then the following are equivalent.

1) $f(\alpha x + (1 - \alpha)y) < \alpha f(x) + (1 - \alpha)f(y) - \dfrac{\alpha(1 - \alpha)\mu}{2}\|x - y\|^2, \quad \alpha \in [0, 1],$

2) $f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \dfrac{\mu}{2}\|y - x\|^2, \quad \forall x, y,$

3) $(\nabla f(x) - \nabla f(y))^\top (x - y) \geq \mu\|x - y\|^2, \quad \forall x, y.$

2.3 Optimal solution and optimal conditions

In this subsection, we introduce the local and global minima and first and second order optimality conditions of the unconstrained objective function (1.1.2). In a minimization problem, we seek to get a global minimizer of the function f. Basically, it is a point where the underlying function f attains its least value. However, it is not always possible to find the global minima since we usually have only local information about f. Therefore, we seek to get the minimum value of the function in some neighborhoods. More detailed information can be found in [8, 89].

Definition 2.3.1. *Let* (1.1.2) *be an unconstrained minimization problem. A vector x_* is said to be an unconstrained local minimum of a function f if there is a neighborhood $\mathcal{N}$ of x_* such that*

$$f(x_*) \leq f(x), \quad \forall x \in \mathcal{N}.$$

We say that x_* is strict local minimizer if there exists a neighborhood $\mathcal{S}$ of x_* such that the above inequality becomes strict, *i.e.,*

$$f(x_*) < f(x), \quad \forall x \in \mathcal{S}, \text{ with } x \neq x_*.$$

Definition 2.3.2. *Let* (1.1.2) *be an unconstrained minimization problem. A vector x_* is said to be an unconstrained global minimum of a function f, if $f(x_*)$ attains its minimum value, that is,*

$$f(x_*) \leq f(x), \quad \forall x \in \mathbb{R}^n.$$

Theorem 2.3.1 (Necessary optimality conditions [8])**.** *Let* (1.1.2) *be an unconstrained minimization problem and f, and x_* be a local minimum of f. Assume that is continuously differentiable in an open set $\mathcal{N}$ that contains x_*, then*

$$\nabla f(x_*) = 0. \qquad \textit{(First order necessary condition)}$$

Moreover, if $\nabla^2 f(x)$ exists and f is twice continuously differentiable in $\mathcal{N}$, then

$$\nabla^2 f(x_*) \textit{ is positive semidefinite.} \qquad \textit{(Second order necessary condition)}$$

Next, we introduce the sufficient condition, which is a certain condition on the derivative of f such that it guarantees that x_* is a local minimizer.

Theorem 2.3.2 (Second order sufficient condition [8])**.** *Suppose that* (1.1.2) *be an objective function and f is a twice continuously differentiable function. Let $\nabla^2 f(x)$ be a continuous in a neighborhood of x_* and x_* is a strict local minimizer of f if $\nabla f(x_*) = 0$ and $\nabla^2 f(x_*)$ is positive definite.*

Definition 2.3.3. *Let* (1.1.2) *unconstrained minimization problem, and p_k is some search direction of some solution method. Then p_k is called descent direction if*

$$\nabla f(x_k)^\top p_k < 0.$$

Theorem 2.3.3 ([89])**.** *Let f be a convex function. Any local minimizer x_* of f is a global minimizer of f. Moreover, if f is differentiable, then any stationary point x_* is also a global minimizer of f.*

Proposition 2.3.1. *Let f be a continuously differentiable function, then the following conditions are equivalent by strong convexity. μ is a strongly convex parameter, x_* is a minimizer, and $f(x_*)$ is the optimal value of f.*

1) $\dfrac{1}{2}\|\nabla f(x)\|^2 \geq \mu(f(x) - f(x_*)), \quad \forall x.$

2) $\|\nabla f(x) - \nabla f(y)\| \geq \mu \|x - y\|, \quad \forall x, y.$

3) $f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \dfrac{1}{2\mu} \|\nabla f(y) - \nabla f(x)\|^2, \quad \forall x, y.$

4) $(\nabla f(x) - \nabla f(y))^\top (x - y) \leq \dfrac{1}{\mu} \|\nabla f(x) - \nabla f(y)\|^2, \quad \forall x, y.$

2.4 Expectation and Variance

In this section, we discuss the basic probability theory in terms of the gradient of the objective function. First, let X be any discrete random variable.

Definition 2.4.1. *The expectation of a random variable is the mean of the N values,* i.e.,

$$\mathbb{E}[X] = \sum_x x f_X(x) = \sum_x x \mathbb{P}(X = x),$$

where $f_X(x)$ is a probability function.

Definition 2.4.2. *Let X be any random variable. Then the variance of X is the mean squared deviation of a random variable from its mean,* i.e.,

$$Var[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

Next, consider a dataset $X \in \mathbb{R}^{n \times d}$ that has n samples and d features or parameters.

Expectation of stochastic gradient: Let (1.1.5) be an unconstrained minimization problem. Then, for a stochastic gradient method, let x_k be the computed stochastic gradient at ith sample, which is $\nabla f_i(x_k)$, then the expectation of the stochastic gradient is

$$\mathbb{E}[\nabla f_i(x_k)] = \frac{1}{n} \sum_{j=1}^{n} \nabla f_j(x_k) = \nabla f(x_k).$$

Thus, the expectation of the stochastic gradient is a full gradient.

Chapter 3

A Regularized Limited Memory BFGS method with simultaneous use Wolfe line search for Large-Scale Unconstrained Optimization

3.1 Introduction

Consider the problem of large-scale unconstrained optimization by recalling objective function (1.1.1):

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x), \tag{3.1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice-continuously differentiable function. In order to solve it, an iteration of the quasi-Newton-type method can be given as

$$x_{k+1} = x_k + d_k,$$

where $x_k \in \mathbb{R}^n$ is the k^{th} iteration and $d_k \in \mathbb{R}^n$ is search direction.

Classical solution approaches for solving (3.1.1), such as the steepest descent method, Newton's method, and the BFGS method [34, 89], prove unsuitable for large-scale problems. The steepest descent method often shows convergence; the Newton's method computes a Hessian matrix and solves a linear equation at each iteration; and the BFGS method poses challenges for large-scale problems due to requiring $O(n^2)$ memory for storing and computing the approximate Hessian of f.

A prominent quasi-Newton method designed for large-scale problems is the limited memory BFGS (L-BFGS) [78, 87], which efficiently utilizes stored limited memory pairs to approximate the Hessian of f. It stores the past m vector pairs of (s_{k-i}, y_{k-i}), $i = 0, 1, \ldots, m-1$,, where

$$s_k = x_{k+1} - x_k, \text{ and } y_k = \nabla f(x_{k+1}) - \nabla f(x_k),$$

and computes d_k in $O(mn)$ time.

The L-BFGS method incorporates the Wolfe line search to ensure global convergence. However, the line search can sometimes require a greater number of function evaluations. Some trust region methods (TR-method) ensure global convergence, but they require a lesser number of function evaluations compared to the line search [14, 15, 80]. The integration of the L-BFGS method with the TR-method [14, 15] demonstrates competitive performance across various benchmark problems, notably decreasing the number of function evaluations. However, the TR-method requires solving a constrained subproblem.

$$\begin{aligned} &\text{minimize} \quad f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ &\text{subject to} \quad \|d\| \leq \Delta_k, \end{aligned} \tag{3.1.2}$$

in all steps, where approximate Hessian B_k is obtained by L-BFGS and Δ_k is the trust-region radius. This process takes a significant amount of time to solve (3.1.2).

It is notably important that Burakov et al. [13] employ joint limited-memory and trust region techniques.It exploits the eigenvalue decomposition of B_k in order to solve the TR-subproblem.

In order to solve these issues, we consider utilizing a regularization technique instead of the TR-method. This is motivated by the regularized Newton method proposed by Ueda and Yamashita [120, 68, 119] and in regularized L-BFGS [106]. The regularized Newton method calculates a search direction d_k by solving the linear equations given below:

$$(\nabla^2 f(x_k) + \mu_k I)d = -\nabla f(x_k), \tag{3.1.3}$$

and regularized L-BFGS method computes a search direction d_k as a

$$(B_k + \mu_k I)d = -\nabla f(x_k), \tag{3.1.4}$$

where $\mu_k > 0$. If $B_k = \nabla^2 f(x_k)$ and μ_k aligns with the optimal Lagrange multiplier's value at a solution of problem (3.1.2), then d_k serves as a solution to (3.1.2). Notably, the linear equations (3.1.4) prove simpler compared to the subproblem (3.1.2) of TR-method.

In this chapter, we introduce a methodology that combines the regularized L-BFGS method with the simultaneous use of Wolfe line search. We call the proposed method the regularized L-BFGS-SW method. A straightforward implementation involves utilizing a solution to the subsequent equations as a search direction,

$$(B_k + \mu I)d = -\nabla f(x_k), \tag{3.1.5}$$

where B_k is an approximate Hessian given by a certain quasi-Newton method. However, when B_k is calculated by the L-BFGS method, it is difficult to compute $(B_k + \mu I)^{-1}$ [2] Hence, regularized L-BFGS involves directly constructing $(B_k + \mu I)^{-1}$ using the L-BFGS method for

$$f(x) + \mu\|x\|^2,$$

In this context, we employ $(s_k, \hat{y}_k(\mu))$ with $\hat{y}_k(\mu) = y_k + \mu s_k$, as opposed to (s_k, y_k). The inclusion of μs_k in $\hat{y}_k(\mu)$ serves as a regularization term. Consequently, the computation of the search direction d_k can be achieved in $O(mn)$ time, mirroring the computational efficiency of the traditional L-BFGS method.

A drawback of the regularized L-BFGS method is that a step d_k sometimes becomes small, and it causes a large number of iterations. To get a longer step, we propose two techniques: a nonmonotone technique and simultaneous use of the Wolfe line search. Recall that the step length given by the Wolfe condition is allowed to be larger than 1, and hence the step can explore a larger area. Thus, if $f(x_k + \alpha d_k) < f(x_k + d_k)$ for $\alpha > 1$, it would be reasonable to find α via the Wolfe line search. We note that this chapter is an enhancement of the unpublished paper [106] that does not exploit the line search. Due to the simultaneous use of line search, the proposed method becomes more robust than that of [106], which we report in a further section.

A limitation of the regularized L-BFGS method is the unit step length, leading to an increased number of iterations. To address this, we introduce two techniques: a nonmonotone approach and the simultaneous incorporation of the Wolfe line search. It's worth noting that, under the Wolfe condition, the step length can extend beyond 1, enabling exploration of a larger region. Therefore, if $f(x_k + \alpha d_k) < f(x_k + d_k)$ for $\alpha > 1$, determining α through the Wolfe line search is a reasonable strategy. It's crucial to highlight that this chapter represents an advancement over the unpublished paper [106], which did not employ the line search. The simultaneous application of the line search makes the proposed method more robust than that presented in [106], as discussed in further section.

[2]Quite recently, Steck and Kanzow [66] proposed an efficient calculation technique for $(B_k + \mu I)^{-1}$.

3.2 The regularized L-BFGS method

In this section, we present the regularized L-BFGS technique, which oversees the regularized parameter across each iteration. Here, x_k represents the k-th iterative point, B_k denotes the approximate Hessian of $f(x_k)$, and $H_k^{-1} = B_k$. Since regularized L-BFGS computes a search direction d_k as a solution to

$$(B_k + \mu I)d_k = -\nabla f(x_k). \tag{3.2.1}$$

However, since the L-BFGS method updates B_k, constructing B_k explicitly poses challenges. Furthermore, even if we manage to obtain B_k, solving the linear equation (3.2.1) in large-scale scenarios demands a considerable amount of computational time.

In the regularized L-BFGS method, $B_k + \mu I$ is considered an approximation of $\nabla^2 f(x) + \mu I$. Given that B_k is the approximate Hessian of $f(x_k)$, the matrix $B_k + \mu I$ serves as an approximate Hessian for $f(x) + \frac{\mu}{2}\|x\|^2$. Unlike the L-BFGS method, which employs the vector pair (s_k, y_k) to construct the approximate Hessian, where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, the regularized L-BFGS method introduces a modification. In this case, we use the gradients of $f(x) + \frac{\mu}{2}\|x\|^2$ in the computation, leading to the adoption of the modified vector $\hat{y}_k(\mu)$ instead of y_k as follows:.

$$\hat{y}_k(\mu) \ = \ (\nabla f(x_{k+1}) + \mu x_{k+1}) - (\nabla f(x_k) + \mu x_k) = \ y_k + \ \mu \ s_k.$$

Consider $\hat{H}_k(\mu)$ as a matrix constructed by the L-BFGS method utilizing m vector pairs $(s_i, \hat{y}_i(\mu))$, $i = 1, \ldots, m$, along with a suitable initial matrix $\hat{H}_k^{(0)}(\mu)$. Consequently, the computation of $d_k = -\hat{H}_k(\mu)\nabla f(x_k)$ requires $O(mn)$ time, which is similar to the computational efficiency of the original L-BFGS. It's noteworthy that $\hat{H}_k(\mu)$ retains positive definiteness if $s_k^T \hat{y}_k(\mu) > 0$, and $\hat{H}_k^{(0)}$ is positive definite. In instances where $s_k^T \hat{y}_k(\mu) > 0$ is not satisfied, a potential substitution involves replacing $\hat{y}_k(\mu)$ with $\tilde{y}_k(\mu)$.

$$\tilde{y}_k(\mu) = y_k + \left(\max\left\{0, \frac{-s_k^T y_k}{||s_k||^2}\right\} + \mu \right) s_k.$$

Then, the inequality $s_k^T \tilde{y}_k(\mu) > 0$ always holds because

$$s_k^T \tilde{y}_k(\mu) = \max\{0, s_k^T y_k\} + \mu \|s_k\|^2 > 0.$$

Here, $\hat{H}_k(\mu)$ represents the matrix constructed by the L-BFGS method using the initial matrix $\hat{H}_k^{(0)}(\mu)$ and the vector pairs $(s_{k-i}, \hat{y}_{k-i}(\mu))$, $i = 1, \cdots, m$. The corresponding search direction is defined as $d_k(\mu) = -\hat{H}_k(\mu)\nabla f(x_k)$. In the conventional L-BFGS method, $\gamma_k I$ serves as the initial matrix $H_k^{(0)}$, where γ_k is a specific positive constant. Given that

$(B_k^{(0)})^{-1} = H_k^{(0)}$ and $\hat{H}_k^{(0)}$ approximates $(B_k^{(0)} + \mu I)^{-1}$, we may set the initial matrix $\hat{H}_k^{(0)}(\mu)$ as

$$\hat{H}_k^{(0)}(\mu) = (B_k^{(0)} + \mu I)^{-1} = \left(\frac{1}{\gamma_k} + \mu\right)^{-1} I = \frac{\gamma_k}{1+\gamma_k \mu} I. \tag{3.2.2}$$

The regularized L-BFGS approach computes the subsequent iteration as $x_{k+1} = x_k + d_k(\mu)$ without involving a step length. The parameter μ is controlled to ensure global convergence following the methodology proposed in [120]. To determine a suitable search direction and utilize the notion of updating the trust-region radius in the TR-method to control μ. Specifically, it leverages the ratio of the reduction in the objective function value to that of the model function value, defining a ratio function $r_k(d_k(\mu), \mu)$ as

$$r_k(d_k(\mu), \mu) = \frac{f(x_k) - f(x_k + d_k(\mu))}{f(x_k) - q_k(d_k(\mu), \mu)}, \tag{3.2.3}$$

where $q_k : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is given by

$$q_k(d_k(\mu), \mu) = f(x_k) + \nabla f(x_k)^T d_k(\mu) + \frac{1}{2} d_k(\mu)^T \hat{H}_k(\mu)^{-1} d_k(\mu).$$

It is worth noticing that there is no requirement for an explicit computation of the matrix $\hat{H}_k(\mu)^{-1}$ in $q_k(d_k, \mu)$. Given that $d_k(\mu) = -\hat{H}_k(\mu)\nabla f(x_k)$, the expression $d_k(\mu)^T \hat{H}_k(\mu)^{-1} d_k(\mu)$ simplifies to $-d_k(\mu)^T \nabla f(x_k)$. When the ratio $r_k(d_k(\mu), \mu)$ is sizable, indicating a significant reduction in the objective function f compared to the model function, we opt for $d_k(\mu)$ and decrease the parameter μ. Conversely, if $r_k(d_k, \mu)$ is small, signifying a modest decrease in $f(x_k)$, we increase μ and recompute $d_k(\mu)$. Now, we get into the details of the regularized L-BFGS method, given in Algorithm 1.

Algorithm 1 Regularized L-BFGS

```
Step 0 Choose the parameters $\mu_0, \mu_{min}, \sigma_1, \sigma_2, \eta_1, \eta_2, m$ such that $0 < \mu_{min} \leq \mu_0, 0 < \sigma_1 \leq 1 < \sigma_2, 0 < \eta_1 < \eta_2 \leq 1$ and $m > 0$. Choose initial point $x_0 \in \mathbb{R}^n$ and an initial matrix $\hat{H}_k^0$. Set $k := 0$.
Step 1 If some stopping criteria are satisfied, then terminate. Otherwise, go to step 2.
Step 2
Step 2-0 Set $l_k := 0$ and$\bar{\mu}_{l_k} = \mu_k$.
Step 2-1 Compute $d_k(\bar{\mu_{l_k}})$ using Algorithm 2.
Step 2-2 Compute$r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k})$. If $r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k}) < \eta_1$, then update $\bar{\mu}_{l_{k+1}} = \sigma_2 \bar{\mu}_{l_k}$, set $l_k = l_k + 1$, and go to Step 2-1. Otherwise, go to Step 3.
Step 3 If $\eta_1 \leq r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k}) < \eta_2$ then update $\mu_{k+1} = \bar{\mu}_{l_k}$. If $r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k}) \geq \eta_2$ then update $\mu_{k+1} = \max[\mu_{\min}, \sigma_1 \bar{\mu_{l_k}}]$. Update $x_{k+1} = x_k + d_k(\bar{\mu}_{l_k})$. Set $k = k + 1$ and go to Step 1.
```

In Step 2-1, we compute $d_k(\mu)$ from $(s_k, \hat{y}_k(\mu))$ by the L-BFGS updating scheme in [87].

The details of step 2-1 are given as follows:.

```
Algorithm 2  L-BFGS with $(s_k, \hat{y}_k(\mu))$
Step 0 Set $p \leftarrow \nabla f(x_k)$.
Step 1 Repeat the following process with $i = k-1, k-2, \dots, k-l$;
           $r_i \leftarrow \tau_i s_i^T p,$
           $p \leftarrow p - r_i(y_k + \mu s_k),$
       where $\tau_i = (s_i^T(y_k + \mu s_k))^{-1}$.
Step 2 Set $q \leftarrow \hat{H}_k^0(\mu)p$.
Step 3 Repeat the following process with $i = k-t, k-t+1, \dots, k-1$;
           $\beta \leftarrow \tau_i(y_k + \mu s_k)^T q,$
           $q \leftarrow q + (r_i - \beta)s_i.$
Step 4 Get the search direction by $d_k(\mu) = -q$.
```

Remark 3.2.1. Tarzangh and Peyghami [113] proposed the L-BFGS with,

$$\bar{y}_k = y_k + \beta_k \frac{\omega_k}{s_k^T u} u, \tag{3.2.4}$$

where $u \in \mathbb{R}^n$ is a vector such that $s_k^T u \neq 0$, $\omega_k = (2(f(x_k) - f(x_{k+1})) + (g_k + g_{k+1}))$ and $\beta_k \in \{0, 1, 2, 3\}$. The additional term $\beta \frac{\omega_k}{s_k^T u} u$ is considered a regularization. Note that [113] does not control u for global convergence. Instead it adopts the usual line search.

It is crucial to note that the regularized parameter is pretty sensitive and requires a balanced choice of regularized parameter as it significantly affects the trust-region ratio and hence convergence. Also, it uses a few user-defined constants to tune the regularized parameter μ. Therefore, the optimal trust-region ratio can be determine through a thoughtful tuning process which involves with the experimenting with appropriate constants.

3.3 Regularized L-BFGS with simultaneous use with Wolfe line search

The regularized L-BFGS method does not utilize a line search, and hence it cannot take a longer step. Usually, the iterate with line search can be given as

$$x_{k+1} = x_k + \alpha_k d_k, \tag{3.3.1}$$

where α_k is a step length. The usual L-BFGS [78, 87] uses a step length α_k that satisfies the Wolfe conditions,

$$\begin{aligned} f(x_k + \alpha_k d_k) &\leq f(x_k) + c_1 \alpha_k d_k^T \nabla f(x_k); && (3.3.2) \\ d_k^T \nabla f(x_k + \alpha_k d_k) &\geq c_2 d_k^T \nabla f(x_k); && (3.3.3) \\ |d_k^T \nabla f(x_k + \alpha_k d_k)| &\leq c_2 |d_k^T \nabla f(x_k)|, && (3.3.4) \end{aligned}$$

where $0 < c_1 < c_2 < 1$. Please note that α_k can exceed 1. Consequently, $\alpha_k d_k$ has the potential for a significant reduction in f. Considering this, employing a line search in addition to the regularization technique seems reasonable. However, determining α_k is time consuming, so it's essential to avoid it if $\alpha_k d_k$ does not provide sufficient improvement.

To optimize the effectiveness of conditional use of line search, we rely on the curvature condition (3.3.3), ensuring that the step is not too short. Following the completion of step 3 in Algorithm 1, we initially check whether $x_k + d_k(\mu)$ satisfies the curvature condition (with $\alpha_k = 1$ in (3.3.3)) or not. If the condition is not satisfied, signifying that $x_k + d_k(\mu)$ is a short step, we compute α_k using the strong Wolfe condition. This computation aims to facilitate a longer step. Specifically, we search for α_k from $x_k + d_k$ in the direction of d_k so that (3.3.2)-(3.3.4) hold with $x_k := x_k + d_k(\mu)$, $d_k := d_k(\mu)$, and then set $x_{k+1} = x_k + (1+\alpha_k) d_k(\mu)$.

Next, we discuss the conditions under which we conduct the Wolfe line search. As mentioned previously, we exploit the strong Wolfe condition (3.3.3) when the following conditions hold,

$$d_k(\mu)^T \nabla f(x_k + d_k(\mu)) < c_2 d_k(\mu)^T \nabla f(x_k) \quad \text{and} \quad \mu = \mu_{min}. \qquad (3.3.5)$$

Keep in mind that $\|d_k(\mu_{min})\| \geq \|d_k(\mu)\|$ holds when $\mu > \mu_{min}$. Consequently, $d_k(\mu_{min})$ represents the most substantial step achievable when applying RL-BFGS alone. The condition (3.3.5) indicates that whenever $x_k + d_k(\mu_{min})$ promises progress, we opt for an extended step through a robust Wolfe line search. This approach, an augmentation of the proposed method, is termed Regularized L-BFGS with Strong Wolfe Line Search (RL-BFGS-SW). The RL-BFGS-SW method is introduced below.

For this process, we use s_k and $\hat{y}_k(\mu)$ whenever the strong Wolfe line search is used. We summarize the details of $y_k, \hat{y}_k(\mu)$ and s_k in Table 3.1.

Algorithm 3 RL-BFGS with line search (RL-BFGS-SW)

```
Step 0 Choose the parameters $\mu_0, \mu_{min}, \sigma_1, \sigma_2, \eta_1, \eta_2, m$ such that $0 < \mu_{min} \le \mu_0, 0 < \sigma_1 \le 1 < \sigma_2, 0 < \eta_1 < \eta_2 \le 1$ and $m > 0$. Choose initial point $x_0 \in \mathbb{R}^n$ and an initial matrix $\hat{H}_k^0$. Set $k := 0$.
Step 1 If some stopping criteria are satisfied, then terminate. Otherwise go to step 2.
Step 2
Step 2-0 Set $l_k := 0$ and $\bar{\mu}_{l_k} = \mu_k$.
Step 2-1 Compute $d_k(\bar{\mu_{l_k}})$ by Algorithm 2.
Step 2-2 Compute $r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k})$. If $r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k}) < \eta_1$, then update
    $\bar{\mu}_{l_{k+1}} = \sigma_2 \bar{\mu}_{l_k}$, set $l_k = l_k + 1$, and go to Step 2-1. Otherwise, go to Step 3.
Step 3 If $\eta_1 \le r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k}) < \eta_2$ then update $\mu_{k+1} = \bar{\mu}_{l_k}$.
    If $r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k}) \ge \eta_2$ then update $\mu_{k+1} = \max[\mu_{\min}, \sigma_1 \bar{\mu_{l_k}}]$.
Step 4 If $d_k(\mu)^T \nabla f(x_k + d_k(\mu)) < c_2 d_k(\mu)^T \nabla f(x_k)$ and $\mu_k = \mu_{min}$,
    then find $\alpha_k$ by strong Wolfe line search and set $x_{k+1} = x_k + d_k + \alpha_k d_k$,
    otherwise $x_{k+1} = x_k + d_k$.
Step 5 Set $k = k + 1$ and go to Step 1.
```

Table 3.1: Comparison of $y_k, \hat{y}_k$ and s_k.

L-BFGS	RL-BFGS	RL-BFGS-SW (when line search is used)
$x_{k+1} = x_k + \alpha_k d_k$	$x_{k+1} = x_k + d_k(\mu)$	$x_{k+1} = x_k + (\alpha_k + 1) d_k(\mu)$
$s_k = \alpha_k d_k$	$s_k = d_k(\mu)$	$s_k = (\alpha_k + 1) d_k(\mu)$
$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$	$\hat{y}_k(\mu) = y_k + \mu_k d_k(\mu)$	$\hat{y}_k(\mu) = y_k + \mu_k(\alpha_k + 1) d_k(\mu)$

3.4 Global convergence

In this section, we show the global convergence of the regularized L-BFGS method with certain modifications. First, we give a definition of Lipschitz continuity.

Definition 3.4.1 (Lipschitz continuity). *Let S be a subset of $\mathbb{R}^n$ and $f : S \to \mathbb{R}$.*

i) The function f is said to be Lipschitz continuous on S if there exists a positive constant L_f such that

$$|f(x) - f(y)| \le L_f \|x - y\| \ \forall x, y \in S.$$

ii) Suppose that the function f is differentiable. ∇f is said to be Lipschitz continuous on S if there exists a positive constant L_g such that

$$\|\nabla f(x) - \nabla f(y)\| \le L_g \|x - y\| \ \forall x, y \in S.$$

Next, we need the following assumptions.

Assumption 3.4.1.

(i) *The objective function f is twice continuously differentiable.*

(ii) *The level set of f at the initial point x_0 is compact, i.e., $\Omega = \{x \in \mathbb{R}^n | f(x) \le f(x_0)\}$ is compact.*

(iii) *There exist positive constants M_1 and M_2 such that*

$$M_1\|z\|^2 \le z^T\nabla^2 f(x)z \le M_2\|z\|^2 \ \forall x \in \Omega \quad \textit{and} \quad z \in \mathbb{R}^n.$$

(iv) *There exists a minimum $f_{\min}$ of f.*

(v) *There exists a constant $\underline{\gamma}$ such that $\gamma_k \ge \underline{\gamma} > 0$ for all k, where γ_k is a parameter in (3.2.2).*

These assumptions are similar to those for the global convergence of the original L-BFGS method [78].

Under above assumptions, we have the following properties. First, let

$$G(x) = \nabla^2 f(x), \quad G_k = G(x_k), \quad \bar{G}_k = \int_0^1 G(x_k + \tau d_k)d\tau.$$

It then follows from mean value theorem that

$$y_k = \bar{G}_k s_k, \tag{3.4.1}$$

as $s_k = x_{k+1} - x_k = (x_k + d_k) - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ and hence we have

$$\hat{y}_k(\mu) = y_k + \mu s_k = (\bar{G}_k + \mu I)s_k. \tag{3.4.2}$$

It follows from Assumption 3.4.1 (iii) that $\lambda_{\min}(\bar{G}_k) \ge M_1$ and $\lambda_{\max}(\bar{G}_k) \le M_2$. Therefore, we have that

$$\begin{aligned} M_1\|s_k\|^2 &\le s_k^T y_k \le M_2\|s_k\|^2, \\ \frac{1}{M_2}\|y_k\|^2 &\le s_k^T y_k \le \frac{1}{M_1}\|y_k\|^2, \\ (M_1+\mu)\|s_k\|^2 &\le s_k^T \hat{y}_k(\mu) \le (M_2+\mu)\|s_k\|^2. \end{aligned} \tag{3.4.3}$$

Given that the sequence x_k is contained within the compact set Ω, and under the assumptions in Assumption 3.4.1 (i) and (ii), where f is twice continuously differentiable, there exists a positive constant L_f such that

$$\|\nabla f(x_k)\| \leq L_f \quad \text{for all } k. \tag{3.4.4}$$

Next, we get into the characteristics of the eigenvalues of $\hat{B}_k(\mu)$, the inverse of $\hat{H}_k(\mu)$. It's important to note that the matrix $\hat{B}_k(\mu)$ is shaped by the BFGS formula, utilizing vector pairs $(s_k, \hat{y}_k(\mu))$, with the initial matrix $\hat{B}_k^{(0)}(\mu) = \hat{H}_k^{(0)}(\mu)^{-1}$. Thus, we have

$$\begin{aligned} \hat{B}_k(\mu) &= \hat{B}_k^{(\tilde{m}_k)}(\mu) \\ \hat{B}_k^{(l+1)}(\mu) &= \hat{B}_k^{(l)}(\mu) - \frac{\hat{B}_k^{(l)}(\mu)s_{j_l}s_{j_l}^T\hat{B}_k^{(l)}(\mu)}{s_{j_l}^T\hat{B}_k^{(l)}(\mu)s_{j_l}} + \frac{y_{j_l}\hat{y}_{j_l}^T}{\hat{y}_{j_l}^T s_{j_l}}, \quad l = 0, \cdots, \tilde{m}_k - 1 \end{aligned} \tag{3.4.5}$$

where $\tilde{m}_k = \min\{k+1, m\}$ and $j_l = k - \tilde{m} + l$. It is worth noting that these formulations are used in [17, 78].

Our attention now shifts to the trace and determinant of $\hat{B}_k(\mu)$. Initially, we establish that the trace of $\hat{B}_k^{(l)}(\mu)$ is of the order $O(\mu)$.

Lemma 3.4.1. *[106, Lemma 3.1]Suppose that Assumption 3.4.1 holds. Then,*

$$\operatorname{tr}(\hat{B}_k^{(l)}(\mu)) \leq M_3 + (2m+n)\mu, \quad l = 0, \cdots, \tilde{m}_k$$

where $M_3 = \frac{n}{\underline{\gamma}} + mM_2$.

Proof. The proof of this Lemma can be found in [112, Lemma 31]. ■

The next lemma provides a lower bound the determinant of $\hat{B}_k(\mu)$. The following lemma corresponds to [106, Lemma 3.2]. Since the proof of [106, Lemma 3.2] has a flaw, we give a complete proof here.

Lemma 3.4.2. *Suppose that Assumption 3.4.1 holds. Then,*

$$\det(\hat{B}_k(\mu)) \geq M_4\mu^n,$$

where

$$M_4 = \min\left\{\left(\frac{M_1}{M_3 + (2m+n)}\right)^{\tilde{m}}, \left(\frac{1}{M_3 + (2m+n)}\right)^{\tilde{m}}\right\}.$$

Proof. It is worth noting that the determinant of the updated approximate matrix using the BFGS updating scheme possesses the following property, as established in [91, 96]:

$$\det(\hat{B}_k^{(l+1)}(\mu)) = \det(\hat{B}_k^{(l)}(\mu))\frac{s_{j_l}^T\hat{y}_{j_l}(\mu)}{s_{j_l}^T\hat{B}_k^{(l)}(\mu)s_{j_l}}.$$

Then, we have

$$\begin{aligned}
\det(\hat{B}_k(\mu)) &= \det(\hat{B}_k^{(0)}(\mu)) \prod_{l=0}^{\tilde{m}-1} \frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{s_{j_l}^T \hat{B}_k^{(l)}(\mu) s_{j_l}} \\
&= \det(\hat{B}_k^{(l)}(\mu)) \prod_{l=0}^{\tilde{m}-1} \frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{s_{j_l}^T s_{j_l}} \frac{s_{j_l}^T s_{j_l}}{s_{j_l}^T \hat{B}_k^{(l)}(\mu) s_{j_l}} \\
&\geq \det(\hat{B}_k^{(0)}(\mu)) \prod_{l=0}^{\tilde{m}-1} \frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{s_{j_l}^T s_{j_l}} \frac{s_{j_l}^T s_{j_l}}{\lambda_{\max}(\hat{B}_k^{(l)}(\mu)) s_{j_l}^T s_{j_l}} \\
&= \det(\hat{B}_k^{(0)}(\mu)) \prod_{l=0}^{\tilde{m}-1} \frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{\|s_{j_l}\|^2} \frac{1}{\lambda_{\max}(\hat{B}_k^{(l)}(\mu))}.
\end{aligned}$$

Since $\hat{B}_k^{(0)}(\mu)$ is symmetric positive-definite, Lemma 3.4.1 conveys that $\lambda_{\max}(\hat{B}_k^{(l)}(\mu)) \leq M_3 + (2m+n)\mu$. Furthermore, we have $\frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{\|s_{j_l}\|^2} \geq M_1 + \mu$ from (3.4.3). Therefore, it follows that

$$\begin{aligned}
\det(\hat{B}_k(\mu)) &\geq \det(\hat{B}_k(\mu)^{(0)}) \left(\frac{M_1 + \mu}{M_3 + (2m+n)\mu} \right)^{\tilde{m}} \\
&= \det\left(\frac{1 + \gamma_k \mu}{\gamma_k} I \right) \left(\frac{M_1 + \mu}{M_3 + (2m+n)\mu} \right)^{\tilde{m}}.
\end{aligned}$$

We have

$$\frac{M_1 + \mu}{M_3 + (2m+n)\mu} = \frac{M_1}{M_3 + (2m+n)\mu} + \frac{\mu}{M_3 + (2m+n)\mu}. \tag{3.4.6}$$

If $\mu \leq 1$, then we have

$$\frac{M_1}{M_3 + (2m+n)\mu} \geq \frac{M_1}{M_3 + (2m+n)},$$

and if $\mu > 1$, then we get

$$\frac{\mu}{M_3 + (2m+n)\mu} = \frac{1}{M_3/\mu + (2m+n)} \geq \frac{1}{M_3 + (2m+n)}.$$

It then follows from (3.4.6) that

$$\left(\frac{M_1 + \mu}{M_3 + (2m+n)\mu} \right)^{\tilde{m}} \geq M_4,$$

where

$$M_4 = \min\left\{ \left(\frac{\mathrm{M}_1}{\mathrm{M}_3 + (2\mathrm{m} + \mathrm{n})} \right)^{\tilde{\mathrm{m}}}, \left(\frac{1}{\mathrm{M}_3 + (2\mathrm{m} + \mathrm{n})} \right)^{\tilde{\mathrm{m}}} \right\}.$$

Consequently, we set

$$\begin{aligned}\det(\hat{B}_k(\mu)) &\geq \left(\frac{1}{\gamma_k}+\mu\right)^n M_4\\ &\geq M_4\mu^n.\end{aligned}$$

This completes the proof. ■

From the previous two lemmas, we get $\lambda_{\max}(\hat{H}_k(\mu)) \to 0$ as $\mu \to \infty$. The next lemma corresponds to [106, Lemma 3.3]. Since the proof given in [106, Lemma 3.3] has a flaw, we give a complete proof here.

Lemma 3.4.3. *Suppose that Assumption 3.4.1 holds. Then, for all $k \geq 0$,*

$$\lambda_{\max}(\hat{H}_k(\mu)) \leq M_5\frac{1}{\mu},\ \forall \mu \in [\mu_{\min}, \infty),$$

where

$$M_5 = \frac{1}{M_4}\left(\frac{M_3}{\mu_{\min}}+(2m+n)\right)^{n-1}.$$

Furthermore, $\lim_{\mu\to\infty}\lambda_{\max}(\hat{H}_k(\mu)) = 0$.

Proof. We have from Lemmas 3.4.1 and 3.4.2 that

$$\begin{aligned}\operatorname{tr}(\hat{B}_k(\mu)) &\leq M_3 + (2m+n)\mu,\\ \det(\hat{B}_k(\mu)) &\geq M_4\mu^n.\end{aligned}$$

Since $\hat{B}_k(\mu)$ is symmetric positive-definite, we have

$$\begin{aligned}\operatorname{tr}(\hat{B}_k(\mu)) &\geq \lambda_{\min}(\hat{B}_k(\mu))\\ \det(\hat{B}_k(\mu)) &\leq \lambda_{\min}(\hat{B}_k(\mu))\{\lambda_{\max}(\hat{B}_k(\mu))\}^{n-1}.\end{aligned}$$

Therefore, we have

$$\begin{aligned}\lambda_{\min}(\hat{B}_k(\mu)) &\geq \frac{\det(\hat{B}_k(\mu))}{\{\lambda_{\max}(\hat{B}_k(\mu))\}^{n-1}}\\ &\geq \frac{M_4\mu^n}{\{(M_3+(2m+n)\mu)\}^{n-1}}.\end{aligned}$$

It then follows from Assumption 3.4.1 (v) that

$$\lambda_{\max}(\hat{H}_k(\mu)) = \frac{1}{\lambda_{\min}(\hat{H}_k^{-1}(\mu))}$$

$$
\begin{aligned}
&= \frac{1}{\lambda_{\min}(\hat{B}_k(\mu))} \\
&\leq \frac{\{(M_3 + (2m+n)\mu)\}^{n-1}}{M_4 \mu^n} \\
&= \frac{1}{M_4}\left(\frac{M_3 + (2m+n)\mu}{\mu}\right)^{n-1}\frac{1}{\mu}. \qquad (3.4.7)
\end{aligned}
$$

Since $\mu \geq \mu_{\min}$, we have

$$
\frac{M_3 + (2m+n)\mu}{\mu} = \frac{M_3}{\mu} + (2m+n) \leq \frac{M_3}{\mu_{\min}} + (2m+n).
$$

It then follows from (3.4.7) that

$$
\begin{aligned}
\lambda_{\max}(\hat{H}_k(\mu)) &\leq \frac{1}{M_4}\left(\frac{M_3}{\mu_{\min}} + (2m+n)\right)^{n-1}\frac{1}{\mu} \\
&= M_5\frac{1}{\mu}.
\end{aligned}
$$

Hence, we have

$$
\lim_{\mu\to\infty} \lambda_{\max}(\hat{H}_k(\mu)) = 0.
$$

This completes the proof. ■

Now, we give an upper bound for $\|d_k(\mu)\|$.

Lemma 3.4.4. *[106, Lemma 3.4]Suppose that Assumption 3.4.1 holds. Then,*

$$
\|d_k(\mu)\| \leq U_d,
$$

where

$$
U_d = \frac{L_f M_5}{\mu_{\min}}.
$$

Proof. The proof of this theorem can be found in [112, Lemma 34]. ■

Lemma 3.4.4 indicates that

$$
x_k + \nu d_k(\mu) \in \Omega + \mathrm{B}(0, U_d), \quad \forall \nu \in [0,1], \quad \forall \mu \in [\mu_{\min}, \infty), \quad \forall k \geq 0.
$$

Furthermore, considering the compactness of $\Omega + \mathrm{B}(0, U_d)$ and the twice-continuity differentiability of f, the gradient $\nabla f(x_k)$ is Lipschitz continuous on $\Omega + \mathrm{B}(0, U_d)$. In other words, there exists a positive constant L_g such that

$$\|\nabla^2 f(x_k)\| \le L_g \quad \forall x_k \in \Omega + \mathrm{B}(0, U_d). \tag{3.4.8}$$

Next, we advocate the behaviors of μ that satisfy the termination criteria $r_k(d_k(\mu),\, \mu) \ge \eta_1$ in the inner iterations of Step 2-2 in Algorithm 1.

Lemma 3.4.5. *[106, Lemma 3.5]Suppose that Assumption 3.4.1 holds. Then, we have*

$$\begin{aligned} &f(x_k) - f(x_k + d_k(\mu)) - \eta_1(f(x_k) - q_k(d_k(\mu), \mu)) \\ &\qquad\qquad\qquad \ge \frac{1}{2}((2-\eta_1)\lambda_{\min}(\hat{H}_k(\mu)^{-1}) - L_g)\|d_k(\mu)\|^2. \end{aligned}$$

Proof. The proof of this Lemma can be found in [112, Lemma 35]. ■

From Lemma 3.4.5, if μ satisfies

$$\lambda_{\min}(\hat{H}_k^{-1}(\mu)) \ge \frac{L_g}{2-\eta_1}, \tag{3.4.9}$$

then we have

$$r_k(d_k(\mu), \mu) \ge \eta_1, \tag{3.4.10}$$

that is, the inner loops of Algorithm 1 must terminate.

Next, we provides an upper bound of the parameter μ_k.

Lemma 3.4.6. *[106, Lemma 3.6]Suppose that Assumption 3.4.1 holds. Then, for any $k \ge 0$,*

$$\mu_k^* \le U_\mu,$$

where

$$U_\mu = \sigma_2 M_5 \frac{L_g}{2-\eta_1}.$$

Proof. The proof of this theorem can be found in [112, Lemma 36]. Note that we have used σ_2 [112] instead of γ_2 [106]. ■

Next, we give a lower bound for the reduction in the model function q_k.

Lemma 3.4.7. *[106, Lemma 3.7]Suppose that Assumption 3.4.1 holds. Then, we have*

$$f(x_k) - q_k(d_k(\mu), \mu) \ \ge\ M_6\|\nabla f(x_k)\|^2,$$

where

$$M_6 = \frac{1}{2(M_3 + (2m+n)\mu_{\min})}.$$

Proof. The proof of this Lemma can be found in [112, Lemma 37]. ∎

This lemma provides a lower bound for the reduction in the objective function value when x_k is not a stationary point.

Lemma 3.4.8. *[106, Lemma 3.8] Suppose that Assumption 3.4.1 holds. If there exists a positive constant ϵ_g such that $\|\nabla f(x_k)\| \geq \epsilon_g$, then we have $f(x_k) - f(x_{k+1}) \geq \rho \epsilon_g^2$, where $\rho = \eta_1 M_6$.*

Proof. The proof of this Lemma can be found in [112, Lemma 38]. ∎

Next, we prove the main theorem of this section.

Theorem 3.4.1. *[106, Theorem 3.1]Suppose that Assumption 3.4.1 holds. Then, $\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0$ or there exists $K \geq 0$ such that $\|\nabla f(x_K)\| = 0$.*

Proof. The proof of this theorem can be found in [112, Theorem 31]. ∎

It is important to note that since $f(x_k + d_k(\mu)) > f(x_k + (1+\alpha_k)d_k(\mu))$ from (3.3.2) and (3.3.5), Algorithm 3 also has the global convergence property.

3.5 Implementation issues

Based, the trust-region ratio of the regularized L-BFGS does not demonstrate satisfactory improvement, causing notable rise in the regularized parameter μ for certain large-scale test problems. Both scenarios culminate in a limited step, demanding a considerable number of iterations to converge on a solution. To address this challenge, we apply two techniques similar to RL-BFGS, accompanied by a discussion on how to appropriately set γ_k in the initial matrix $\hat{H}_k^{(0)}$ [112, Section 4.3] and nonmonotone decreasing technique [112, Section 4.2].

3.5.1 Nonmonotone decreasing technique

In Algorithm 1, we govern the regularized parameter μ to ensure the descent condition $f(x_{k+1}) < f(x_k)$. However, μ can occasionally become excessively large, especially for certain ill-posed problems, leading to a need for an extensive number of function evaluations. To address this challenge, we introduce the concept of a nonmonotone line search technique [55, 107]. This modification involves replacing the ratio function $r_k(d_k(\mu), \mu)$ with the following new ratio function $\bar{r}_k(d_k(\mu), \mu)$:

$$\bar{r}_k(d_k(\mu), \mu) = \frac{\max_{0\leq j\leq m(k)} f(x_{k-j}) - f(x_k + d_k(\mu))}{f(x_k) - \ q_k(d_k(\mu), \mu)},$$

where

$$m(0) = 0, \quad 0 \leq m(k) \leq \min\{m(k-1)+1, M\},$$

and M is a nonnegative integer constant. This modification retains the global convergence of the regularized L-BFGS method.

In the upcoming numerical experiments detailed in the subsequent section, we employ the original ratio function $r_k(d_k(\mu), \mu)$ for $k < M$. However, for $k \geq M$, we switch to utilizing the new ratio function $\bar{r}_k(d_k(\mu), \mu)$.

3.5.2 Scaling initial matrix

The regularized L-BFGS method utilizes the initial matrix in each iteration, as given below:

$$\hat{H}_k^{(0)}(\mu) = \frac{\gamma_k}{1+\gamma_k \mu} I.$$

The parameter γ_k indicates the scale of $\nabla^2 f(x)$. Thus, we use the scaling parameter γ_k used in [6, 15, 87, 97, 105], that is, we set

$$\gamma_k = \frac{s_{k-1}^T y_{k-1}}{\|y_{k-1}\|^2}.$$

It is known that the L-BFGS method with such scaling in the initial matrix has significant performance [15, 87]. Note that it demands $\gamma_k > 0$ to ensure the positive-definiteness of $\hat{H}_k^{(0)}(\mu)$. If $s_{k-1}^T y_{k-1} < \alpha \|s_{k-1}\|^2$, then we set $\gamma_k = \alpha \frac{\|s_{k-1}\|^2}{\|y_{k-1}\|^2}$, where α is a small positive constant.

3.6 Numerical results

In this section, we compare the L-BFGS, the regularized L-BFGS (RL-BFGS), and the regularized L-BFGS with line search (RL-BFGS-SW). We also compare our method with the existing regularized L-BFGS [113]. For the regularized ones, we adopt the nonmonotone techniques and the initial matrix discussed in Section 4. We have used the MCSRCH (line search routine) and parameters of the original L-BFGS [88] to find a step length in the RL-BFGS-SW.

We have solved 297 problems chosen from CUTEst [51]. All algorithms were coded in MATLAB 2018a. We have used an Intel Core i5 1.8GHz CPU with 8 GB of RAM on Mac OS X. We have chosen an initial point x_0 given in CUTEst.

We set the termination criteria as,

$$\frac{\|\nabla f(x_k)\|}{\max(1, \|x_k\|)} < 10^{-5} \text{ or } n_f > 50000 \text{ or } k > 50000 \text{ or } \mu_k > 10^{15}, \tag{3.6.1}$$

where n_f is the number of function evaluations. We regard the trails as failing when $n_f > 50000$, the number of iterations exceeds 50000, or $\mu_k > 10^{15}$.

We compare the algorithms with the distribution function proposed in [38]. Let $\mathcal{S}$ be a set of solvers, and let $\mathcal{P}_\mathcal{S}$ be a set of problems that can be solved by all algorithms in $\mathcal{S}$. We measure the required evaluations to solve problem p by solver $s \in \mathcal{S}$ as $t_{p,s}$, and the best $t_{p,s}$ for each p as t_p^*, which means $t_p^* = \min\{t_{p,s} | s \in \mathcal{S}\}$. The distribution function $F_s^\mathcal{S}(\tau)$, for a method s is defined by,

$$F_s^\mathcal{S}(\tau) = \frac{|\{p \in \mathcal{P}_\mathcal{S} | t_{p,s} \leq \tau t_p^*\}|}{|\mathcal{P}_\mathcal{S}|}, \ \tau \geq 1. \tag{3.6.2}$$

The algorithm whose $F_s^\mathcal{S}(\tau)$ is close to 1 is considered to be superior compared to other algorithms in $\mathcal{S}$.

3.6.1 Numerical behavior for some parameters in RL-BFGS

Since the RL-BFGS uses several parameters, we need to investigate the effect of these parameters so that we can choose the optimal ones. First, we consider σ_1 and σ_2 that control regularized parameters. We perform numerical experiments with 9 different sets of (σ_1, σ_2) in Table 3.2. The remaining parameters are set to

$$\eta_1 = 0.01, \eta_2 = 0.9, \mu_{min} = 1.0 \times 10^{-3}, m = 7, M = 10.$$

Table 3.2 shows the number of successes and rate of success for all 297 problems. Figure 3.1 shows the distribution function of these parameter sets in terms of CPU time.

Table 3.2: The number of success and rate of success at each (σ_1, σ_2).

P	σ_1	σ_2	Number of successes	Success rate (%)
P_1	0.1	2.0	254	85.52
P_2	0.1	5.0	259	87.2
P_3	0.1	10.0	265	89.22
P_4	0.2	2.0	254	85.52
P_5	0.2	5.0	255	85.85
P_6	0.2	10.0	257	86.53
P_7	0.5	2.0	249	83.83
P_8	0.5	5.0	253	85.18
P_9	0.5	10.0	254	85.52

From Table 3.2 and Figure 3.1 it is clear that $(\sigma_1, \sigma_2) = (0.1, 10.0)$ is the best. Therefore, we set $\sigma_1 = 0.1$ and $\sigma_2 = 10.0$ for all further experiments.

Next, we compare the number m of vector pairs in the L-BFGS procedure. Note that the original L-BFGS usually choose it in $3 \leq m \leq 7$ [87]. Thus, we compare $m = 3, 5, 7$. The remaining parameters are set to

$$\sigma_1 = 0.1, \sigma_2 = 10, \eta_1 = 0.01, \eta_2 = 0.9, \mu_{min} = 1.0 \times 10^{-3}, M = 10.$$

Table 3.3: The number of success and rate of success at each m.

Memory	Number of successes	Success rate (%)
3	246	82.82
5	253	85.18
7	265	89.22

From Table 3.3 we see that $m = 7$ is the best, also Figure 3.2 shows that $m = 7$ is initially better in terms of CPU time. Therefore, we set $m = 7$ for further experiments. Finally, we compare the behavior of nonmonotone parameters M. We compare $M = 0, 4, 6, 8,\ 10, 12$. Note that $M = 0$ implies the usual monotone decreasing case. The remaining parameters are set to

$$\sigma_1 = 0.1, \sigma_2 = 10, \eta_1 = 0.01, \eta_2 = 0.9, \mu_{min} = 1.0 \times 10^{-3}, m = 7.$$

Figure 3.3 shows the distribution function of the nonmonotone parameter in terms of the CPU time.

From Table 3.4 and Figure 3.3 it is clear that $M = 10$ is better. Therefore, we use $M = 10$ in the next section.

Table 3.4: The number of success and rate of success at each M.

Nonmonotone	Number of successes	Success rate (%)
Monotone(M=0)	257	86.53
4	261	87.87
6	263	88.55
8	264	88.88
10	265	89.22
12	264	88.88

3.6.2 Comparisons of RL-BFGS-SW, RL-BFGS and L-BFGS method

We compare the RL-BFGS-SW, the RL-BFGS, and the L-BFGS methods in terms of the number of function evaluations and CPU time. For all numerical experiments, the parameters

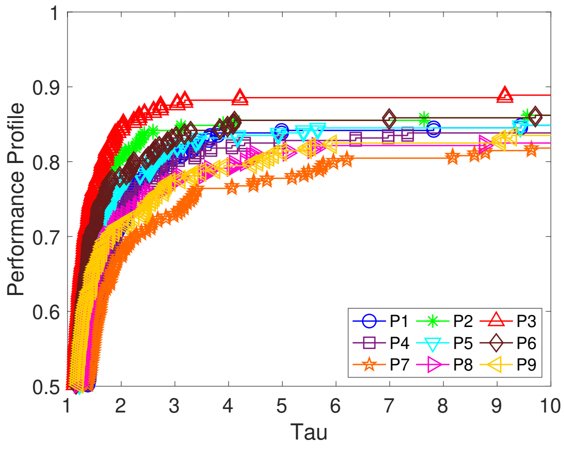


Figure 3.1: Comparison of (σ_1, σ_2).

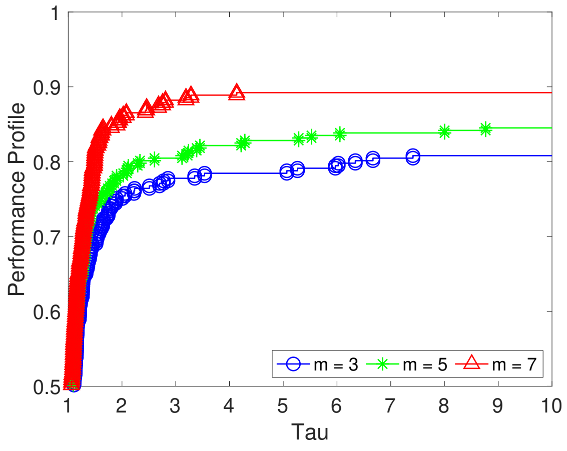


Figure 3.2: Comparison of $m = 3, 5, 7$.

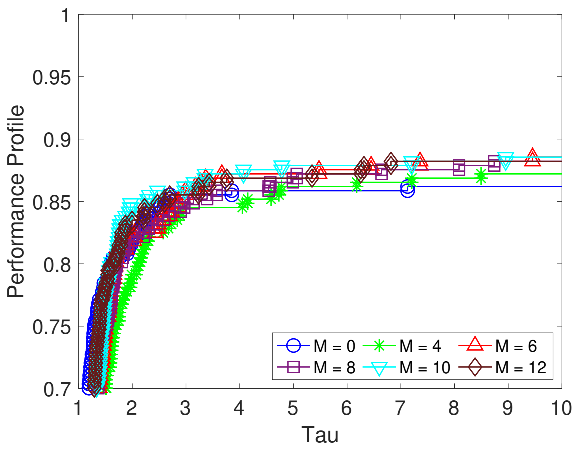


Figure 3.3: Comparison of M.

in RL-BFGS and RL-BFGS-SW are as follows:

$$\eta_1 = 0.01, \eta_2 = 0.9, \mu_{min} = 1.0 \times 10^{-3}, M = 10, m = 7, \sigma_1 = 0.1, \sigma_2 = 10.$$

Figures 3.4 and 3.5 shows the results of the number of successes and rate of successes for $\mathcal{P}_\mathcal{S}$ in terms of function evaluations and CPU time, respectively. Here $\mathcal{S}$ is the set of problems that are solved by all three algorithms. Figure 3.4 shows that L-BFGS can solve 83.5% of test problems while RL-BFGS and RL-BFGS-SW can solve 89.22% and 87.87% of problems, respectively. On the other hand, Figure 3.4 shows that L-BFGS is faster than the regularized ones for the solved problems in terms of the number of function evaluations for 75% of problems. However, RL-BFGS and RL-BFGS-SW can capture attention with the ability to solve 89.22% and 87.87% respectively, as the height of the profile performance shows for $\tau > 4$. Moreover, Figure 3.5 shows that all solvers take almost equal time to solve

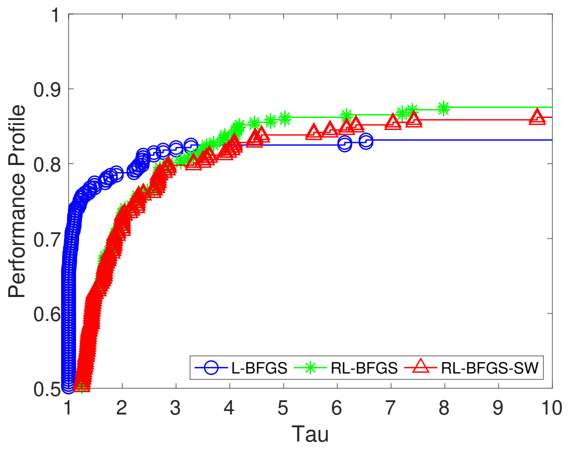


Figure 3.4: Comparison of n_f.

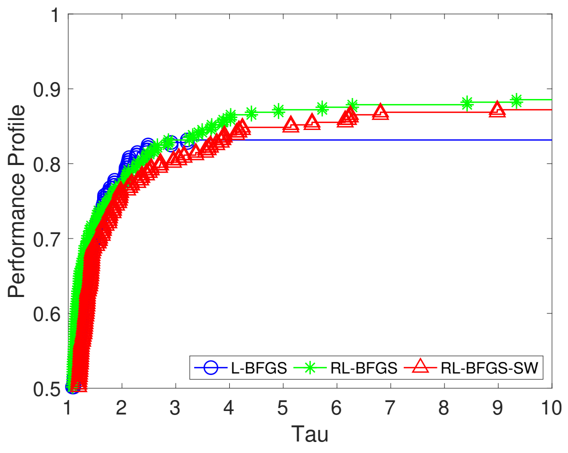


Figure 3.5: Comparison of CPU time.

83.5% of the problems.

The above numerical results indicate that the numerical behaviors of the RL-BFGS and the RL-BFGS-SW are almost the same. To see the differences, we present the numerical results, which compare the performance of each algorithm.

We first observed that the RL-BFGS-SW performs the line search for 65 problems and does not use it for the remaining problems. Then, we check the results for those 65 problems and compare them in terms of function evaluations. We found that RL-BFGS-SW requires fewer number function evaluations than RL-BFGS for 39 test problems, while RL-BFGS requires fewer number of function evaluations than RL-BFGS-SW for 26 test problems among 65 test problems. Now we compare our methods with the existing regularized L-BFGS

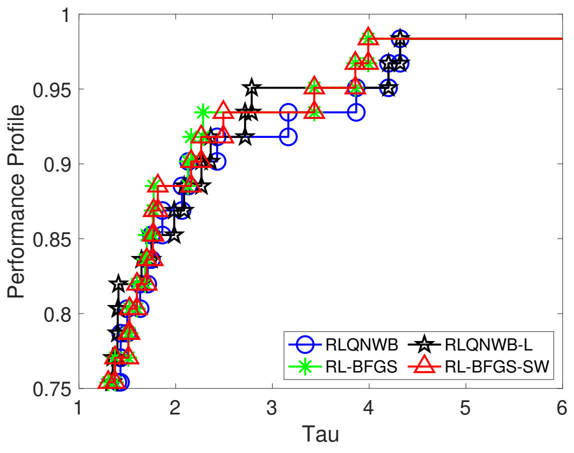


Figure 3.6: Comparison of n_f.

methods [113], (see Remark 3.2.1). Note that the chapter proposed several updated rules of β_k in (3.2.4). We chose two methods, RLQNWB and RLQNWB-L among them since they are

Table 3.5: Test functions [51]

Problem	Dim	Problem	Dim	Problem	Dim
ARGLINA	200	ARWHEAD	1000	BDQRTIC	1000
BIGGSB1	1000	BROWNAL	200	BROYDN7D	500
BRYBND	5000	CHAINWOO	10000	COSINE	10000
CRAGGLVY	100	DIXMAANA	9000	DIXMAANB	9000
DIXMAANC	9000	DIXMAAND	9000	DIXMAANE	9000
DIXMAANF	9000	DIXMAANG	3000	DIXMAANH	3000
DIXMAANJ	3000	DIXMAANK	3000	DIXMAANL	3000
DIXON3DQ	500	DQDRTIC	10000	DQRTIC	1000
ENGVAL1	1000	EDENSCH	1000	EG2	1000
CHNROSNB	50	FMINSRF2	5625	FMINSURF	1024
GENHUMPS	5000	GENROSE	1000	LIARWHD	10000
MANCINO	100	MODBEALE	50	MOREBV	5000
NONCVXU2	500	NONCVXUN	100	NONDIA	10000
NONDQUAR	2000	OSCIPATH	10000	MSQRTALS	529
MSQRTBLS	529	PENALTY1	200	PENALTY2	50
POWELLSG	5000	QUARTC	1000	SCHMVETT	1000
SINQUAD	200	SPARSQUR	10000	SPMSRTLS	4999
TOINTGOR	50	TOINTGSS	5000	TOINTPSP	50
TQUARTIC	5000	VAREIGVL	1000	WOODS	10000
TRIDIA	500	POWER	500	-	-

superior to the other update rules. To compare the proposed methods with the RLQNWB and RLQNWB-L, we used the numerical results provided by Peyghami and Tarzanagh [113] [3]. We solved 59 test problems in Table 3.5, and provided performance profiles in terms of function evaluations. We set the termination criteria as,

$$\|\nabla f(x_k)\| \leq 10^{-5} \text{ or } \mu_k > 10^{15},$$

or the number of iterations and function evaluations exceeds 10,000 and 50,000, respectively.

We first note that RL-BFGS and RL-BFGS-SW were unable to solve a problem CHAINWOO. Figure 3.6 shows that both RL-BFGS and RL-BFGS-SW take the fewer number of function evaluations and slightly outperform the RLQNWB and RLQNWB-L, respectively.

Remark 3.6.1. Quite recently, Steck and Kanzow [66] proposed a similar regularized L-BFGS method that computes $(B_k + \mu I)^{-1}$ explicitly. They reported that their method outperformed the proposed RL-BFGS without line search because of $(B_k + \mu I)^{-1}$.

[3] `http://wp.kntu.ac.ir/peyghami/pdf/MRLBFGS.rar`

3.7 Conclusions

In this chapter, we introduced the regularized L-BFGS method with the simultaneous use of Wolfe line search. We established global convergence under suitable assumptions and delved into various implementation considerations. The numerical experiments demonstrated that, in an overall comparison, the proposed method successfully addresses more problems than the original L-BFGS.

Acknowledgment

This chapter is based on Tankaria, H., Sugimoto, S. & Yamashita, N, "A regularized limited memory BFGS method for large-scale unconstrained optimization and its efficient implementations," *Computational Optimization and Applications*, **82**, 61–88 (2022). DOI: `https://doi.org/10.1007/s10589-022-00351-5`. Reproduced with permission from Springer Nature.

Chapter 4

A Stochastic Variance Reduced Gradient using Barzilai-Borwein Techniques as Second Order Information

In this chapter, our focus lies on elevating the Stochastic Variance Reduced Gradient (SVRG) method by incorporating the curvature information from the objective function. We suggest a variance reduction in stochastic gradients using the computationally efficient Barzilai-Borwein (BB) method, integrating it into the search direction of SVRG. Furthermore, we introduce a BB-step size variant. Our work establishes a linear convergence theorem, applicable not just to our proposed method but also to other existing SVRG variants incorporating second-order information.

One of the important tasks of supervised machine learning is to solve the empirical risk minimization problem [102]. In this chapter, we focus on empirical risk minimization defined as

$$\min_{w\in\mathbb{R}^d} F(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w; a_i, b_i), \tag{4.0.1}$$

where $f_i(w; a, b) : \mathbb{R}^d \to \mathbb{R}$ represents the composition of a loss function ℓ and a prediction function h parametrized by w, denoted as $f_i(w; a_i, b_i) = \ell(h(w; a_i); b_i)$.

For simplicity, we use $f_i(w)$ as shorthand for $f_i(w; a_i, b_i)$. For instance, in the context of least squares regression with n training samples $(a_i, b_i)_{i=1}^n$, where $a_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$, the loss function is expressed as $f_i(w) = (w^T a_i - b_i)^2$. Similarly, for ℓ_2-regularized logistic regression, the formulation is $f_i(w) = \log(1 + \exp(-b_i a_i^T w)) + \frac{\lambda}{2}\|w\|_2^2$, where λ is an ℓ_2-regularized parameter, and $a_i \in \mathbb{R}^d$, $b_i \in -1, 1$ correspond to samples in a binary classification problem.

Massive amount of data presents a prevalent challenge when minimizing the objective function F with a large n. We focus on minimizing the average of a twice continuously differentiable and strongly convex function, i.e., each $f_i(w)$ being both strongly convex and twice continuously differentiable. Large-scale machine learning problems pose computational challenges due to extensive training data. To address these challenges, efficient optimization techniques are imperative.

The classical full gradient method [22] computes the full gradient as the name indicates and uses iterations of the form as follows:

$$w_{k+1} = w_k - \eta_k \nabla F(w_k) = w_k - \frac{\eta_k}{n} \sum_{i=1}^{n} \nabla f_i(w_k), \tag{4.0.2}$$

where η_k is a stepsize. The convergence rate of the full gradient is sublinear for convex functions and linear for strongly convex functions. However, its computational cost is high, involving the computation of n gradients ∇f_i at each iteration. As an alternative, a widely adopted approach for solving large-scale optimization problems is using stochastic gradient methods, which employ either a single or a small subset (mini-batch) of training data in each iteration. The stochastic gradient descent (SGD) [98] chooses a single training sample $i \in \{1, 2, \ldots, n\}$ uniformly at random and updates w_k as follows:

$$w_{k+1} = w_k - \eta_k \nabla f_i(w_k), \tag{4.0.3}$$

For global convergence, SGD requires a diminishing step size η_k, leading to slow convergence due to the variance of the stochastic gradient. By reducing the variance of the stochastic gradient, we aim to enable a constant step size, potentially resulting in a faster convergence rate.

In the previous decade, numerous researchers have dedicated efforts to mitigate the variance of the stochastic gradient while maintaining a constant step size [64, 125, 99, 31]. Stochastic average gradient (SAG)[100, 99] introduced biased updates, achieving a linear convergence rate. Stochastic Average gradient accelerated (SAGA)[31] improved upon SAG by utilizing an unbiased estimate. Additionally, there has been notable exploration in the realm of dual minimization [103, 127]

The stochastic variance reduced gradient [64] (SVRG) method explicitly reduces the variance through an unbiased estimate of the gradient:

$$SVRG \qquad w_{k+1} = w_k - \eta_k v_k, \tag{4.0.4}$$

where $v_k = \nabla f_i(w_k) - \nabla f_i(\tilde{w}) + \nabla F(\tilde{w})$ and $\tilde{w}$ is the resultant vector at the end of the previous epoch. Johnsan and Zhang [64] showed that the variance of v_k is diminishing as w_k approaches a solution, and hence the sequence converges linearly with a constant step size.

However, the standard SVRG does not exploit the information on the Hessian of F.

Considerable effort has been dedicated to stochastic variance-reduced gradient methods incorporating second-order information. It is noteworthy that second-order information can be applied in two distinct ways. The conventional approach involves utilizing the second-order approximation of the objective function, as seen in methods like stochastic quasi-Newton [101, 16, 81] and SVRG-BB [110].

$$w_{k+1} = w_k - \eta_k B_k^{-1} g_i(w_k), \tag{4.0.5}$$

where $g_i(w_k)$ is any stochastic gradient that uses first-order information such that $\mathbb{E}[g_i(w_k)] = \nabla F(w_k)$ and B_k is an approximation of the Hessian $[\nabla^2 f(w)]$.

An alternative way to use second-order information is to further reduce the variance via the variance reduction technique of the stochastic gradient as

$$w_{k+1} = w_k - \eta_k(v_k + (\nabla^2 F(\tilde{w}) - \nabla^2 f_{i_t}(\tilde{w}))(w_k - \tilde{w})), \tag{4.0.6}$$

where $v_k = \nabla f_{i_t}(w_k) - \nabla f_{i_t}(\tilde{w}) + F(\tilde{w})$.

Gower et al.[53] introduced SVRG-2, a variant of SVRG leveraging second-order information to control the variance of stochastic gradient (4.0.6). This method computes the Hessian at a cost of $O(nd^2)$, and its theoretical analysis indicates superior variance reductions, achieving convergence in a small number of epochs. The authors also proposed various Hessian approximations, including low-rank and diagonal Hessians. It is important to note that, in scenarios with a small dimension d, using SVRG-2 [53] with the true Hessian may be feasible. However, SVRG-2 might become time-consuming when dealing with a large d. Additionally, the linear convergence of diagonal and low-rank approximations is not demonstrated in [53].

In order to address these issues, we propose the Barzilai-Borwein approximation [6] to further control the variance using SVRG.

Table 5.1 shows the clarification among the various variants of SVRG, which are first and second order methods with variance reduction through first and second order information.

Table 4.1: Various variance reduced methods

	First-order model	Second-order model
Variance reduction via first-order information	SVRG [64]	SVRG-BB [110]
Variance reduction via second-order information	SVRG-2 and SVRG-2D [53], SVRG-2BB (Proposed)	SVRG-2BBS (Proposed)

4.1 The existing methods: SVRG and SVRG-2

In this section we introduce the SVRG [64] and the SVRG-2 [53].

Generally, the stochastic gradient methods attempt to estimate the true gradient accurately. Since the expectation of the stochastic gradient $\nabla f_i(w)$ is the true gradient $\nabla F(w)$, i.e., $\mathbb{E}[\nabla f_i(w)] = \frac{1}{n}\sum_{i=1}^n \nabla f_i(w) = \nabla F(w), i \in \{1, \ldots, n\}$. Thus, in order to get the best estimate of the stochastic gradient, it is natural to apply variance reduction techniques.

First we give a common framework of SVRG and SVRG-2.

Algorithm 4

```
Parameters: update frequency $m$ and step size $\eta$, $\tilde{g}$ is gradient of $F$ at $\tilde{w}$
Initialize $\tilde{w}_0$
1. for $k = 1, 2, \ldots$
2.   $\tilde{w} = \tilde{w}_{k-1}$
3.   $\tilde{g} = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{w})$
4.   $\tilde{A}$ is given
5.   $w_0 = \tilde{w}$
6.     for $t = 1, \ldots, m$
7.          Randomly pick $i_t \in \{1, \ldots, n\}$
8.          $\tilde{A}_{i_t}$ such that $\mathbb{E}[\tilde{A}_{i_t}] = \tilde{A}$ is given
9.          $w_t = w_{t-1} - \eta\ (\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \tilde{g} + (\tilde{A} - \tilde{A}_{i_t})(w_{t-1} - \tilde{w}))$
10.    end for
11. option 1 $\tilde{w}_k = w_m$
    option 2 $\tilde{w}_k = w_t$ for randomly chosen $t \in \{0, \ldots, m-1\}$
12. end for
Output $\tilde{w}_k = w_m$
```

The search direction of Algorithm 1 uses $\nabla f_{i_t}(\tilde{w})$ at $\tilde{w}$ for a current epoch k, and $\tilde{w}$ does not update till the next epoch k. If $\tilde{w}$ does not change much, then it leads to slow convergence. Therefore it is important to use the matrices $\tilde{A}$ and A_{i_t} which is computationally affordable and satisfies following properties:

1. Unbiased estimate of $\tilde{A}_{i_t}$, that is
$$\mathbb{E}[\tilde{A}_{i_t}] = \tilde{A}.$$

2. Approximation of $\tilde{A}_{i_t}$ and $\tilde{A}$ such that,
$$\tilde{A}_{i_t} \approx \nabla^2 f_{i_t}(\tilde{w}) \text{ and } \tilde{A} \approx \nabla^2 F(\tilde{w}).$$

From property 1, we have $\mathbb{E}(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \tilde{g} + (\tilde{A} - \tilde{A}_{i_t})(w_{t-1} - \tilde{w})) = \nabla F(w_{t-1})$, and hence we can expect that Algorithm 1 finds a solution. The property 2 is to reduce the variance of w_t.

Algorithm 1 includes the well-known SVRG and its variants as follows:

$$\textbf{SVRG [64]}: \quad \tilde{A} = \tilde{A}_{i_t} = 0, \tag{4.1.1}$$

$$\textbf{SVRG-2 [53]}: \quad \tilde{A} = \nabla^2 f(\tilde{w}), \tilde{A}_{i_t} = \nabla^2 f_{i_t}(\tilde{w}) \text{ with option 1}, \tag{4.1.2}$$

$$\textbf{SVRG-2D [53]}: \tilde{A} = \text{diagonal}(\nabla^2 f(\tilde{w})), \tilde{A}_{i_t} = \text{diagonal}(\nabla^2 f_{i_t}(\tilde{w})) \text{ with option 1}, \tag{4.1.3}$$

where a diagonal(G) denotes a diagonal matrix whose diagonal entries are G, that is

$$G_{i,j} = \begin{cases} g_{i,j} & \text{if the } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{4.1.4}$$

The SVRG proposed [64] to computes a variance-reduced gradient where the gradient of F at $\tilde{w}$, i.e., $\tilde{g}$ does not update till the end of the epoch. Hence, the reduction of the variance is not much, which causes slow convergence. Moreover, it needs to use a smaller step size to maintain linear convergence.

The SVRG-2 proposed by Gower et. al. [53] exploits the difference of Hessian $(\nabla^2 F(\tilde{w}) - \nabla^2 F_i(\tilde{w}))$ by multiplying with $(w_t - \tilde{w})$, which uses the current parameter w_t and updates at every iteration. In practice, when the Hessian is sparse or d is small, calculating $\nabla^2 F(\tilde{w})$ and computing $\nabla^2 F(\tilde{w})(w_{t-1} - \tilde{w})$ is computationally tractable. However, when d is large, it is not a good idea to use original Hessian since it costs $O(nd^2)$.

The SVRG-2D proposed by Gower et. al. [53] computes the diagonal Hessian, which does not perform well as seen in section 5.

4.2 SVRG-2 with the Barzilai-Borwein approximation

In this section, we first introduce the standard BB-method [6], then we propose to use the BB-method as $\tilde{A}$ and $\tilde{A}_{i_t}$ in Algorithm 4 to further reduce the variance. Finally, we propose to use BB-step size along with the proposed BB variant of SVRG-2.

4.2.1 Barzilai-Borwein method

In this section we recall the Barzilai and Borwein method [6] for solving unconstrained optimization deterministically.

Barzilai and Borwein [6] proposed a two-point step size gradient method [6] which is also known as the BB-method. This method is popular to solve unconstrained optimization problems which is motivated from the quasi-Newton methods.

Consider the unconstrained optimization problem:

$$\min_{w} f(w),$$

where f is a differentiable function.

The quasi-Newton method [89] solve minimization problem by using the approximate Hessian which needs to satisfy the secant equation. The quasi-Newton method takes the iteration of the form:

$$w_{k+1} = w_k - \eta_k B_k^{-1} \nabla f(w_k),$$

where B_k is an approximation of the Hessian matrix of f at the w_k. The approximate Hessian B_k needs to satisfy the following secant equation [89]:

$$B_k s^k = y^k,$$

where $s^k = w_k - w_{k-1}$ and $y^k = \nabla f(w_k) - \nabla f(w_{k-1})$ for $k \geq 1$. The well-known BFGS method requires $O(d^2)$ time and space complexities for computing B_k.

In practice when d is large, it is time consuming to compute the B_k. The BB-method restricts B_k as $B_k = \frac{1}{\alpha_k}\mathrm{I}$, with $\alpha > 0$ and gets α_k by the secant equation in the least square sense, i.e.,

$$\min_{\alpha} \left\| \frac{1}{\alpha} s^k - y^k \right\|^2, \tag{4.2.1}$$

Note that when $(s^k)^\top y^k > 0$ the solution of (4.2.1),

$$\alpha_k = \frac{\|s^k\|^2}{(s^k)^T y^k}.$$

Then it is clear that,

$$B_k = \frac{(s^k)^T y^k}{\|s^k\|^2}\mathrm{I}. \tag{4.2.2}$$

An alternative way to get α_k can be given as follows.

$$\min_{\alpha} \left\| s^k - \alpha y^k \right\|^2,$$

and the solution is given by,

$$\alpha_k = \frac{(s^k)^T y^k}{\|y^k\|^2}, \quad B_k = \frac{\|y^k\|^2}{(s^k)^T y^k}\mathrm{I}. \tag{4.2.3}$$

Remark 4.2.1. *Note that when f is strongly convex, we have $(s^k)^\top y^k > 0$.*

4.2.2 SVRG-2 with the Barzilai-Borwein method (SVRG-2BB)

In this subsection, we propose Algorithm 1 with the BB-methods (4.2.2). The previous subsection introduced two different approximations of B_k of (4.2.2) and (4.2.3). We adopt the (4.2.2) for the rest of the chapter because (4.2.2) is convenient for convergence analysis.

Now we propose to use the BB-method for a Hessian approximation along with Algorithm 1. We call it the SVRG-2BB method.

SVRG-2BB: Algorithm 1 with (4.2.2) for the approximate Hessian of $f_i(w)$, that is

$$\tilde{A}^k = \frac{(s^k)^T y^k}{\|s^k\|^2}\mathrm{I} = \frac{1}{n}\sum_{i=1}^{n} \frac{(s^k)^T(\nabla f_i(\tilde{w}_k) - \nabla f_i(\tilde{w}_{k-1}))}{\|s^k\|^2}, \tag{4.2.4}$$

$$\tilde{A}^k_{i_t} = \frac{(s^k)^T(\nabla f_{i_t}(\tilde{w}_k) - \nabla f_{i_t}(\tilde{w}_{m-1}))}{\|s^k\|^2}, \tag{4.2.5}$$

where $s^k = \tilde{w}_k - \tilde{w}_{k-1}$ and $y^k = \nabla F(\tilde{w}_k) - \nabla F(\tilde{w}_{k-1})$. Hence, SVRG-2BB is Algorithm 4 with (4.2.4) in step 4 and (4.2.5) in step 8.

It is easy to see that $\mathbb{E}[\tilde{A}^k_{i_t}] = \tilde{A}^k$ for the above approximations derived from the BB-method. Since the approximate Hessian of BB-method is $\alpha_k \mathrm{I}$, we can treated $\tilde{A}^k$ and $\tilde{A}^k_{i_t}$ as scalars. Then, it is clear that $(\tilde{A} - \tilde{A}_{i_t})(w_{t-1} - \tilde{w})$ is the same direction of the $(w_{t-1} - \tilde{w})$, and v_t given as

$$v_t = \nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \tilde{g} - \tilde{A}^{k-1}_{i_t}(w_{t-1} - \tilde{w}) + \tilde{A}^{k-1}(w_{t-1} - \tilde{w}), \tag{4.2.6}$$

where computational costs is $O(n)$ when $\nabla f_{i_t}(w)$ is provided.

Remark 4.2.2. *It is important to note that $(s^k)^T y^k = (w_k - w_{k-1})^T(\nabla f(w_k) - \nabla f(w_{k-1})) > 0$ is not necessarily true when the objective function is not convex. In order to handle this issue, we give a simple remedy to use SVRG-2BB as follows:*

$$\bar{A}^k = \max\left\{\frac{(s^k)^T(y^k)}{\|s^k\|^2}, \delta\right\},$$

where $\delta > 0$.

4.2.3 SVRG-2BB with a Barzilai-Borwein step size (SVRG-2BBS)

In this subsection, we propose SVRG-2BB with the BB-step size. The typical acceleration technique of the SVRG is to use a quasi-Newton method as the quadratic model of the objective function and generate an iterate as

$$w_{k+1} = w_k - \eta_k B_k^{-1}(\nabla f_i(w_k) - \nabla f_i(\tilde{w}) + \nabla F(\tilde{w})), \tag{4.2.7}$$

where B_k is either Hessian or approximate Hessian of the objective function (4.0.1) at previous epoch. One of such methods is SVRG-BB [110] which employs the second-order via the information from BB-method and $\eta_k = 1/m$. SVRG-BB updates B_k^{-1} as follows,

$$B_k^{-1} := \frac{\|s_k\|^2}{(s_k^T y_k)} \mathrm{I},$$

where $s^k = \tilde{w}_{k-1} - \tilde{w}_{k-2}$, $y^k = \nabla F(\tilde{w}_{k-1}) - \nabla F(\tilde{w}_{k-2})$ and m is the length of an epoch. Note that SVRG-BB is regarded as Algorithm 4 with

$$\eta_k = \frac{B_k^{-1}}{m} = \frac{1}{m} \cdot \frac{\|s_k\|^2}{(s_k^T y_k)}.$$

Similarly, we exploit the idea of using BB-step size with SVRG-2BB so that we can make use of the gradient information and parameters to further reduce the variance, i.e., to make the most use of gradient information from the BB-method. We call it SVRG-2BBS.

SVRG-2BBS: SVRG-2BB with variable stepsize η_k^t defined by

$$\eta_k^t = \frac{\xi_t}{m_1} \frac{\|\tilde{w}_{k-1} - \tilde{w}_{k-2}\|^2}{((\tilde{w}_{k-1} - \tilde{w}_{k-2})^T(\tilde{g}_{k-1} - \tilde{g}_{k-2}))}, \tag{4.2.8}$$

where $\xi_t > 0$.

Note that SVRG-BB [110] computes the BB-stepsize per epoch. Whereas in SVRG-2BBS, we control and update the BB-step size by multiplying an appropriate constant ξ_t. Note also that the denominator of the BB-step size in (4.2.8) usually becomes very large since the number of samples $m = 2n$ is large. To overcome the difficulty, we modify η_k by multiplying ξ_t and by replacing constant m with m_1.

Remark 4.2.3. *It is noteworthy to mention that the update frequency in the inner **for** loop of SVRG-2BBS is $m = 2n$ (i.e., similar to SVRG). We replaced m with m_1 only in the stepsize η_k^t of SVRG-2BBS and not in the update frequency.*

4.3 Convergence analysis

In this section, we show the linear convergence of Algorithm 4 when F is strongly convex.

We first analyze the difference in the variances of the SVRG and Algorithm 4. We make the following assumptions.

Assumption 4.3.1. *Each f_i is μ_i-strongly convex, i.e., there exists $\mu_i > 0$ such that*

$$f_i(w) \geq f_i(z) + \nabla f_i(z)^T(w - z) + \frac{\mu_i}{2}\|z - w\|^2 \ \forall w, z \in \mathbb{R}^d.$$

Moreover the gradient of each f_i is L-Lipschitz continuous, *i.e.*,

$$\|\nabla f_i(w) - \nabla f_i(z)\| \leq L\|w - z\| \ \forall w, z \in \mathbb{R}^d.$$

Under this assumption, it is clear that $\nabla F(w)$ is also L-Lipschitz continuous:

$$\|\nabla F(w) - \nabla F(z)\| \leq L\|w - z\| \ \forall w, z \in \mathbb{R}^d. \tag{4.3.1}$$

Next lemmas are useful for our analysis. We omit its proof since it is directly follows from (4.3.1).

Lemma 4.3.1. *If f_i is L-Lipschitz continuous, then*

$$(w - z)^T(\nabla f_i(w) - \nabla f_i(z)) \leq L\|w - z\|^2,$$

and hence $s^T y \leq L\|s\|^2$ for $s = w - z$ and $y = \nabla f_i(w) - \nabla f_i(z)$.

The next lemma gives the result of the Lipschitz continuous Hessian.

Lemma 4.3.2. [83] *If each $f_i : \mathbb{R}^d \to \mathbb{R}$ be twice continuous differentiable with $\tilde{L}_t$-Lipschitz continuous Hessian, then for any $w, y \in \mathbb{R}^d$ we have,*

$$\|\nabla f_{i_t}(w) - \nabla f_{i_t}(z) - \nabla^2 f_{i_t}(w)(w - z)\| \leq \frac{\tilde{L}_t}{2}\|w - z\|^2.$$

Next two Lemmas give results from the strongly convexity.

Lemma 4.3.3. [83] *If each f_i is μ-strongly convex, then for any $w, y \in \mathbb{R}^d$ we have*

$$\|f_i(w) - f_i(z)\| \geq \mu_i\|w - z\|.$$

Lemma 4.3.4. [83] *If F is μ-strongly convex function, w_* is a minimum $F(w)$,*

$$F(w) - F(w_*) \geq \frac{\mu}{2}\|w - w_*\|^2$$

for all $w \in \mathbb{R}^d$.

Now we compare the variance of the SVRG-2 method with that of the original SVRG. Let $v_t^{svrg} = \nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \nabla F(\tilde{w})$ for the SVRG. Recall that $\mathbb{E}[v_t^{svrg}] = \nabla F(w_{t-1})$. Moreover we have

$$\begin{aligned}
\mathbb{E}[\|v_t^{svrg} - \nabla F(w_{t-1})\|^2] &= \mathbb{E}[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \nabla F(\tilde{w}) - \nabla F(w_{t-1})\|^2] \\
&= \mathbb{E}[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \mathbb{E}[\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})]\|^2] \\
&\le \mathbb{E}[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|^2] \\
&= O(\|w_{t-1} - \tilde{w}\|^2), \qquad (4.3.2)
\end{aligned}$$

where the inequality follows from the fact that $\mathbb{E}[\|\xi - \mathbb{E}[\xi]\|^2] = \mathbb{E}[\|\xi\|^2] - \|\mathbb{E}[\xi]\|^2 \le \mathbb{E}[\|\xi\|^2]$. Let $v_t^{2nd} = \nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \nabla F(\tilde{w}) + (\nabla^2 F(\tilde{w}) - \nabla^2 f_{i_t}(\tilde{w}))(w_{t-1} - \tilde{w})$ for the SVRG-2 method. We see that $\mathbb{E}[v_t^{2nd}] = \nabla F(w_{t-1})$. Moreover we have

$$\begin{aligned}
&\mathbb{E}[\|v_t^{2nd} - \nabla F(w_{t-1})\|^2] \\
&= \mathbb{E}[\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \nabla F(\tilde{w})) - \nabla^2 f_{i_t}(\tilde{w})(w_{t-1} - \tilde{w}) \\
&\qquad\qquad + \nabla^2 F(\tilde{w})(w_{t-1} - \tilde{w}) - \nabla F(w_{t-1})\|^2] \\
&= \mathbb{E}[\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \nabla^2 f_{i_t}(\tilde{w})(w_{t-1} - \tilde{w})) \\
&\qquad\qquad - \mathbb{E}[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \nabla^2 f_{i_t}(\tilde{w})(w_{t-1} - \tilde{w})\|]\|^2] \\
&\le \mathbb{E}[\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \nabla^2 f_{i_t}(\tilde{w})(w_{t-1} - \tilde{w}))\|^2] \\
&= O(\|w_{t-1} - \tilde{w}\|^4), \qquad (4.3.3)
\end{aligned}$$

where the first inequality follows from the fact that $\mathbb{E}[\|\xi - \mathbb{E}[\xi]\|^2] = \mathbb{E}[\|\xi\|^2] - \|\mathbb{E}[\xi]\|^2 \le \mathbb{E}[\|\xi\|^2]$, and the last inequality follows from Lemma 4.3.2 with $z = w_{t-1}$ and $w = \tilde{w}$.

The inequality (4.3.2) and (4.3.3) suggests that the variance of v_t^{2nd} is smaller than that of v_t^{svrg} when w_{t-1} is close to $\tilde{w}$. To reduce the computational cost, Algorithm 4 exploits $\tilde{A}^{k-1}$ and $\tilde{A}_{i_t}^{k-1}$ as $\nabla F(w_{k-1})$ and $\nabla F_{i_t}(w_{k-1})$. However, we do not have the variance bound (4.3.2) for arbitrary $\tilde{A}^{k-1}$ and $\tilde{A}_{i_t}^{k-1}$.

Thus we suppose the following assumption on $\tilde{A}^{k-1}$ to prove the linear convergence of Algorithm 4.

Assumption 4.3.2. *There exists a positive constant α such that for all t and k,*

$$\mathbb{E}\left[\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \tilde{A}_{i_t}^{k-1}(w_{t-1} - \tilde{w}))\|^2\right] \le \alpha \mathbb{E}\left[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|^2\right] \quad (4.3.4)$$

Note that, due to (4.3.3), SVRG-2 satisfies Assumption 4.3.2 when w_{t-1} is close to $\tilde{w}$.
The next lemma gives the bound of the gradient difference, and it is useful for further analysis.

Lemma 4.3.5. [64] *Suppose that Assumption 4.3.1 holds. Let* $w_* = \min_w f_i(w) - f(w_*) - \nabla f_i(w_*)^T(w - w_*)$. *Then*

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_{i_t}(w) - \nabla f_{i_t}(w_*)\|^2 \le 2L[F(w) - F(w_*)]. \tag{4.3.5}$$

Under Assumptions 4.3.1 and 4.3.2, we provide an upper bound on the variance.

Lemma 4.3.6. *Suppose that Assumptions 4.3.1 and 4.3.2 hold. Let* v_t *be*

$$v_t = (\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \tilde{g}) + (\tilde{A}^{k-1} - \tilde{A}_{i_t}^{k-1})(w_{t-1} - \tilde{w}) \tag{4.3.6}$$

Then variance $\mathbb{E}[\|v_t - \nabla F(w_{t-1})\|^2]$ *is bounded by* $4\alpha L(F(w_{t-1}) - F(w_*) + F(\tilde{w}) - F(w_*))$.

Proof. Recall that v_t is an unbiased estimate of $\nabla F(w_{t-1})$, i.e., $\mathbb{E}[v_t] = \nabla F(w_{t-1})$. Then its variance can be calculated as

$$\begin{aligned}
&\mathbb{E}[\|v_t - \nabla F(w_{t-1})\|^2] \\
&= \mathbb{E}[\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \tilde{g}) - \tilde{A}_{i_t}^{k-1}(w_{t-1} - \tilde{w}) + \tilde{A}^{k-1}(w_{t-1} - \tilde{w}) - \nabla F(w_{t-1})\|^2] \\
&= \mathbb{E}[\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \tilde{A}_{i_t}^{k-1}(w_{t-1} - \tilde{w})) \\
&\qquad\qquad - \mathbb{E}[\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \tilde{A}_{i_t}^{k-1}(w_{t-1} - \tilde{w})]\|^2] \\
&\le \mathbb{E}[\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \tilde{A}_{i_t}^{k-1}(w_{t-1} - \tilde{w}))\|^2].
\end{aligned}$$

Under the Assumption 4.3.2, it is easy to see that we have

$$\begin{aligned}
\mathbb{E}[\|v_t - \nabla F(w_{t-1})\|^2] &\le \alpha\mathbb{E}[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|^2] \\
&\le \alpha(2\mathbb{E}[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(w_*)\|^2] + 2\mathbb{E}[\|\nabla f_{i_t}(\tilde{w}) - \nabla f_{i_t}(w_*)\|^2]) \\
&\le 4\alpha L(F(w_{t-1}) - F(w_*) + F(\tilde{w}) - F(w_*)).
\end{aligned} \tag{4.3.7}$$

where the last inequality follows from the Lemma 4.3.5. ■

Now we proceed to prove the main result. The proof is similar to [64]. For the completeness, we give the proof.

Theorem 4.3.1. *Consider Algorithm 4 with option-2. Suppose that Assumptions 4.3.1 and 4.3.2 hold. Let* w_* *be the optimal solution of the problem* (4.0.1), *and* m *is sufficiently large so that*

$$\beta = \left[\frac{1}{\mu\eta(1 - \eta L(2\alpha + 1))m} + \frac{2L\eta\alpha}{1 - \eta L(2\alpha + 1)}\right] < 1,$$

then we have linear convergence in expectation:

$$\mathbb{E}[F(\tilde{w}_k) - F(w_*)] \le \beta^k \, \mathbb{E}[F(\tilde{w}_0) - F(w_*)].$$

Proof. Let $v_t = (\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \tilde{g}) + (\tilde{A}^{k-1} - \tilde{A}^{k-1}_{i_t})(w_{t-1} - \tilde{w})$ be the search direction in the k-th epoch of Algorithm 4. Then,

$$\begin{aligned}
\mathbb{E}[\|v_t\|^2] &= \mathbb{E}[\|v_t - \nabla F(w_{t-1}) + \nabla F(w_{t-1}) - \nabla F(w_*)\|^2] \\
&= \mathbb{E}[\|v_t - \nabla F(w_{t-1})\|^2] + \mathbb{E}[\|\nabla F(w_{t-1}) - \nabla F(w_*)\|^2] \\
&\le 4\alpha L(F(w_{t-1}) - F(w_*) + F(\tilde{w}) - F(w_*)) + 2L[F(w_{t-1}) - F(w_*)] \qquad (4.3.8) \\
&= 2L(2\alpha + 1)[F(w_{t-1}) - F(w_*)] + 4\alpha L[F(\tilde{w}) - F(w_*)] \\
&= 2L[(2\alpha + 1)(F(w_{t-1}) - F(w_*)) + 2\alpha(F(\tilde{w}) - F(w_*))], \qquad (4.3.9)
\end{aligned}$$

where the first inequality follows from the Lemma 4.3.6 and Lemma 4.3.5.

Now we bound the distance of w_t to w_*.

$$\begin{aligned}
&\mathbb{E}[\|w_t - w_*\|^2] \\
&= \mathbb{E}[\|w_{t-1} - \eta v_t - w_*\|^2] \\
&= \|w_{t-1} - w_*\|^2 - 2\eta \, \mathbb{E}((w_{t-1} - w_*)^T v_t) + \eta^2 \, \mathbb{E}[\|v_t\|^2] \\
&= \|w_{t-1} - w_*\|^2 - 2\eta(w_{t-1} - w_*)^T \nabla F(w_{t-1}) + \eta^2 \mathbb{E}[\|v_t\|^2] \\
&\le \|w_{t-1} - w_*\|^2 - 2\eta[F(w_{t-1}) - F(w_*)] \\
&\qquad + 2\eta^2 L[(2\alpha + 1) \, (F(w_{t-1}) - F(w_*)) + 2\alpha \, (F(\tilde{w}) - F(w_*))] \\
&= \|w_{t-1} - w_*\|^2 - 2\eta(1 - \eta L(2\alpha + 1)) \, [F(w_{t-1}) - F(w_*)] \\
&\qquad + 4\alpha\eta^2 L \, [F(\tilde{w}) - F(w_*)], \qquad (4.3.10)
\end{aligned}$$

where the first inequality is obtained by the convexity of F. By summing the previous inequality over $t = 0, 1, \ldots, m-1$ and taking expectation with all the history using option 2 at epoch k, we have

$$\begin{aligned}
&\mathbb{E}[\|w_m - w_*\|^2] + 2\eta(1 - \eta L(2\alpha + 1))m \, \mathbb{E}[F(\tilde{w}_k) - F(w_*)] \\
&\qquad \le \mathbb{E}[\|w_0 - w_*\|^2] + 4\alpha\eta^2 Lm \, \mathbb{E}[F(\tilde{w}) - F(w_*)]
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{E}[\|\tilde{w} - w_*\|^2] + 4Lm\eta^2\alpha \, \mathbb{E}[F(\tilde{w}) - F(w_*)] \\
&\le \frac{2}{\mu}\mathbb{E}[F(\tilde{w}) - F(w_*)] + 4\alpha\eta^2 Lm \, \mathbb{E}[F(\tilde{w}) - F(w_*)]
\end{aligned}$$

$$= 2\left(\frac{1}{\mu} + 2Lm\eta^2\alpha\right)\ \mathbb{E}[F(\tilde{w}) - F(w_*)],$$

where the second inequality uses the strong convexity of F. Finally, we have

$$\mathbb{E}[F(\tilde{w}_k) - F(w_*)] \leq \left[\frac{1}{\mu\eta(1-\eta L(2\alpha+1))m} + \frac{2L\eta\alpha}{1-\eta L(2\alpha+1)}\right]\ \mathbb{E}[F(\tilde{w}_{k-1}) - F(w_*)]. \tag{4.3.11}$$

which implies,

$$\mathbb{E}[F(\tilde{w}_k) - F(w_*)] \leq \beta^k\ \mathbb{E}[F(\tilde{w}_0) - F(w_*)]$$

with

$$\beta = \left[\frac{1}{\mu\eta(1-\eta L(2\alpha+1))m} + \frac{2L\eta\alpha}{1-\eta L(2\alpha+1)}\right].$$

■

Next we consider Algorithm 4 with option-1.

Theorem 4.3.2. *Consider Algorithm 4 with option-*1. *Suppose that Assumptions 4.3.1 and 4.3.2 hold. Let w_* be the optimal solution of the problem* (4.0.1), *and*

$$\gamma = \left[(1-2\eta\mu(1-\eta L(2\alpha+1)))^m + \frac{2\alpha\eta L^2}{\mu(1-\eta L(2\alpha+1))}\right]. \tag{4.3.12}$$

Then for k-th epoch, we have

$$\mathbb{E}\|\tilde{w}_k - w_*\|^2 \leq \gamma\ \|\tilde{w}_{k-1} - w_*\|^2. \tag{4.3.13}$$

Proof. We bound the distance of w_t to w_* using (4.3.10),

$$\begin{aligned}
&\mathbb{E}[\|w_t - w_*\|^2] \\
&= \|w_{t-1} - w_*\|^2 - 2\eta(1-\eta L(2\alpha+1))\ [F(w_{t-1}) - F(w_*)] + 4\alpha\eta^2 L\ [F(\tilde{w}) - F(w_*)] \\
&\leq \|w_{t-1} - w_*\|^2 - 2\eta(1-\eta L(2\alpha+1))\ [\nabla F(w_{t-1})^T(w_{t-1} - w_*)] \\
&\qquad\qquad + 4\alpha\eta^2 L\ [\nabla F(\tilde{w})^T(\tilde{w} - w_*)] \\
&\leq \|w_{t-1} - w_*\|^2 - 2\eta\mu(1-\eta L(2\alpha+1))\|w_{t-1} - w_*\|^2 + 4\alpha\eta^2 L^2\ \|\tilde{w} - w_*\|^2 \\
&= (1 - 2\eta\mu(1-\eta L(2\alpha+1)))\|w_{t-1} - w_*\|^2 + 4\alpha\eta^2 L^2\ \|\tilde{w} - w_*\|^2,
\end{aligned} \tag{4.3.14}$$

where the first inequality follows from the convexity of F, the second inequality follows from the strong convexity of F and the Lipschitz continuity of ∇F.

By applying the above inequality over the t and $\tilde{w}_{k-1} = w_0$ and $\tilde{w}_k = w_m$, we have

$$\begin{aligned}
&\mathbb{E}[\|\tilde{w}_k - w_*\|^2] \\
&\leq [1 - 2\eta\mu(1 - \eta L(2\alpha+1))]^m \|\tilde{w}_{k-1} - w_*\|^2 \\
&\qquad\qquad + 4\alpha\eta^2 L^2 \sum_{j=1}^{m} [1 - 2\eta\mu(1 - \eta L(2\alpha+1))]^j \|\tilde{w}_{k-1} - w_*\|^2 \\
&< \left[(1 - 2\eta\mu(1 - \eta L(2\alpha+1)))^m + \frac{2\alpha\eta L^2}{\mu(1 - \eta L(2\alpha+1))}\right] \|\tilde{w}_{k-1} - w_*\|^2 \\
&= \gamma \ \|\tilde{w}_{k-1} - w_*\|^2
\end{aligned}$$

Hence, the linear convergence follows,

$$\mathbb{E}[\|\tilde{w}_k - w_*\|^2] < \gamma \ \|\tilde{w}_{k-1} - w_*\|^2.$$

■

Note that γ is less than 1 when the stepsize η is sufficiently small. Then $\tilde{w}$ converges w_t linearly. Next, we show that Assumption 4.3.2 holds for the SVRG-2, SVRG-2D, and the SVRG-2BB variants, that is, these methods converge linearly. For the rest of analysis, we consider $\nabla^2 F_i$ is to be Lipschitz continuous with constant $\tilde{L}_i$ such that $\tilde{L} = \max_i \tilde{L}_i$ and $\mu = \min_i \mu_i$ for strong convexity of F.

Lemma 4.3.7. *Suppose that there exists M such that $\|w_{t-1} - \tilde{w}\|^2 \leq M$. Suppose also that $\nabla^2 F_i$ is Lipschitz continuous with constant $\tilde{L}_i$. Then Assumption 4.3.2 holds with*

$$\alpha = \frac{\tilde{L}^2}{4\mu^2} M$$

for the SVRG-2, where $\tilde{L} = \max_i \tilde{L}_i$ and $\mu = \min_i \mu_i$.

Proof. Note that the SVRG-2 uses the original Hessian, i.e., $\tilde{A}_{i_t}^{k-1} = \nabla^2 f_{i_t}(\tilde{w}_{k-1})$. Then from Lemma 4.3.2, we have

$$\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \nabla^2 f_{i_t}(\tilde{w}_{k-1})(w_{t-1} - \tilde{w}))\| \leq \frac{\tilde{L}_{i_t}}{2} \|w_{t-1} - \tilde{w}\|^2. \tag{4.3.15}$$

Moreover the strong convexity of f_t with modules μ_t implies

$$\mu_{i_t} \|w_{t-1} - \tilde{w}\| \leq \|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|.$$

From the above inequalities we have

$$\begin{aligned}\mathbb{E}[\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \nabla^2 f_{i_t}(\tilde{w}_{k-1}(w_{t-1} - \tilde{w}))\|^2] \\ &\leq \left[\frac{\tilde{L}_{i_t}^2}{4\mu_{i_t}^2}\|w_{t-1} - \tilde{w}\|^2\right] \mathbb{E}[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|^2] \quad (4.3.16) \\ &\leq \left[\frac{\tilde{L}_{i_t}^2}{4\mu_{i_t}^2}M\right] \mathbb{E}[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|^2] \quad (4.3.17) \\ &\leq \left[\frac{\tilde{L}^2}{4\mu^2}M\right] \mathbb{E}[\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|^2]. \quad (4.3.18)\end{aligned}$$

■

Now we proceed to see Assumption 4.3.2 for the SVRG-2BB and the SVRG-2D.

Lemma 4.3.8. *For SVRG-2BB and SVRG-2D, Assumption 4.3.2 holds with* $\alpha = \frac{4L^2}{\mu}$, *where* $\mu = \min_i \mu_i$.

Proof. First, we give the bound of the approximate Hessian obtained by the SVRG-2BB and the SVRG-2D separately.

i) Let $\tilde{A}_{i_t}^k$ obtained by the BB method, i.e.,

$$\tilde{A}_{i_t}^k = \frac{(s^k)^T(\nabla f_{i_t}(\tilde{w}_k) - \nabla f_{i_t}(\tilde{w}_{m-1}))}{\|s^k\|^2},$$

where $s^k = \tilde{w}_k - \tilde{w}_{k-1}$, $y^k = \nabla f_{i_t}(\tilde{w}_k) - \nabla f_{i_t}(\tilde{w}_{k-1})$ and $i_t \in \{1, \ldots, n\}$ chosen randomly. From the Lemma 4.3.1, we have

$$|(\nabla f_i(w) - \nabla f_i(y))^T(w - y)| \leq \|\nabla f_i(w) - \nabla f_i(y)\|\|w - y\| \leq L\|w - y\|^2 \quad (4.3.19)$$

Therefore, we have $\|\tilde{A}_{i_t}^k\| \leq L$.

ii) Let $\tilde{A}_{i_t}^k$ be obtained by diagonal Hessian. Since f_i has Lipschitz gradient and it is twice differentiable, we have

$$\|\nabla^2 f_i(w)\|_2 \leq L.$$

Since $\tilde{A}_{i_t}$ be a diagonal Hessian of the true Hessian, we have

$$\|\tilde{A}_{i_t}\|^2 \leq \|\nabla^2 f_{i_t}(w)\|^2 \leq L.$$

Consequently, we have $\|\tilde{A}_{i_t}\|^2 \leq L$ for both the SVRG-2BB and the SVRG-2D.

Now we proceed to determine the α.

$$\begin{aligned}
&\|(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) - \tilde{A}_{i_t}^{k-1}(w_{t-1} - \tilde{w}))\|^2 \\
&\quad \leq 2\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|^2 + 2\|A_{i_t}^{k-1}(w_{t-1} - \tilde{w})\|^2 \\
&\quad \leq 2L^2\|w_{t-1} - \tilde{w}\|^2 + 2L^2\|w_{t-1} - \tilde{w}\|^2 \\
&\quad \leq 4L^2\|w_{t-1} - \tilde{w}\|^2 \\
&\quad \leq \frac{4L^2}{\mu_{i_t}}\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|^2 \\
&\quad \leq \frac{4L^2}{\mu}\|\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w})\|^2,
\end{aligned}$$

where the first inequality uses $\|x + y\|^2 \leq 2\|x\|^2 + 2\|y\|^2$, and the second inequality follows from the Lipschitz continuity of the ∇f_{i_t}. Therefore, $\alpha = \frac{4L^2}{\mu}$ for the SVRG-2BB and the SVRG-2D. ■

We have proved the linear convergence for SVRG-2, for SVRG-2BB, and for SVRG-2D. Finally, we prove the linear convergence of SVRG-2BBS. Recall that SVRG-2BBS is option-1 of SVRG-2BB with the BB-step size η_k^t. Recall also that Theorem 4.3.1 and 4.3.2 hold for Algorithm 4 with the fixed stepsize η. We can prove the linear convergence of the SVRG-2BBS similar to these theorems.

Theorem 4.3.3. *Suppose that $\xi_t \in (\xi_0, \xi_1)$ where $0 < \xi_0 \leq \xi_1$, and assume that η_0 and η_1 satisfy $1 - \eta_1 L(2\alpha + 1) > 0$ and $(1 - 2\eta_0\mu(1 - \eta_1 L(2\alpha + 1))) \in [0, 1)$. Then SVRG-2BBS is linearly convergent in expectation:*

$$\mathbb{E}[\|\tilde{w}_k - w_*\|^2] \leq \tilde{\gamma}^k \|\tilde{w}_0 - w_*\|^2, \tag{4.3.20}$$

where

$$\tilde{\gamma} = \left[(1 - 2\eta_0\mu(1 - \eta_1 L(2\alpha + 1)))^m + \frac{2\alpha\eta_1^2 L^2}{\eta_0\mu(1 - \eta_1 L(2\alpha + 1))}\right].$$

Proof. Let η_k^t be a BB-step size in SVRG-2BBS, that is

$$\eta_k^t = \vartheta_k \ \xi_t = \frac{\xi_t}{m_1} \frac{\|\tilde{w}_{k-1} - \tilde{w}_{k-2}\|^2}{((\tilde{w}_{k-1} - \tilde{w}_{k-2})^T(\tilde{g}_{k-1} - \tilde{g}_{k-2}))},$$

where $g_k = \nabla F(\tilde{w}_k)$, and ξ is small positive constant. Note that η_k^t varies at each iteration.

We first obtain the lower and upper bounds of η_k^t using the L-Lipschitz continuity of $\nabla F(w)$ and the strong convexity of $F(w)$.

$$\mu\|\tilde{w}_{k-1} - \tilde{w}_{k-2}\|^2 \leq (g_{k-1} - g_{k-2})^T(\tilde{w}_{k-1} - \tilde{w}_{k-2}) \leq L\|\tilde{w}_{k-1} - \tilde{w}_{k-2}\|^2.$$

From the above inequalities, it is easy to see that η_k^t has lower and upper bound as follows:

$$\frac{\xi_0}{m_1 L} \leq \eta_k^t \leq \frac{\xi_1}{m_1 \mu}.$$

Let $\eta_0 = \frac{\xi_0}{m_1 L}$ and $\eta_1 = \frac{\xi_1}{m_1 \mu}$. In a way similar to show (4.3.10), we can show that

$$\mathbb{E}[\|w_t - w_*\|^2] \leq \|w_{t-1} - w_*\|^2 - 2\eta_0(1 - \eta_1 L(2\alpha+1))\ [F(w_{t-1}) - F(w_*)] \\ + 4\alpha\eta_1^2 L\ [F(\tilde{w}) - F(w_*)].$$

In a way similar to the arguments in the proof of Theorem 4.3.2, summing the inequality over $t = 0, \ldots, m-1$ and taking the expectation of the history, yield

$$\begin{aligned}\mathbb{E}[\|\tilde{w}_k - w_*\|^2] &\leq [1 - 2\eta_0\mu(1 - \eta_1 L(2\alpha+1))]^m \|\tilde{w}_{k-1} - w_*\|^2 \\ &\quad + 4\alpha\eta_1^2 L^2 \sum_{j=1}^{m} [1 - 2\eta_0\mu(1 - \eta_1 L(2\alpha+1))]^j \|\tilde{w}_{k-1} - w_*\|^2 \qquad (4.3.21) \\ &= \left[(1 - 2\eta_0\mu(1 - \eta_1 L(2\alpha+1)))^m + \frac{2\alpha\eta_1^2 L^2}{\eta_0\mu(1 - \eta_1 L(2\alpha+1))}\right] \|\tilde{w}_{k-1} - w_*\|^2 \\ &= \tilde{\gamma}\|\tilde{w}_{k-1} - w_*\|^2\end{aligned}$$

Hence, we have the linear convergence, that is

$$\mathbb{E}[\|\tilde{w}_k - w_*\|^2] < \tilde{\gamma}\ \|\tilde{w}_{k-1} - w_*\|^2.$$

■

4.4 Numerical results

In this section, we present the numerical results for the proposed algorithms.

First, we discuss the experiment setup for the numerical experiments. We performed all experiments on MATLAB R2018a on Intel(R) Xeon(R) CPU E7-8890 v4 @ 2.20GHz with 96 cores, and we used SGDLibrary [67] to perform the existing algorithms. We solve standard learning problems, that is, the ℓ_2-logistic regression:

$$\min_w F(w) = \frac{1}{n}\sum_{i=1}^{n} \log\left[1 + \exp(-b_i a_i^T w)\right] + \frac{\lambda}{2}\|w\|^2, \qquad (4.4.1)$$

and the ℓ_2-squared SVM:

$$\min_w F(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(\left[1 - b_i a_i^T w\right]_+\right)^2 + \frac{\lambda}{2}\|w\|^2, \tag{4.4.2}$$

where $a_i \in \mathbb{R}^d$ is feature vector and $b_i \in \{\pm 1\}$ is target label of the i-th sample, and λ is a ℓ_2 regularizer. We performed numerical experiments on benchmark datasets given in Table 5.3. The datasets are binary classification problems available on LIBSVM [24].

Table 4.2: Details of the datasets used in the experiments

Dataset	**Dimension**	**Training**	**Model**
ijcnn1	22	49990	SVM
mnist-38	784	11,982	SVM
adult	123	32,561	LR
gisette	5000	6000	LR
covtype	54	464808	LR

4.4.1 Comparison of ξ_t

We first conduct the numerical experiments on the three different possibility of ξ_t for the generalized BB step size in SVRG-2BBS. Recall that the step size η_k^t in SVRG-2BBS is given as,

$$\eta_k^t = \frac{\xi_t}{m_1}\frac{\|\tilde{w}_{k-1} - \tilde{w}_{k-2}\|^2}{\left((\tilde{w}_{k-1} - \tilde{w}_{k-2})^T(\tilde{g}_{k-1} - \tilde{g}_{k-2})\right)}, \tag{4.4.3}$$

and ξ_t is computed differently as given in Table 4.3, where ξ_t is of the form of decay $\xi_t = \frac{c_1}{1+c_2T}$, T is current iterate $T = km + t, c_1$ is initial stepsize and $c_2 = \eta_0\lambda$ and in the case of fix ξ_t, we use grid search with $\xi_t = c_1$. Figure 4.1 shows the comparison of M1, M2, and M3 on the

Table 4.3: Generalized variants of SVRG-2BBS

Methods	m_1	ξ_t **type**
M1	$2n$	fix ξ_1
M2	n	decay ξ_t
M3	1	decay ξ_t

adult, covtype, and *gisette* datasets for the *ℓ_2-regularized logistic regression* with $\lambda = 10^{-3}$. We tuned step size for all methods via grid search $c_1 \in \{10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ and we set $m = 2n$ (where n is sample size) as in SVRG [64]. From Figure 4.1 we see that M1 and M3 perform better among all three methods. Therefore, in the further experiments, we use M1 and M3 for comparison with SVRG-2BB and the other existing methods.

Considering the step-size of **M2** and the experiments presented in Figure 4.1, one can note that the choice of $m_1 = n$ seems excessively large as decreasing ξ_t results in a rapid

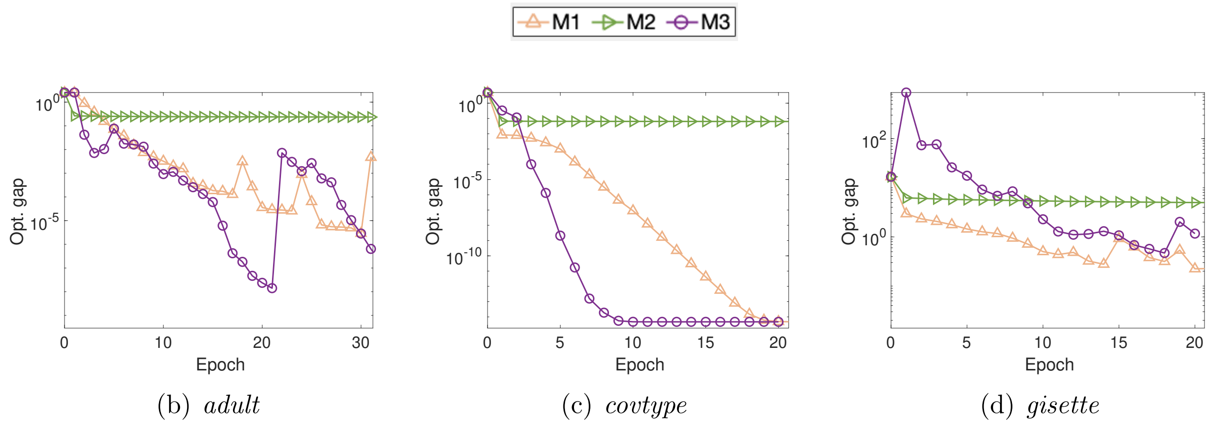


Figure 4.1: Comparison of M1, M2, and M3 with $\lambda = 10^{-3}$ on *adult, covtype,* and *gisette.*

reduction in the step-size η_k^t. Consequently, η_k^t becomes too small across all experiments, leading to almost null updates and proving ineffective in reducing the value of the objective function.

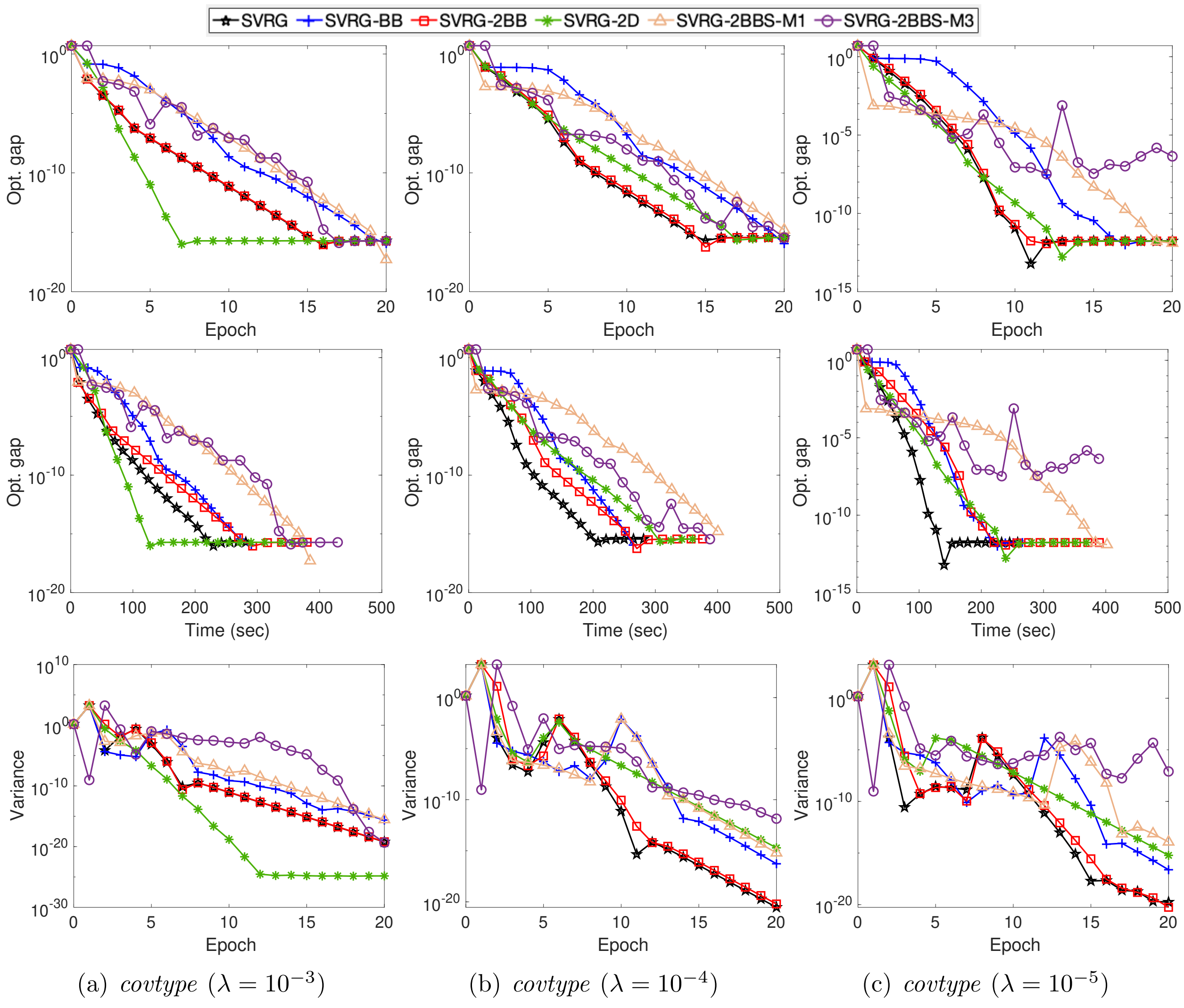
4.4.2 Comparison of SVRG, SVRG-BB, SVRG-2D, and the proposed methods

Next, we compare SVRG, SVRG-BB, and SVRG-2D with the proposed methods SVRG-2BB and SVRG-2BBS (M1 and M3).

We explain the experimental setup. We tuned the stepsize η for all methods via grid search $\eta \in \{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ for SVRG, SVRG-2D, SVRG-2BB, SVRG-BB, and $\xi_T \in \{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ for SVRG-2BBS (M1 and M3). It is essential to note that SVRG-BB highly depends on the initial step size η_0 as can be seen in the numerical results of [110]. This implies that even though we are using adaptive step size, we still need to hunt for the best initial (parameter) step size. Therefore, we are not benefiting from the SVRG-BB in terms of reducing the number of initial parameters, which does not outperform the SVRG-2BB, SVRG, and SVRG-2D.

For all methods, we set $m = 2n$. We present the optimiality gap with epoch and CPU time. Also, we present the variance information with epochs for each dataset. The variance is calculated as $\|v_k - \nabla f(w_k)\|^2$, where v_k is search direction at $t = 0$ of the k-th epoch.

We present numerical results in Figures 4.2-4.6. Each column of the figure represents the results for each dataset. In the figures in the first row of each column, the x-axis denotes epochs, and the y-axis denotes the optimality gap $F(\tilde{w}_k) - F(w_*)$, where w_* is obtained by limited memory BFGS method. In the second row of each column, the x-axis denotes CPU time and the y-axis denotes the optimality gap. In the figures of the third row of each column, the x-axis denotes epochs, and the y-axis denotes the variance $\|v_k - \nabla f(w_k)\|^2$.



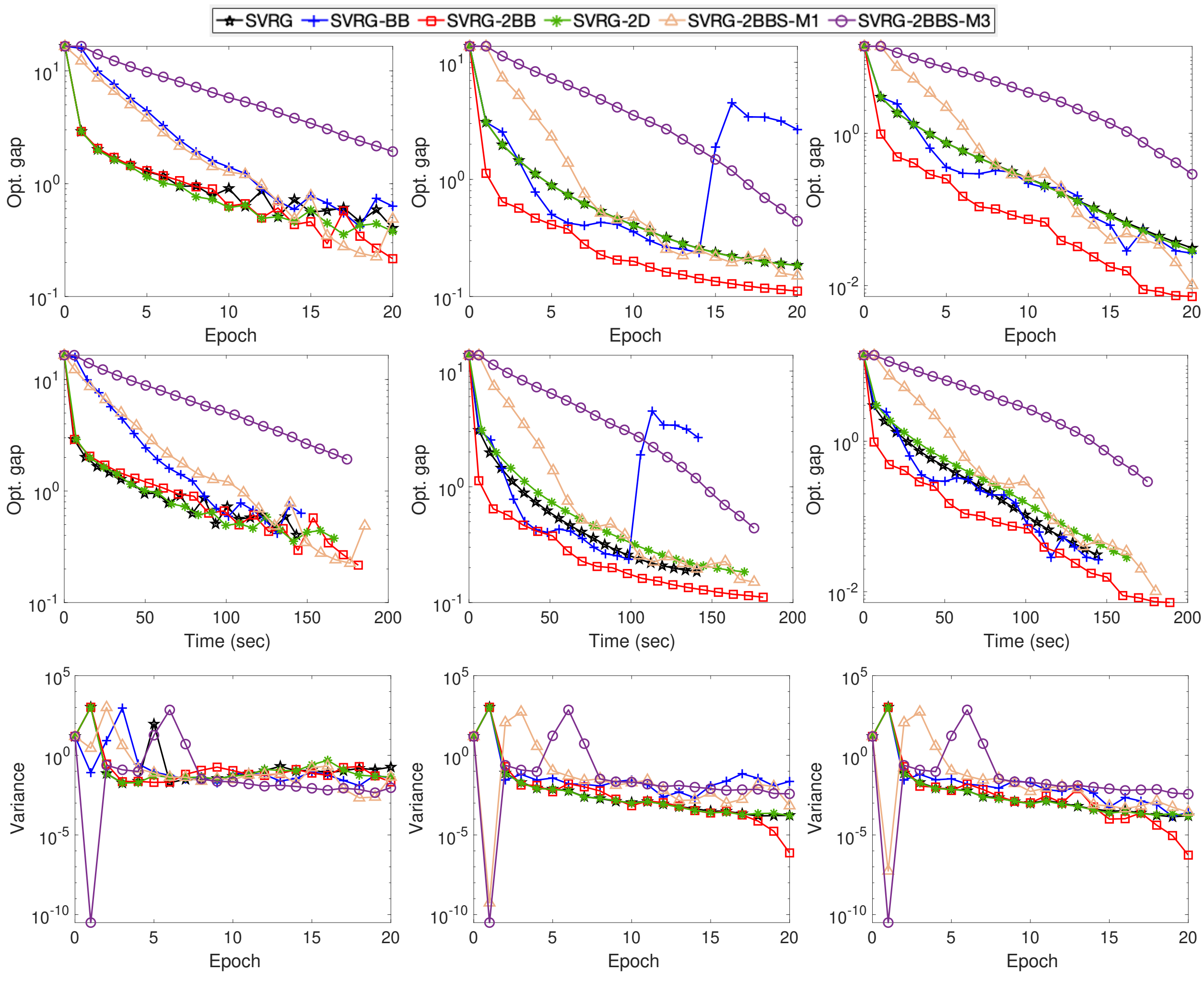
(a) *covtype* ($\lambda = 10^{-3}$) (b) *covtype* ($\lambda = 10^{-4}$) (c) *covtype* ($\lambda = 10^{-5}$)

Figure 4.2: Comparison on *covtype* dataset

We conduct numerical experiments on *covtype, gisette* and *adult* for the ℓ_2*-regularized logistic regression* with $\lambda = \{10^{-3}, 10^{-4}, 10^{-5}\}$ in Figure 4.2, Figure 4.3 and Figure 4.4, respectively. Figure 4.5 shows the numerical experiments on *mnist38* for the ℓ_2*-regularized squared SVM* with $\lambda = \{10^{-3}, 10^{-4}, 10^{-5}\}$, where we used binary class using those samples whose target classes were 3 and 8. Figure 4.6 shows the numerical experiments on *ijcnn1* for the ℓ_2*-regularized squared SVM* with $\lambda = \{10^{-3}, 10^{-4}\}$.

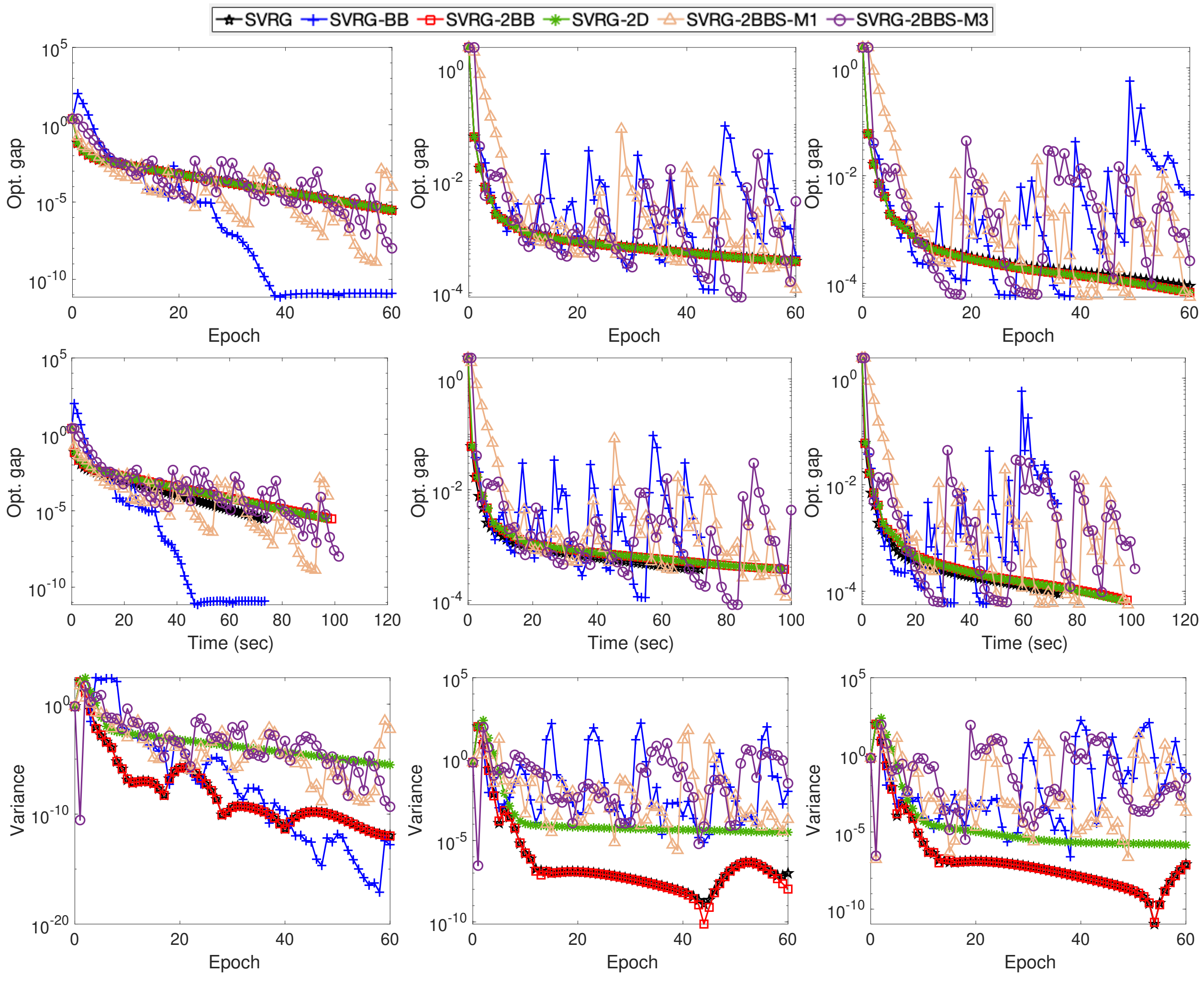
The figures for the ℓ_2*-regularized logistic regression* show that SVRG with second-order information outperforms pure SVRG. SVRG-2BB outperforms all methods when d is large (*i.e.,* when the ratio of n/d is close to 1) and λ is small. SVRG-2D performs better only on small $\lambda = 10^{-3}$ for the *covtype* dataset where n is large and d is very small. However, on the remaining datasets, SVRG-2D performs similarly to SVRG. As the variance of SVRG-BB and SVRG-2BBS is fluctuating, the BB-stepsize can be quite unstable.

The fugues for the ℓ_2-*regularized squares SVM* show that SVRG is unable to outperform other methods. Figure 4.5 shows SVRG-2D performs well in the initial epochs; however, SVRG-2BB, SVRG-2BBS (M1 and M3), and SVRG-BB perform similarly in the later epochs. Figure 4.6 shows that SVRG-2BB, SVRG-2BBS (M1 and M3), and SVRG-BB outperform SVRG and SVRG-2D for $\lambda = 10^{-3}$.



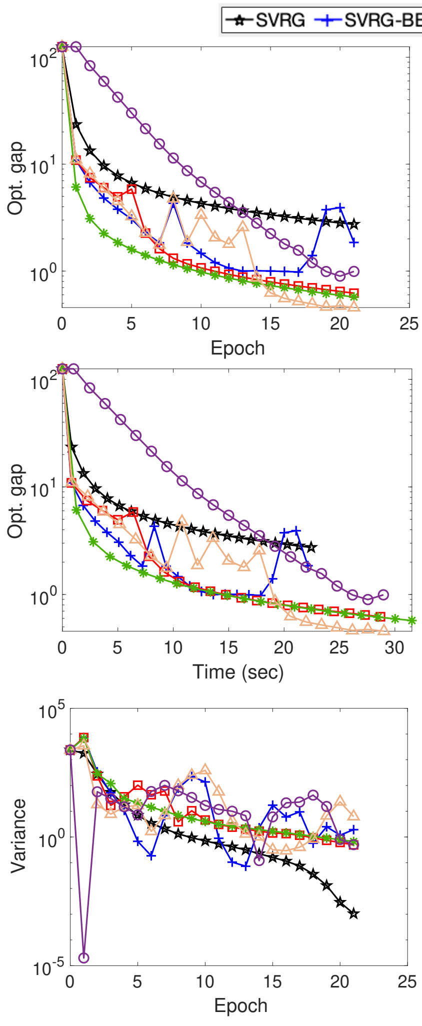
(a) *gisette* ($\lambda = 10^{-3}$) (b) *gisette*($\lambda = 10^{-4}$) (c) *gisette* ($\lambda = 10^{-5}$)

Figure 4.3: Comparison on *gisette* dataset

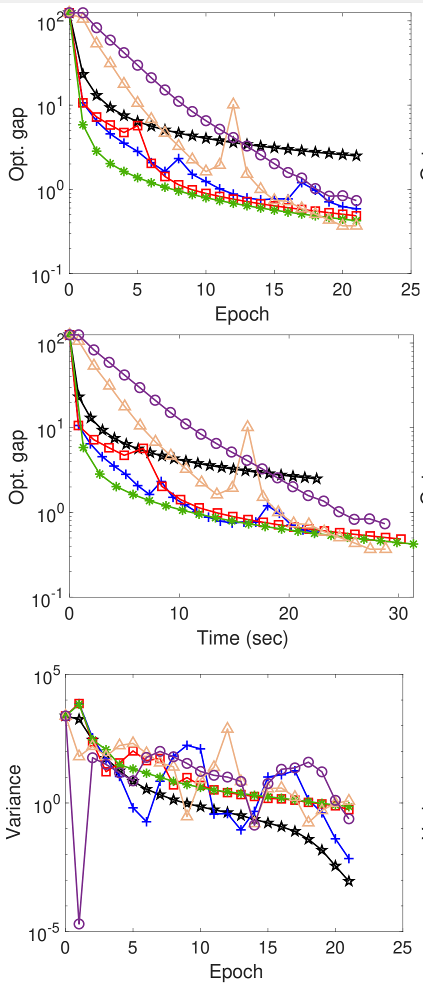


(a) *adult* ($\lambda = 10^{-3}$) (b) *adult* ($\lambda = 10^{-4}$) (c) *adult* ($\lambda = 10^{-5}$)

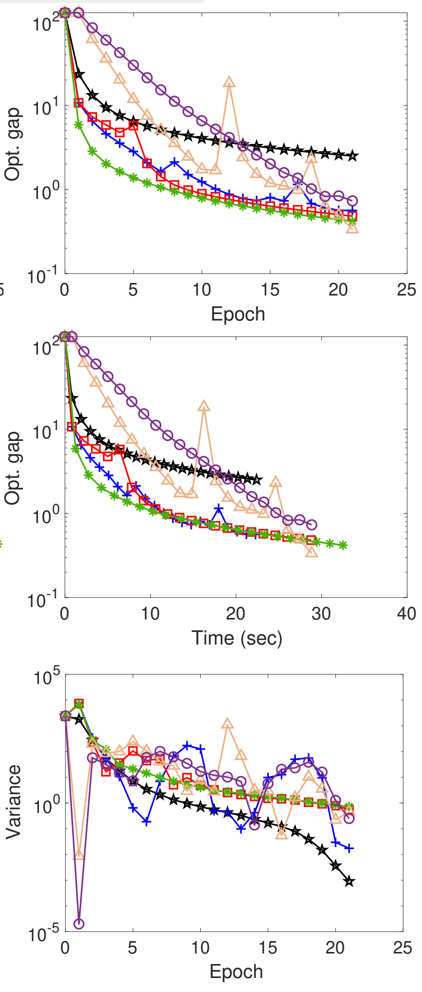
Figure 4.4: Comparison on *adult* dataset



(a) *mnist38* ($\lambda = 10^{-3}$)

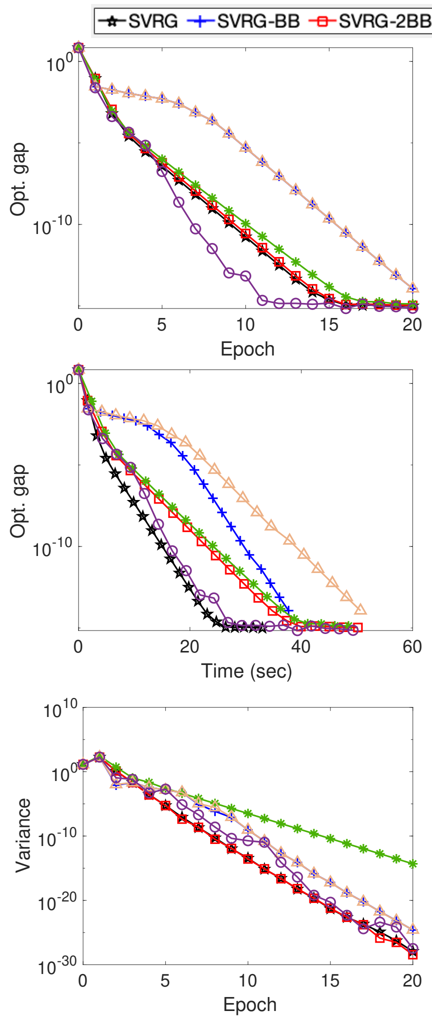


(b) *mnist38* ($\lambda = 10^{-4}$)

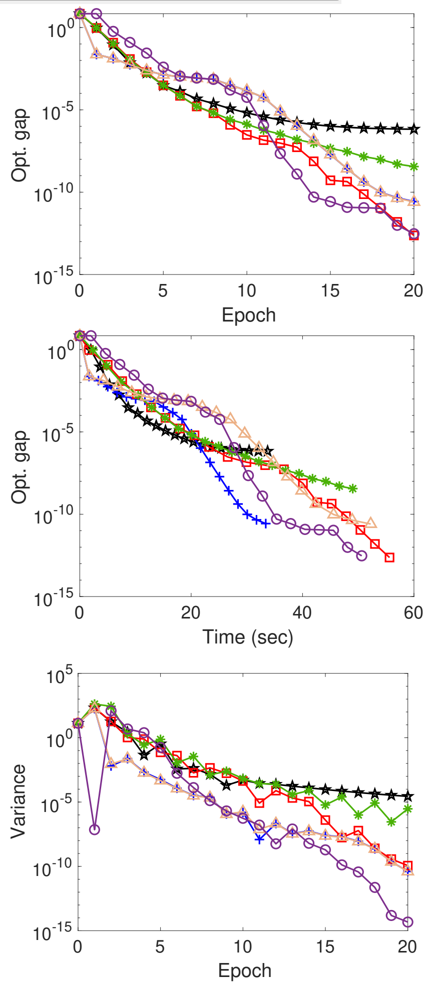


(c) *mnist38* ($\lambda = 10^{-5}$)

Figure 4.5: Comparison on *mnist38* dataset



(a) *ijcnn1* ($\lambda = 10^{-2}$)



(b) *ijcnn1* ($\lambda = 10^{-3}$)

Figure 4.6: Comparison on *ijcnn1* dataset

4.5 Conclusions

In this chapter, we proposed a method for reducing the variance of stochastic gradients using the Barzilai-Borwein approximation. We then incorporate the BB-step size of SVRG-BB [110] with SVRG-2BB. When the objective function is strongly convex, we proved the linear convergence of Algorithm 4 which includes SVRG-2BB, SVRG-2 [53], diagonal Hessian approximation SVRG-2D [53], SVRG-2BBS. We conducted the numerical experiments on the benchmark datasets and showed that the proposed method with a constant step size performs better than the existing variance-reduced methods for some test problems. In terms of optimal cost with respect to both CPU time and epoch, experimental results on benchmark datasets show that the SVRG-2BB performs stable.

Acknowledgment

This chapter is based on Hardik Tankaria, Nobuo Yamashita, "A stochastic variance reduced gradient using Barzilai-Borwein techniques as second order information," *Journal of Industrial and Management Optimization*, 2024, 20(2): 525-547. DOI: `https://doi.org/10.3934/jimo.2023089`. Reproduced with permission from American Institute of Mathematical Science.

Chapter 5

A Regularized Nyström method for Large-scale Unconstrained Optimization

5.1 Introduction

Efficiently optimizing diverse functions holds paramount importance in machine learning, especially with the surge in data volume. Managing large-scale optimization problems has emerged as a critical task. This chapter introduces a solution to this challenge by employing the Nyström approximation to approximate the Hessian matrix of the objective function. Our methodology targets the resolution of large-scale unconstrained optimization problems, characterized by the following form:

$$\min_{w \in \mathbb{R}^d} f(w), \tag{5.1.1}$$

where f is twice continuously differentiable and convex. (We provide the dedicated section for non-convex function in further section). The classical second-order optimizers like Newton's method yield quadratic convergence in solving (5.1.1). However, their applicability diminishes in high-dimensional optimization scenarios due to substantial per-iteration expenses and memory demands. To tackle this challenge, we present a method for low-rank Hessian approximation employing the iterative use of the Nyström method or, more broadly, a random column subset selection method to approximate the Hessian. This approach serves as a computationally efficient alternative, addressing the constraints posed by traditional second-order optimizers in high-dimensional problems.

5.1.1 Background

To optimize problem (5.1.1), first-order optimization methods such as stochastic gradient descent (SGD) [98], AdaGrad, stochastic variance-reduced gradient (SVRG) [64], Adam [71], and the stochastic recursive gradient algorithm (SARAH), possibly augmented with momentum, are preferred for large-scale optimization problems owing to their more affordable computational costs, which are linear in dimensions per epoch $O(nd)$, where n is the number of data samples. However, the convergence of the first-order methods is notably slow, and they are sensitive to hyperparameter choices and ineffective for ill-conditioned problems.

In contrast, Newton's method does not depend on the parameters of specific problems and requires only minimal hyperparameter tuning for self-concordant functions, such as ℓ_2-regularized logistic regression. However, Newton's method involves a computational complexity of $\Omega(nd^2 + d^{2.37})$ [2] per iteration and thus is not suitable for large-scale settings. To reduce this computational complexity, the subsampled Newton's method and random projection (or sketching) are commonly used to reduce the dimensionality of the problem and solve it in a lower-dimensional subspace. The subsampled (a.k.a. mini-batch) Newton method performs well for large-scale but relatively low-dimensional problems by computing the Hessian matrix on a relatively small sample. However, it is time-consuming for high-dimensional problems. Randomized algorithms [76, 93] estimate the Hessian in Newton's method using a random embedding matrix $S \in \mathbb{R}^{m\times n}$, $H_S(w) := X^\top S^\top SX$, where X is a matrix satisfying $X^\top X = \nabla^2 f(w)$. Specifically, their approximation used the square root of the generalized Gauss-Newton (GGN) matrix as a low-rank approximation instead of deriving it from actual curvature information, whereas S is a random projection matrix of size $m\times n$ and m is given by a user. Moreover, the Newton sketch [93] requires a substantially large sketch size, which can be as big as some multiple of the dimension d of w, which is not ideal and overmatches the objective of a low-rank Hessian approximation.

Recently, Derezinski et al. [36] proposed the Newton-LESS method, which is based on the leverage score-specified embeddings. It sparsified the Gaussian sketching and reduced the computational cost, with similar convergence properties as the dense Gaussian sketching.

Gower et al. [52] proposed the randomized subspace Newton (RSN) method. RSN is the randomized subspace Newton that computes the sketch of Hessian by sampling the embedding matrix S and approximating the Hessian as $S(S^\top HS)^\dagger S^\top$.

Talwalkar in his thesis [108] proposed the Nyström logistic regression algorithm, where the Nyström method is used to approximate the Hessian of the regularized logistic regression. Thus, it can be regarded as a variant of Nyström-SGD. However, the thesis [108] only considered regularized logistic regression, in which the Hessian can be explicitly obtained, with deterministic optimization. In contrast, we propose the regularized Nyström method for deterministic and stochastic optimization, such that the value of the regularizer depends on the norm of the gradient or stochastic gradient, respectively. We also show its theoretical aspects in terms of rank and the number of randomly picked columns.

The limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm [78] is a widely used quasi-Newton method. More specifically, it estimates the Hessian inverse using the past difference of gradients and updates. The online BFGS (oBFGS) [101] method is a stochastic version of regularized BFGS and L-BFGS with gradient descent. Kolte et al. [73] proposed two variants of a stochastic quasi-Newton method incorporating a variance-reduced gradient. The first variant used a sub-sampled Hessian with singular value thresholding. The second variant used the LBFGS method to approximate the Hessian inverse. The stochastic quasi-Newton method (SQN) [16] used the Hessian vector product computed on a subset of each mini-batch instead of approximating the Hessian inverse from the difference between the current and previous gradients, as in LBFGS. SVRG-SQN [81] also incorporated variance-reduced gradients.

5.2 Nyström approximation and its properties

Dealing with large datasets, the computational complexity of second-order optimization methods poses a significant challenge. Consequently, there is a necessity to explore Hessian approximation techniques that are both computationally viable and theoretically grounded. In the past few decades, researchers have delved into diverse matrix approximation methods. A prevailing strategy involves attaining a low-rank approximation by strategically selecting specific components of the original matrix through various methodologies. One such popular method in this context is the Nyström approximation [39], initially introduced for the kernel approximation. The Nyström approximation is a low-rank approximation of a positive semidefinite matrix that leverages partial information from the original matrix to construct an approximate matrix of lower rank. The Nyström method can be categorized as a variant of the column subset selection problem. Talwalkar [109] proposed minimizing the error using low-coherence bounds of the Nyström method. Michel Derezinski [35] proposed improvements in the approximation guarantees of column subset selection and the Nyström method using spectral properties.

Definition 5.2.1 (Nyström approximation). *Let $H \in \mathbb{R}^{d\times d}$ be a symmetric positive semi-definite matrix. Then, choose m columns of H randomly to form a $d \times m$ matrix C. Let $m \times m$ be a matrix M such that it is formed by the intersection of those m columns and corresponding m rows of H. M_k is the best k-rank approximation of M. A k-rank Nyström* *approximation N_k of H can be defined as*

$$N_k = CM_k^{\dagger}C^{\top}, \tag{5.2.1}$$

where $M_k^{\dagger}$ is a pseudo-inverse of M_k. Letting $H = \nabla^2 f(w)$ to be a Hessian matrix of the objective function (5.1.1)*, following theorem shows the distance between the Hessian H and the Nyström approximation N of H.*

Theorem 5.2.1. *[39, Theorem 3] Let H be a $d \times d$ matrix and let $N_k = CM_k^{\dagger}C^{\top}$ be a k-rank ($k \leq m$) is a Nyström approximation [39, Algorithm 2] by sampling m columns of H with probabilities $\{p_i\}_{i=1}^d$ such that*

$$p_i = \frac{H_{ii}^2}{\sum_{i=1}^d H_{ii}^2}. \tag{5.2.2}$$

Let $r = rank(M)$ and let H_k be the best k-rank approximation of the H. In addition, let $\varepsilon > 0$ and $\vartheta = 1 + \sqrt{8\log(1/\varrho)}$. If $k < r$, and (a) $m \geq 64k\vartheta^2/\varepsilon^4$, (b) $m \geq 4\vartheta^2/\varepsilon^4$, then with probability at least $1 - \varrho$

$$\|H - N_k\|_\nu \leq \|H - H_k\|_\nu + \varepsilon \sum_{i=1}^d H_{ii}^2, \tag{5.2.3}$$

for (a) $\nu = F$ (Frobenius) and (b) $\nu = 2$ (spectral). If $k \geq r$, then $M_k = M$ and

$$\|H - N_r\|_2 \leq \varepsilon \sum_{i=1}^d H_{ii}^2,$$

in expectation and with high probability.

We denote above upper bound as $U_{Nys} = \|H - H_k\|_\nu + \varepsilon \sum_{i=1}^d H_{ii}^2$ for the rest of chapter.

An alternative way to define a k-rank Nyström approximation is via a zero-one sampling matrix.

Remark 5.2.1. *Consider an instance of a function $f(w) = \ell(Aw)$, where $A \in \mathbb{R}^{n\times d}$ and $\ell : \mathbb{R}^n \to \mathbb{R}$ has separable form such that $\ell(Aw) = \sum_{i=1}^n \ell_i(\langle a_i, w\rangle)$. The Hessian of f can be computed as $\nabla^2 f(w) = X^{\top}X$, where $X = diag\{\ell_i''\}_{i=1}^n A$.*

The zero-one matrix $W \in \mathbb{R}^{d\times m}$ can be constructed as follows.

$$W_{i,j} = \begin{cases} 1 & \text{if the } i\text{-th column is chosen in} \\ & \text{the } j\text{-th random trail of choosing columns,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.2.4}$$

We can write the Nystöm approximation using zero-one matrix as follows:

$$C(M_k)^{\dagger}C^{\top} = (HW)(W^{\top}HW)_k^{\dagger}(HW)^{\top}. \tag{5.2.5}$$

Drineas and Mahoney [39] shows that the uniform sampling case of scaled Nyström brings the same expression as the (5.2.5). It can be defined as follows:

$$C(M_k)^{\dagger}C^{\top} = (HWD)((WD)^{\top}HWD)_k^{\dagger}(HWD)^{\top},$$

where $D \in \mathbb{R}^{m\times m}$ is a scaling matrix that has diagonal entries $1/\sqrt{mp_{i_l}}$, p_{i_l} is a probability $\mathbf{P}(i_l = i) = p_i$ given in (5.2.2) of the Theorem 5.2.1 and i_l is a column chosen in lth independent trail. Moreover, $C := HW$, which is the sampled column matrix of the true Hessian, and $M := W^{\top}HW$, which is the intersection matrix in (5.2.1). However, if we let $m = k$ and then in the case of uniform sampling, the probability $p_i = 1/d$, and scaling matrix have diagonal entries $D_{ii} = \sqrt{\frac{d}{m}}$ which obtains the approximation (5.2.1) that is exactly the same as (5.2.5).

Let $S = WD$ and one can compute Nyström approximation using S. Note that the generalized Nyström method analyzed in [46, 49, 118] considers the theory with the Gaussian and various interesting random matrices S. Therefore, we also consider S to be a Gaussian random matrix.

Lemma 5.2.1. *[48] Let S be a $d\times m$ random matrix such that s_{ij} are independently sampled from the normal distribution* $\mathrm{N}(0, 1/\mathrm{m})$*, then there exists $\mathcal{C} > 0$ such that*

$$\|S^{\top}S\| \leq \mathcal{C}\frac{d}{m}.$$

with probability at least $1 - 2\exp(-m)$, where $\mathcal{C}$ is an absolute constant.

One can prove above lemma from the [122, Theorem 4.6.1]. For the rest of theoretical analysis, we consider the matrix S to be a generalized random matrix given in Lemma 5.2.1.

5.2.1 Approximation of Hessian by Nyström approximation

In this section, we first define a formulation of the Nyström approximation for the objective function (5.1.1) and propose the regularized Nyström algorithm for the unconstrained optimization problem.

Let $H = \nabla^2 f(w)$ be a Hessian of the objective function, and we pick $\Omega \subseteq \{1, 2, \ldots, d\}$ indices uniformly at random such that $m = |\Omega|$ and compute the Nyström approximation as

$$N_k = CM_k^{\dagger}C^{\top} = ZZ^{\top}, \tag{5.2.6}$$

where $C \in \mathbb{R}^{d\times m}$ is a matrix consisting of m columns ($m \ll d$) of H, M is $m \times m$ intersection matrix of m columns and m rows with same indices, and the rank of M is $k \leq m$. We first obtain the best k rank approximation using the singular value decomposition (SVD) of M_k as $M_k = U_k \Sigma_k U_k^\top$, $U_k \in \mathbb{R}^{m\times k}$ are singular vectors and $\Sigma_k \in \mathbb{R}^{k\times k}$ consisting of k singular values. Then we compute the pseudo-inverse $M_k^\dagger = U_k \Sigma_k^{-1} U_k^\top$. Using $U\Sigma_t^{-1/2}$ from the pseudo-inverse of M and C, we compute $Z = CU_k\Sigma_k^{-1/2} \in \mathbb{R}^{d\times k}$, and we get $N_k = ZZ^\top$. Note that the number of columns m is a hyperparameter.

Example 5.2.1. *ℓ_2-regularization: In this example, we present the relation between the ℓ_2 regularization and a fixed rank Nyström approximation. Consider the ℓ_2 regularized optimization problem:*

$$\min_{w\in\mathbb{R}^d} \left\{ f(w) := \sum_{i=1}^{n} f_i(w) + \frac{\lambda}{2}\|w\|^2 \right\}, \tag{5.2.7}$$

where each f_i is convex, twice continuously differentiable, and $\lambda > 0$, and hence f is a strongly convex function. Then the Hessian of the ℓ_2-regularized function can be given as $H = \sum_{i=1}^n \nabla^2 f_i(w) + \lambda I$ and $\lambda_{\min}(\nabla^2 f(w)) \geq \lambda$. For this problem, we have $C = S^\top \left(\sum_{i=1}^n \nabla^2 f_i(w)\right) + \lambda S^\top I$ and $M = S^\top \left(\sum_{i=1}^n \nabla^2 f_i(w)\right) S + \lambda S^\top S \in \mathbb{R}^{m\times m}$, where S is $S = WD$ with the zero-one matrix W given in (5.2.4), and D is a scaling matrix. When S has rank m, matrix M becomes positive definite, and hence it becomes the fixed-ranked Nyström approximation, which also helps in the convergence to get the minimum eigenvalue of M^{-1}. Hence, we can write it as

$$N = CM^{-1}C^\top$$

for fixed rank Nyström approximation.

5.3 A Regularized Nyström-gradient method

The second-order optimization methods often utilize the regularized approximated Hessian. The regularized parameters can be obtained through approaches such as the trust-region method or by adaptively determining the regularization parameter based on the gradient information. These approaches have been explored in previous works such as [77, 120, 112], which propose iterative formulations similar to Chapter 3:

$$w_{t+1} = w_t - \eta_t (A_t + \rho_t \mathbf{I})^{-1} \nabla f(w_t), \tag{5.3.1}$$

where, A_t represents a Hessian approximation, and $\rho_t > 0$ is a regularized parameter.

Now, consider A_t to be Nyström approximation N_t in equation (5.3.1). To ensure non-singularity and obtain a descent direction, we compute a regularized Nyström approximation.

Then we can write an iterate of the regularized Nyström approximation as

$$w_{t+1} = w_t - \eta_t(N_t + \rho_t\mathbf{I})^{-1}\nabla f(w_t). \tag{5.3.2}$$

Since we are approximating the Hessian using Nyström method augmented with a regularizer in a similar quasi-Newton framework that uses the multiple of gradient in the search direction, we call our novel method "A Regularized Nyström gradient method (NGD)". The regularized parameter $\rho_t > 0$ is determined based on the gradient information. Specifically, we set $\rho_t = c_1\|\nabla f(w_t)\|^\gamma$ as similar to given in [120], where $c_1 > 0$. We consider ρ_t to be either $c_1\sqrt{\|\nabla f(w_t)\|}$ for $\gamma = 1/2$, $c_1\|\nabla f(w_t)\|$ for $\gamma = 1$, or $c_1\|\nabla f(w_t)\|^2$ for $\gamma = 2$ as shown in Table 5.1. We denote $\nabla f(w_t) = g_t$ for the rest of the chapter.

Table 5.1: Relation between proposed methods and value of γ

Proposed methods	Value of γ	Regularizer ρ_t	Regularized Nyström
NGD	$\gamma = 1/2$	$\rho_t = c_1\|g_t\|^{1/2}$	$N_t + c_1\|g_t\|^{1/2}$
NGD1	$\gamma = 1$	$\rho_t = c_1\|g_t\|^1$	$N_t + c_1\|g_t\|^1$
NGD2	$\gamma = 2$	$\rho_t = c_1\|g_t\|^2$	$N_t + c_1\|g_t\|^2$

To efficiently compute the inverse of $(N_t + \rho_t)$ given in (5.3.2), we use the Sherman–Morrison–Woodbury identity as

$$p_t = (N_t + \rho_t I)^{-1} g_t = \frac{1}{\rho_t} g_t - Q_t Z_t^\top g_t, \tag{5.3.3}$$

where p_t is search direction at tth iteration, N_t is Nyström approximation computed at w_t, g_t is a gradient computed at w_t and $Q_t = \frac{1}{\rho_t^2} Z_t(I_k + \frac{1}{\rho_t} Z_t^\top Z_t)^{-1}$. Here, $(I_k + \frac{1}{\rho_t} Z_t Z_t^\top) \in \mathbb{R}^{k\times k}$, and its inverse can be computed much more quickly than the inverse of $(N_t + \rho_t I)$ directly. We use the backtracking line search with Armijo's line search rule that finds a step size $\eta_t = \alpha^{(\ell)} = \tau\alpha^{(\ell-1)}$, starting from $\ell = 0$, the initial step size $\eta_0 = \alpha^{(0)}$, and finds the least positive integer $\ell \geq 0$ and increased ℓ by $\ell + 1$ until the

$$f(w_t + \alpha^{(\ell)} p_t) \leq f(w_t) + \alpha^{(\ell)} \beta g_t^\top p_t, \tag{5.3.4}$$

holds, where $\alpha, \beta \in (0, 1)$. Next, we introduce the main algorithm.

The efficiency of the method depends on both the rank of the Hessian and the choice of the sketching matrix S. For example, if the sketch size goes to 1, then the method reduces to a scaled gradient descent. Next, we discuss the computational complexity of the proposed algorithm.

Algorithm 5 A Regularized Nyström-Gradient Algorithm

```
1: **Initialize** Initial parameters $w_0$, desired rank $|\Omega| = m$, $\alpha, \beta \in (0,1)$, and maximum iterations $t_{\max}$
2: $t \leftarrow 0$
3: **repeat**
4:    $g_t = \nabla f(w_t)$
5:    randomly pick indices set $\Omega \subseteq \{1, 2, \dots, d\}$ such that $m = |\Omega|$
6:    compute $C_t$ ($\Omega$ columns of the Hessian)
7:    compute $Z_t$ using (5.2.6) and compute $\rho_t$
8:    $Q_t = \frac{1}{\rho_t^2} Z_t (I_k + \frac{1}{\rho_t} Z_t^T Z_t)^{-1}$
9:    Compute $(N_t + \rho_t I)^{-1} g_t$ using (5.3.3)
10:   Use backtracking line search with Armijo's rule to find $\eta_t$ using (5.3.4)
11:   $w_{t+1} = w_t - \eta_t p_t$
12:   $t = t + 1$
13: **until** $t = t_{\max}$ or some termination criteria is satisfied
14: **return** $w_t$
```

5.3.1 Computational complexity

Here, we analyze the per-iteration computational complexity of the proposed method for the fixed-rank Nyström approximation on ℓ_2-regularized objective function.

Consider an ℓ_2-regularized logistic regression objective function as given in Equation (4.4.1). First-order methods like gradient descent require $O(dn)$ computations to determine the gradient. Meanwhile, second-order methods such as Newton's method entail a complexity of $O(nd^2)$ for computing the Hessian and an additional $O(d^3)$ for obtaining its inverse. Consequently, the total cost of employing the full Newton's method sums up to $O(d^3 + nd^2)$.

In contrast, the proposed approach incurs a cost of $O(ndm)$ for computing C_t. The cost of computing the SVD of $M_t = U_t \Sigma_t U_t^\top$ is $O(m^3)$. Since Σ_t is diagonal, pseudo-inverse can be computed by just inverting diagonal elements, which is negligible as $m \ll d$, and $O(dm^2 + m^3)$ for the computation of Z_t. Lastly, the cost of Q_t costs $O(dm^2 + m^3)$ for the Sherman-Morrison update $(Z_t Z_t^\top + \rho_t I)^{-1} g_t$. Thus, the overall computational cost per iteration stands at $O(dm^2 + m^3 + ndm)$, which is directly proportional to that of the first-order method and significantly smaller than that of Newton's method.

5.3.2 Regularized Nyström as Newton sketch

In this section, we introduce an alternate definition of the Nyström approximation. The Nyström approximation can be obtained by sampling the embedding (random sketch) matrix. We further show that the resultant formulation of an alternate definition of the Nyström approximation can be interpreted as a Newton sketch-based method [93, 76]. Consider the Nyström approximation and let $H = X_{d\times n}^\top X_{n\times d}$ and zero-one $d \times m$ matrix W in (5.2.4) with $C_X = XW$. Let SVD of XW be $\widehat{U}\widehat{\Sigma_t}\widehat{V}^\top$, and $M = (C_X^\top C_X) = \widehat{V}\widehat{\Sigma_t}^2\widehat{V}^\top$. Then, similar to [39, Lemma 4] we obtain,

$$\begin{aligned} C(M_k)^\dagger C^\top &= (HW)(W^\top HW)_k^\dagger (HW)^\top \\ &= (X^\top C_X)(C_X^\top C_X)_k^\dagger (X^\top C_X)^\top \\ &= X^\top(\widehat{U}\widehat{\Sigma_{tk}}\widehat{V}^\top)(\widehat{V}\widehat{\Sigma_{tk}}^{-2}\widehat{V}^\top)(\widehat{V}\widehat{\Sigma_{tk}}\widehat{U}^\top)X \\ &= X^\top \widehat{U}_k\widehat{U}_k^\top X, \end{aligned} \tag{5.3.5}$$

where $\widehat{U}_k$ is the $k-$rank matrix. The right-hand side of (5.3.5) is similar to the Newton sketch [93] with two differences: **1)** embedding matrix P depends on the size of n and not d, whereas the zero-one matrix $W \in \mathbb{R}^{d\times m}$ depends on the d and **2)** the natural orthogonal matrix $\widehat{U}_k$ in the proposed method is replaced by a randomized embedding matrix $P^\top \in \mathbb{R}^{n\times m}$, which is expected to be orthogonal in principle. *i.e.,* $\mathbb{E}[P^\top P] = I$, whereas the proposed method produces the natural orthogonal matrix; *i.e.,* $\widehat{U}\widehat{U}^\top = I$. Consequently, Newton-Sketch needs a large and thick column matrix P (assuming most data has $n > d$) to approximate the Hessian.

If we let $X^\top X = \nabla^2 f(w)$ then, our approximation is of the form of

$$H_W = X^\top \widehat{U}\widehat{U}^\top X + \rho I \tag{5.3.6}$$

More generally, the approximation given above can be written in the form of an embedding matrix as follows: Let $Y = \rho I_d$, and let $Y^{1/2} = \sqrt{\rho}\cdot I_d$ be a $d \times d$ matrix. Then, by defining the embedding matrix, $\bar{S} = \begin{bmatrix} \widehat{U}_{m\times n}^\top & \mathbf{0}_{m\times d} \\ \mathbf{0}_{d\times n} & I_d \end{bmatrix}$ and partial Hessian $\bar{H} = \begin{bmatrix} X^\top \\ Y^{1/2} \end{bmatrix}$, we get

$$H_S = \bar{H}^\top \bar{S}^\top \bar{S}\bar{H},$$

which is identical to the (5.3.6) and hence H_S^{-1} is non-singular, where H_S is the Nyström approximation for H_S.

5.4 Convergence analysis

In this section, we provide the analysis that is based on selecting the number of columns, m, in the Nyström approximation. We investigate the distance between the Newton's direction and the NGD's search direction that is based on the rank of matrix M. We further prove the linear convergence of the proposed algorithm. Moreover, in the last subsection, we discuss the closeness of the inverse of regularized Nyström with the inverse of Hessian. This analysis offers insights into the overall convergence behavior of the algorithm. It is important to note that our convergence analysis is based on the objective function defined in equation (5.2.7).

Next, we provide the convergence analysis. First, we need following assumptions.

Assumption 5.4.1. *i) The objective function* (5.2.7) *is twice continuously differentiable and f is L_g-smooth,* i.e.,

$$\|\nabla^2 f(w_t)\| \leq L_g, \quad \forall\ w_t \in \mathbb{R}^d. \tag{5.4.1}$$

ii) The objective function (5.2.7) *is strongly convex.*

Assumption 5.4.2. *S_t is a random matrix whose entries are independently sampled Normal distribution with mean* 0 *and variance* $1/m$*, satisfies*

$$\|S^\top S\| \leq \mathcal{C}\frac{d}{m},$$

for some $\mathcal{C} > 0$.

Assumption 5.4.3. *For dimension d, we have a constraint on the value of m such that $m = o(d)$.*

Note that Assumption 5.4.3 is important, as in the case where $m = d$, the Nyström approximation results in the Hessian, *i.e.,* $HH^\dagger H = H$ and it turns out to be the Newton's method.

In the next lemma, we obtain a lower bound of the *minimum* eigenvalue and an upper bound of the *maximum* eigenvalue of $(N_t + \rho_t I)^{-1}$.

Lemma 5.4.1. *Suppose that Assumption 5.4.1, and 5.4.2 hold. Let w_t iterate obtained by Algorithm 5, and for some m, the maximum and minimum eigenvalues of $(N_t + \rho_t I)^{-1}$ are given as*

$$\lambda_{\min}[(N_t + \rho_t I)^{-1}] \geq \frac{1}{\frac{\mathcal{C}L_g^2 d}{m\lambda} + c_1\|g_t\|^\gamma}, \quad \text{and} \quad \lambda_{\max}[(N_t + \rho_t I)^{-1}] = \frac{1}{c_1\|g_t\|^\gamma}. \tag{5.4.2}$$

Proof. First we obtain the bound on *minimum* eigenvalue of $(N_t + \rho I)^{-1}$.

$$\begin{aligned}
\lambda_{\min}[(N_t + \rho_t I)^{-1}] &= \frac{1}{\lambda_{\max}(N_t + \rho_t I)} \\
&\geq \frac{1}{\lambda_{\max}(H_t S_t (S^\top H_t S_t^\dagger) S^\top H_t) + \rho_t I} \\
&\geq \frac{1}{\|H_t\|^2 \|S_t^\top S_t\| \|(S^\top H_t S_t)^{-1}\| + \rho_t}
\end{aligned}$$

$$
\begin{aligned}
&\geq \frac{1}{L_g^2\left(\frac{\mathcal{C}d}{m}\right)\left(\frac{1}{\lambda}\right)+\rho_t} \\
&= \frac{1}{\frac{\mathcal{C}L_g^2 d}{m\lambda}+c_1\|g_t\|^{\gamma}},
\end{aligned}
\tag{5.4.3}
$$

where the third inequality follows from the $\|H\| \leq L_g$, Lemma 5.2.1, and since f is strongly convex and $m \ll d$, $(S^\top HS) \succeq \lambda I$. Now we find obtain the bound on *maximum* eigenvalue of $(N_t+\rho I)^{-1}$.

$$
\begin{aligned}
\lambda_{\max}[(N_t+\rho_t I)^{-1}] &= \frac{1}{\lambda_{\min}(N_t+\rho_t I)} \\
&= \frac{1}{\rho_t} \\
&= \frac{1}{c_1\|g_t\|^{\gamma}}.
\end{aligned}
$$

Since N_t is positive semi-definite ρ_t is *minimum* eigenvalue of $(N_t+\rho_t I)$. This completes the proof. ■

5.4.1 Linear convergence

Next, we discuss a lemma related to search direction to obtain the linear convergence.

Lemma 5.4.2. *Let p_t be a descent direction of Algorithm 5 at iteration t, then*

$$g_t^\top p_t \leq -\rho_t\|p_t\|^2.$$

Proof. Let $p_t = -(N_t+\rho_t I)^{-1}g_t$ be a search direction. Next, consider

$$
\begin{aligned}
-g_t^\top p_t &= g_t^\top(N_t+\rho_t I)^{-1}g_t \\
&= ((N_t+\rho_t I)^{-1}g_t)^\top(N_t+\rho_t I)(N_t+\rho_t I)^{-1}g_t \\
&= p_t(N_t+\rho_t I)p_t \\
&\geq \rho_t\|p_t\|^2,
\end{aligned}
$$

where the last inequality comes from the fact that N_t is positive semidefinite.
This completes the proof. ■

Assumption 5.4.4. *For all $\boldsymbol{x}, \boldsymbol{y}$, the gradient is Lipschitz,* i.e.,

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L_g \|\boldsymbol{x} - \boldsymbol{y}\|.$$

Finally, in the next theorem, we prove the linear convergence.

Theorem 5.4.1. *Suppose that Assumption 5.4.1 - 5.4.4 hold. Let $\{w\}$ be a sequence generated by Algorithm 5 and w_* be the optimal point. Then there exists $0 < \xi < 1$, with probability at least $1 - 2\exp(-m)$, we have*

$$f(w_{t+1}) - f(w_*) \leq \xi \; (f(w_t) - f(w_*)),$$

where

$$\xi = \left(1 - 4\beta(1-\beta)\frac{m\lambda^2 \rho_t}{L_g(\mathcal{C}dL_g^2 + m\lambda\rho_t)}\right).$$

Proof. Since ∇f is Lipschitz continuous, we have

$$\begin{aligned} f(w_{t+1}) &\leq f(w_t) + g_t^\top (w_{t+1} - w_t) + \frac{L_g}{2}\|w_{t+1} - w_t\|^2 \\ &= f(w_t) + \eta_t g_t^\top p_t + \frac{\eta_t^2 L_g}{2}\|p_t\|^2. \end{aligned}$$

Let $u^2 = -g_t^\top p_t$, then using Lemma 5.4.2, we have $\|p_t\|^2 \leq -\frac{g_t^\top p_t}{\rho_t} = \frac{u^2}{\rho_t}$, and we get

$$\begin{aligned} f(w_{t+1}) &\leq f(w_t) + \eta_t g_t^\top p_t + \frac{\eta_t^2 L_g}{2}\|p_t\|^2 \\ &\leq f(w_t) + \eta_t(-u^2) + \frac{\eta_t^2 L_g}{2\rho_t}u^2 \\ &= f(w_t) - \eta_t\left(1 - \frac{\eta_t L_g}{2\rho_t}\right)u^2. \end{aligned}$$

Hence the exit condition of backtracking line search $f(w_t + \eta_t p_t) \leq f(w_t) + \beta\eta_t g_t^\top p_t$ satisfies if we take

$$\left(1 - \frac{\eta_t L_g}{2\rho_t}\right) = \beta,$$

and step size $\eta_t = 2(1-\beta)\rho_t/L_g$. Therefore, it stops when $\eta_t \geq 2\rho_t/L_g$ and we have

$$f(w_{t+1}) \leq f(w_t) - 2\beta(1-\beta)\frac{\rho_t}{L_g}u^2. \tag{5.4.4}$$

Since $u^2 = -g_t^\top p_t$, and from Lemma 5.4.1,

$$u^2 = -g_t^\top p_t = g_t^\top (N_t + \rho_t I)^{-1} g_t \geq \frac{m\lambda}{\mathcal{C}dL_g^2 + m\lambda\rho_t}\|g_t\|^2.$$

Hence, by (5.4.4) we get

$$f(w_{t+1}) \leq f(w_t) - 2\beta(1-\beta)\frac{m\lambda\rho_t}{L_g(\mathcal{C}dL_g^2 + m\lambda\rho_t)}\|g_t\|^2. \tag{5.4.5}$$

Subtracting $f(w_*)$ from both sides of (5.4.5), and from strong convexity of f, we have $\|g_t\|^2 \geq 2\lambda(f(w_t) - f(w_*))$, which implies

$$\begin{aligned} f(w_{t+1}) - f(w_*) &\leq f(w_t) - f(w_*) - 4\beta(1-\beta)\frac{m\lambda^2\rho_t}{L_g(\mathcal{C}dL_g^2 + m\lambda\rho_t)}(f(w_t) - f(w_*)) \\ &= \left(1 - 4\beta(1-\beta)\frac{m\lambda^2\rho_t}{L_g(\mathcal{C}dL_g^2 + m\lambda\rho_t)}\right)(f(w_t) - f(w_*)). \end{aligned}$$

This completes the proof. ∎

5.5 Stochastic variant of the regularized Nyström gradient method

In this section, we discuss the stochastic variant of the Nyström gradient. In the context of machine learning, it is usual to work with a large number of samples, making it computationally challenging to compute the full gradient at every iteration. To address this challenge, we employ the stochastic gradient with the Nyström approximation. In this stochastic variant of the NGD, given a mini-batch $\mathcal{B}$, we compute the stochastic gradient $g_{t,\tau} = \nabla f_{\mathcal{B}}(w_t)$ for t^{th} iteration of τ^{th} epoch and compute the regularized Nyström $N_\tau + \rho_\tau I$, once per epoch with regularized parameter $\rho_\tau = c_1\|g_{t-1,\tau}\|^\gamma$ [9]. We compute the search direction with ρ_τ as follows:

$$p_{t-1} = (N_\tau + \rho_\tau I)^{-1} g_{t-1,\tau} = \frac{1}{\rho_\tau} g_{t-1,\tau} - Q_\tau Z_\tau^\top g_{t-1,\tau}, \tag{5.5.1}$$

where $Q_\tau = \frac{1}{\rho_\tau^2} Z_\tau (I + \frac{1}{\rho_\tau} Z_\tau^T Z_\tau)^{-1}$ and Z_τ similar to (5.2.6). Furthermore, we use the diminishing step size η_t for the stochastic variant. Next, we present the Algorithm 6 for

[9]that the regularizer ρ_τ is stochastic gradient and not full gradient. We update the ρ_τ in the beginning of the epoch τ, with $\nabla f_{\mathcal{B}}(w_t)$ of $(\tau - 1)$th epoch.

a stochastic variant of a regularized Nyström method and we call this variant NSGD.

Table 5.2: Search direction and γ in for NSGD

Proposed methods	Regularizer ρ (Value of γ)	Search direction
NSGD	$\rho_\tau = c_1\|g_{t-1,\tau}\|^\gamma$ ($\gamma = 1/2$)	$p_{t-1} = (N_\tau + \rho_\tau)^{-1} g_{t-1,\tau}$

Algorithm 6 A regularized Nyström with stochastic gradient: NSGD Algorithm

Parameters: Update frequency ℓ and initial step size η_0

```
1: Initialize $w_0, \tau = 1$
2: for $t = 1, 2, \ldots$ do
3:     randomly pick batch $\mathcal{B} \sim \{1, \ldots, n\}$
4:     $g_{t\text{-}1,\tau} = \nabla f_{\mathcal{B}}(w_{t\text{-}1})$
5:     if $(t-1)$ mod $\ell = 0$ then
6:         randomly pick indices set $\Omega \subseteq \{1, 2, \ldots, d\}$ such that $m = |\Omega|$
7:         compute $C_\tau$ ($\Omega$ columns of the Hessian) at $w_{t-1}$
8:         compute $Z_\tau$ similar to (5.2.6) and compute $\rho_\tau$
9:         $Q_\tau = \frac{1}{\rho_\tau^2} Z_\tau (I + \frac{1}{\rho_\tau} Z_\tau^T Z_\tau)^{\text{-}1}$
10:        $\tau = \tau + 1$
11:    end if
12:    Compute $p_{t-1}$ using (5.5.1)
13:    $w_t = w_{t-1} - \eta_t p_{t-1}$
14: end for
```

Remark 5.5.1. *Note that $\rho_\tau = 0$ even when full gradient is not zero. Thus, we use $\rho_\tau = \max(c_4, \|g_{t-1,\tau}\|)$, where c_4 is some small positive constant.*

5.6 Non-convex case

In this section, we consider the non-convex case of the objective function. While there is a significant amount of research on developing low-rank Hessian approximated methods for convex function [78, 87, 121, 52], there has been less focus on non-convex function [120, 48]. Standard Nyström method conventionally targets symmetric positive semi-definite (SPSD) matrix, but it encounters an issue in the non-convex domain as non-convex function does not produce the SPSD Hessian. In order to tackle this issue, we incorporate an additional term to tackle the negative eigenvalues of Hessian. Featuring an additional term helps to obtain a SPSD matrix with a fixed-rank regularized Nyström approximation.

5.6.1 Nyström approximation for non-convex function and algorithm

Now consider a ℓ_2-regularized objective function (5.2.7)

$$f(w) = \sum_{i=1}^{n} f_i(w) + \frac{\lambda}{2}\|w\|^2, \tag{5.6.1}$$

where each f_i is twice continuously differentiable and non-convex and $\lambda > 0$. Let $H^f = \sum_{i=1}^{n} \nabla^2 f_i(w)$ and $S \in \mathbb{R}^{d\times m}$ be a zero-one random matrix similar to (5.2.4).

$$C = HS = \nabla^2 f(w)S = H^f S + \lambda S \tag{5.6.2}$$

$$M = S^\top HS = S^\top \left(H^f + \lambda I\right) S = S^\top H^f S + \lambda I.$$

Note that $S^\top S = I_d$. Since H^f is the Hessian of a non-convex function, the eigenvalues of $S^\top H^f S$ can be non-positive. In order to tackle this issue, we use the regularization term $\mu = c_2 \max(0, -\lambda_{\min}(S^\top H^f S))$, and $c_2 > 0$.

$$\begin{aligned} M + \mu I &= S^\top HS + c_2 \max(0, -\lambda_{\min}(S^\top H^f S)) \\ &= S^\top H^f S + c_2 \max(0, -\lambda_{\min}(S^\top H^f S)) + \lambda I, \end{aligned} \tag{5.6.3}$$

and we get positive definite matrix $M + \mu I$ and $\lambda_{\min}(M + \mu I) = \lambda$. Finally, the fixed rank Nyström-type approximation is given by

$$N = (C(M + \mu I)^{-1} C^\top).$$

Since the minimum eigenvalue of $M + \mu I$ is λ, N is a fixed-rank Nyström-type SPSD low-rank approximation. Moreover, $(M + \mu I)^{-1}$ is always an $m \times m$ matrix, and m is the fixed rank, which is a user-defined parameter. Moreover, we need a non-singular matrix to compute the search direction. We compute $\rho_t = c_1\|g_t\|^\gamma$, where γ can be 1/2, 1 or 2 and $c_1 > 0$.

$$\begin{aligned} B &= N + \rho I \\ &= C(M + \mu I)^{-1} C^\top + c_1\|g_t\|^\gamma I \\ &= C(M + \mu I)^{-1} C^\top + \rho_t I. \end{aligned} \tag{5.6.4}$$

To obtain the inverse of B, we apply the Sherman-Morrison-Woodbury formula for t^{th} iteration. We obtain the following search direction:

$$\begin{aligned} p_t &= -B_t^{-1} g_t \\ &= -(N_t + \rho I)^{-1} g_t \\ &= -\frac{1}{\rho_t} g_t + \frac{1}{\rho_t^2} Q_t C_t^\top g_t, \end{aligned} \tag{5.6.5}$$

where $Q_t = C_t(M_t + \mu_t I + \frac{1}{\rho_t} C_t^\top C_t)^{-1}$.

As we mentioned in the section on strongly convex, (5.3.1) is similar to RL-BFGS-SW given in Chapter 3. In Chapter 3, we used the strong Wolfe condition, and here we used backtracking line search.

Next, we demonstrate the algorithm of regularized Nyström method 7 as fixed rank regularized Nystrŏm for non-convex function.

Remark 5.6.1. *It is worth mentioning that calculating the eigenvalues of $S^\top H^f S$ is affordable because it is an $m \times m$ matrix and m is much smaller than d. Also, inverting matrix $M + \mu I$ is not a burden as it is a $m \times m$ matrix.*

Algorithm 7 A regularized Nyström for non-convex function

```
1: Initialize Initial parameters $w_0$, desired rank $|\Omega| = m$, $\alpha, \beta \in (0,1)$, and maximum
   iterations $t_{\max}$
2: $t \leftarrow 0$
3: repeat
4:     $g_t = \nabla f(w_t)$
5:     randomly pick indices set $\Omega \subseteq \{1, 2, \dots, d\}$ such that $|\Omega| = m$
6:     Compute $M_t$ and obtain $\mu_t$
7:     Compute $Q_t$ search direction using (5.6.5),
           $$p_t = -(N_t + \rho_t I)^{-1} g_t$$
       .
8:     Compute step size using Armijo's rule: Find least positive integer $\ell_t$ such that
           $$f(w_t + \alpha^{(\ell_t)} p_t) \le f(w_t) + \alpha^{(\ell_t)} \beta g_t^\top p_t$$
9:     Set $\eta_t = \alpha^{(\ell_t)}$, $w_{t+1} = w_t + \eta_t p_t$, and $t = t + 1$.
10: until $t = t_{\max}$ or some termination criteria is satisfied
```

5.6.2 Global convergence

In this section, we provide the global convergence analysis. First, we need following assumptions.

Assumption 5.6.1. *i) The objective function* (5.2.7) *is twice continuously differentiable.*
ii) Let w_0 be an initial point and the level set of the objective function $\Gamma := \{w \in \mathbb{R}^d : f(w) \leq f(w_0)\}$ is compact and $\{w\} \in \Gamma$.
iii) There exists a minimum $f_{\min}$ of f.

Since we are using Armijo's backtracking rule to have $f(w_{t+1}) \leq f(w_t)$, for any $t \in \mathbb{N}$, implies that the sequence $\{w_t\}$ generated by the proposed algorithm 7 is included in the level set Γ. Similarly, from Assumption 5.6.1(i) and (ii), there exists $L_g > 0$ such that for all $w \in \Gamma$,

$$\|\nabla^2 f(w_t)\| \leq L_g, \quad \forall t \in \Gamma. \tag{5.6.6}$$

Moreover, from Assumption 5.6.1(ii) it follows that there exists $U_g > 0$ such that

$$\|g_t\| \leq U_g, \quad \forall t \geq 0, \tag{5.6.7}$$

and assume that there exists $\varepsilon > 0$ such that $\varepsilon \leq \|g_t\|$. Note that one can always assume $\varepsilon > 0$,

$$\varepsilon \leq \|g_t\|, \tag{5.6.8}$$

when $w_t \neq w_*$, where w_* is the optimal point.

Lemma 5.6.1. *Suppose that Assumption 5.4.2 and 5.6.1 hold. Let w_t iterate obtained by Algorithm 5, and for some m, the maximum and minimum eigenvalues of $(N_t + \rho_t I)^{-1}$ are bounded*

$$\lambda_{\min}[(N_t + \rho_t I)^{-1}] \geq \frac{\lambda m}{L_g^2 \mathcal{C} d + m\lambda c_1 U_g^\gamma}, \quad \textit{and} \quad \lambda_{\max}[(N_t + \rho_t I)^{-1}] \leq \frac{1}{c_1 \varepsilon^\gamma}. \tag{5.6.9}$$

Proof. This Lemma can be prove similar to Lemma 5.4.1, and from assumptions it is easy to see that

$$\begin{aligned}\lambda_{\min}[(N_t + \rho_t I)^{-1}] &= \frac{1}{\frac{\mathcal{C} L_g^2 d}{m\lambda} + c_1\|g_t\|^\gamma} \\ &\geq \frac{\lambda m}{L_g^2 \mathcal{C} d + m\lambda c_1 U_g^\gamma},\end{aligned}$$

and similarly,

$$\lambda_{\max}[(N_t + \rho_t I)^{-1}] = \frac{1}{c_1\|g_t\|^\gamma}$$

$$\leq \frac{1}{c_1 \varepsilon^{\gamma}}.$$

Since N_t is positive semi-definite ρ_t is *minimum* eigenvalue of $(N_t + \rho_t I)$. This completes the proof. ■

Lemma 5.6.2. *Suppose that Assumption 5.6.1 holds. Assume that there exists* $\varepsilon > 0$ *such that* $\varepsilon \leq \|g_t\|$. *Then the search direction* p_t *satisfies*

$$\|p_t\| \leq b(\varepsilon), \tag{5.6.10}$$

where

$$b(\varepsilon) := c_4 \max\left(U_g^{1-\gamma}, \frac{1}{\varepsilon^{\gamma-1}}\right).$$

Proof. From (5.6.7), we have $\|g_t\| \leq U_g$ and $\varepsilon \leq \|g_t\|$. We prove this with two cases of of γ.

- **case 1:** $\gamma \leq 1$. In this case, the computing upper bound of search direction p_t, we have

$$\begin{aligned}\|p_t\| &\leq \frac{\|g_t\|}{\rho_t} \\ &= \frac{\|g_t\|}{c_1 \|g_t\|^{\gamma}}, \\ &= c_4 \|g_t\|^{1-\gamma} \\ &\leq c_4 U_g^{1-\gamma},\end{aligned}$$

  where $c_4 = 1/c_1$.

- **case 2:** $\gamma > 1$, implies $1 - \gamma < 0$, and hence we get

$$\begin{aligned}\|p_t\| &\leq c_4 \|g_t\|^{1-\gamma} \\ &\leq \frac{c_4}{\varepsilon^{\gamma-1}}.\end{aligned}$$

It follows from above cases that

$$\|p_t\| \leq c_4 \max\left(U_g^{1-\gamma}, \frac{1}{\varepsilon^{\gamma-1}}\right). \tag{5.6.11}$$

This completes the proof. ■

From above Lemma, we have $w_t + \tau p_t \in \Gamma + B(0, b(\varepsilon)),\ \forall \tau \in [0, 1]$. The compactness of

$\Gamma + B(0, b(\varepsilon))$ and f is twice continuously differentiable, it follows that there exists $U_H > 0$ such that

$$\|\nabla^2 f(w)\| \leq U_H, \quad \forall w \in \Gamma + B(0, b(\varepsilon)). \tag{5.6.12}$$

Next, we obtain a step size that is related to a constant that satisfies the Armijo's rule.

Lemma 5.6.3. *Suppose that Assumption 5.4.2, 5.4.3, and 5.6.1 hold, and there exists $\varepsilon > 0$ such that $\|g_t\| \geq \varepsilon$. Then, the step size $\eta_t > 0$*

$$\eta_t \leq \frac{2(1-\beta)\lambda m c_1^2 \varepsilon^{2\gamma}}{(U_H^2 c_3 d + m\lambda c_1 U_g^\gamma) U_H}, \tag{5.6.13}$$

satisfies Armijo's rule (5.3.4)*.*

Proof. Since f is twice continuously differentiable, we consider a 2nd order Taylor's theorem, there exists $\tau_t \in [0,1]$ such that

$$f(w_t + \eta_t p_t) = f(w_t) + \eta_t g_t^\top p_t + \frac{1}{2}\eta_t^2 p_t^\top \nabla^2 f(w_t + \tau_t \eta_t p_t) p_t.$$

Adding $\beta \eta_t g_t^\top p_t$ both side and rearranging above equation, computing p_t and using Lemma 5.6.1, we get

$$\begin{aligned}
& f(w_t) - f(w_t + \eta_t p_t) + \beta \eta_t g_t^\top p_t \\
&= (\beta - 1)\eta_t g_t^\top p_t - \frac{1}{2}\eta_t^2 p_t^\top \nabla^2 f(w_t + \tau_t \eta_t p_t) p_t \\
&= \frac{\eta_t U_H}{2 c_1^2 \varepsilon^{2\gamma}} \left(\frac{2(1-\beta)\lambda m c_1^2 \varepsilon^{2\gamma}}{(U_H^2 c_3 d + m\lambda c_1 U_g^\gamma) U_H} - \eta_t \right) \|g_t\|^2 \\
&\geq 0.
\end{aligned}$$

■

Lemma 5.6.4. *Suppose that Assumption 5.4.2, 5.4.3, and 5.6.1 hold and there exists $\varepsilon > 0$ such that $\|g_t\| \geq \varepsilon$. Then the step size η_t satisfies the lower bound*

$$\eta_t \geq \eta_{\min}(\varepsilon), \tag{5.6.14}$$

where

$$\eta_{\min}(\varepsilon) = \min\left(1, \frac{2(1-\beta)\alpha\lambda m c_1^2 \varepsilon^{2\gamma}}{(U_H^2 c_3 d + m\lambda c_1 U_g^\gamma) U_H}\right).$$

Proof. Proof can be establish using previous Lemma 5.6.3 and Armijo's rule. ■

In the next lemma, when $f(w_t) \neq f_{\min}$, we provide a lower bound on the reduction in the difference between two consecutive values of f.

Lemma 5.6.5. *Suppose that Assumption 5.4.2, 5.4.3, and 5.6.1 hold and there exists $\varepsilon > 0$ such that $\|g_t\| \geq \varepsilon$, then with probability at least $1 - 2\exp(-m)$, we have*

$$f(w_t) - f(w_{t+1}) \geq U_1 \varepsilon^2,$$

where

$$U_1 := \frac{\beta \eta_{\min}(\varepsilon) \lambda m}{U_H^2 c_3 d + m\lambda c_1 U_g^{\gamma}}.$$

Proof. From Armijo's rule and using previous Lemma 5.6.4, it follows that

$$\begin{aligned} f(w_t) - f(w_{t+1}) &\geq -\beta \eta_t g_t^\top p_t \\ &= \beta \eta_t g_t^\top (N_t + \rho_t I)^{-1} g_t \\ &\geq \beta \eta_{\min}(\varepsilon) \lambda_{min}[(N_t + \rho_t I)^{-1}] \|g_t\|^2 \\ &= U_1 \varepsilon^2. \end{aligned} \tag{5.6.15}$$

This completes the proof. ∎

Theorem 5.6.1. *Suppose the Assumption 5.4.2, 5.4.3, and 5.6.1 hold. Then with the probability at least $1 - 2t\exp(-m)$,*

$$\liminf_{t\to\infty} \|g_t\| = 0,$$

or there exists a $t > 0$, such that $\|g_t\| = 0$.

Proof. Since iterate is similar to the Chapter 3, the proof of this theorem can be given in a similar way to the Theorem 3.4.1. ∎

Further theoretical properties are explored in the Appendix, specially dedicated to this chapter. Appendix is structured into the three sections. In the initial section, we discuss the theorem of exact Nyström. The subsequent section focuses on establishing an upper bound on the distance between the search direction of Newton's method and NGDs. Finally, the third section provides an upper bound in terms of the closeness of proposed approximation to the regularized Hessian inverse.

5.7 Numerical experiments

In this section, we demonstrate the numerical results for the proposed algorithms explained in the previous sections.

First, we discuss the experiment setup for the numerical experiments. We performed Figure experiments on MATLAB R2018a on Intel(R) Xeon(R) CPU E7-8890 v4 @ 2.20GHz with 96 cores and Figure on MATLAB R2019a on Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz with 32 cores. We implemented the existing and proposed methods in MATLAB using the SGDLibrary [67]. We solve standard learning problems, that is, the ℓ_2-logistic regression:

$$\min_w F(w) = \frac{1}{n}\sum_{i=1}^{n} \log\left[1+\exp(-b_i a_i^T w)\right] + \frac{\lambda}{2}\|w\|^2,$$

where $a_i \in \mathbb{R}^d$ is feature vector and $b_i \in \{\pm 1\}$ is target label of the i-th sample, and λ is a ℓ_2 regularizer. We evaluated the numerical experiments on benchmark datasets given in Table 5.3. The datasets are binary classification problems, and all datasets are available on LIBSVM [24]. We demonstrate the performance of the proposed and existing methods on the ℓ_2-regularized logistic regression problem. We optimize the constant c_1 in regularizer $\rho_t = c_1\|g_t\|^\gamma$ using a grid search $c_1 \in \{10^0, 10^{-1}, 10^{-2}, 10^{-3}\}$. For each method, the best-performing model was selected based on the minimum cost error of the training.

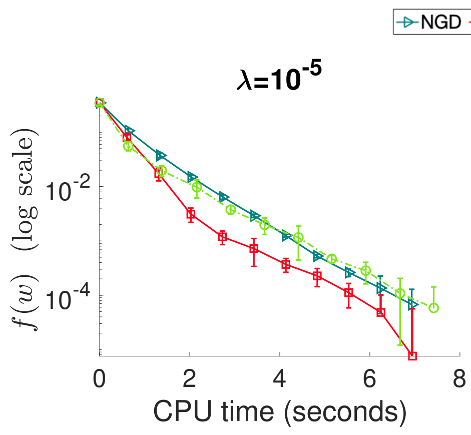
Table 5.3: Details of the datasets used in the experiments

Dataset	**Dim**	**Train**	**Test**	Density
adult[1]	$123+1$	32,561	16,281	0.1128
gisette[1]	$5,000+1$	6,000	1,000	0.9910
epsilon[1]	$2,000+1$	50,000	50,000	1
real-sim[1]	$20,958+1$	57,909	14,400	0.0024
w8a[1]	$300+1$	49,749	14,951	0.0388

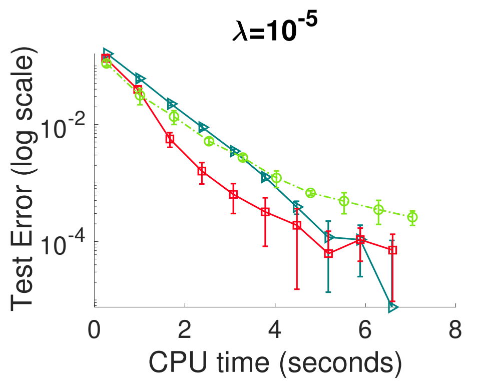
First we study the performance of NGD, NGD1 and NGD2 to see the behaviour of different $\rho = c_1\|g_t\|^\gamma$, where $\gamma = 1/2$ for NGD, $\gamma = 1$ for NGD1, and $\gamma = 2$ for NGD2. We computed the ℓ_2-regularized *logistic regression* with $\lambda = 10^{-5}$ on the *ijcnn1* and *adult* datasets.

Figure 5.1 shows the training cost and test error with CPU time for *adult* and *ijcnn1*. Moreover, it shows that NGD1 outperforms NGD and NGD2 for *adult* dataset, and NGD outperforms NGD1 and NGD2 for *ijcnn1* dataset. Therefore, in the next subsection, we consider NGD and NGD1 to compare the behavior with varying numbers of selected columns.

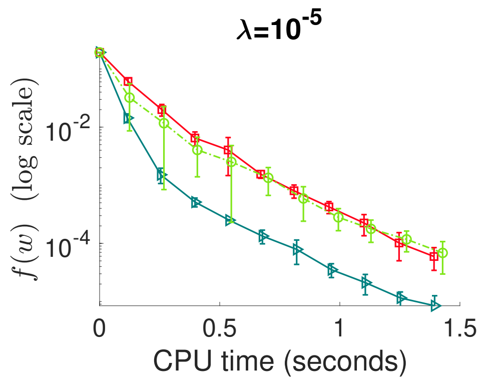
[1]Available at LIBSVM [24] `https://www.csie.ntu.edu.tw/cjlin/libsvm/`



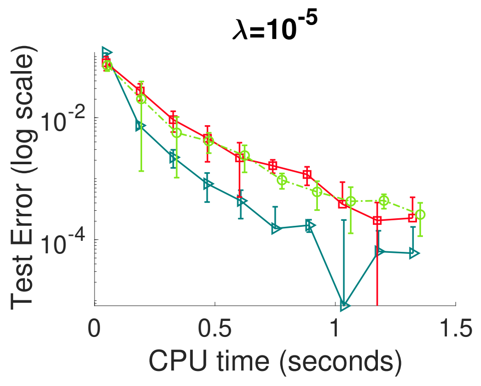
(a) *adult*



(b) *adult*



(c) *ijcnn1*



(d) *ijcnn1*

Figure 5.1: First two figures(from left) shows the experiments on *adult* for $m = 25$ and last two figures shows the experiments on *ijcnn1* for $m = 5$. (a) and (c) shows the cost with respect to CPU time. (b) and (d) shows the test error with respect to CPU time.

5.7.1 Comparison of strength for varying numbers of selected columns

In this subsection, we demonstrate the comparison of various sketch sizes (no. of selected columns) for high-density data *gisette* and sparse data *w8a* on *logistic regression* with $\lambda = 10^{-5}$ to observe the robustness of the proposed methods. We keep the same c_1 in ρ_t for each dataset to compare the different numbers of selected columns. Figure 5.2 shows the numerical performance for the *gisette* dataset and computed the NGD1 for $m = 50(1\%), 250(5\%), 500(10\%)$ and $m = 1000(20\%)$. As shown in Figure 5.2, due to the high density, only $m = 250(5\%)$ of columns are sufficient to get the minimum value of the objective function within the comparative CPU time. Also, similar behavior can be observed in the test error as well. When $m = 1000$, the decrement in the value of the gradient norm surpasses all cases of $m < 1000$. Additionally, $m = 250$ and $m = 500$ perform a similar reduction in the value of the norm of the gradient.

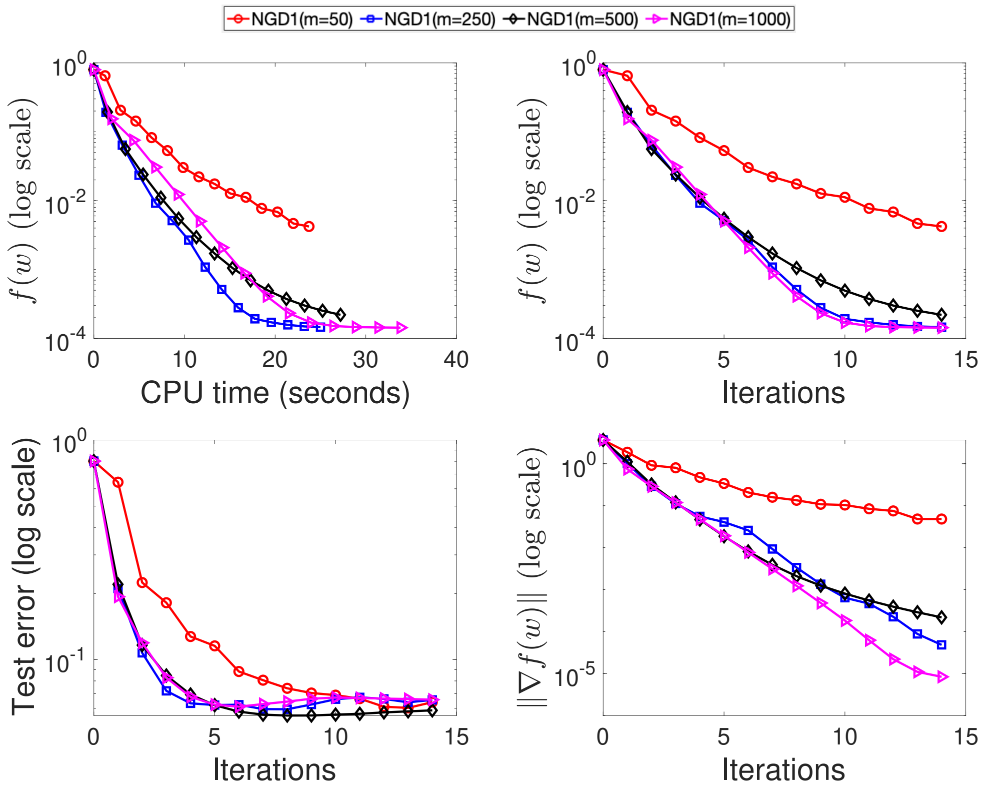


Figure 5.2: Column comparison on *gisette* dataset

Figure 5.3 shows the numerical performance on the *w8a* dataset and computed NGD for $m = 30(10\%), 60(20\%), 100(33\%)$ and $m = 150(50\%)$. Figure 5.3 shows that due to sparse data, it requires picking more number columns to obtain the minimum value of the objective function in the comparative CPU time. All cases of m exhibit almost similar test errors. When $m = 150$ and $m = 100$, the decrement in the value of the norm of gradient is comparable, whereas for the cases $m = 30$ and $m = 60$, it does not decrease significantly.

5.7.2 Comparison with randomized subspace Newton

In this subsection, we compare the NGDs with the randomized subspace Newton (RSN) [52]. RSN computes the iterate with $w_{t+1} = w_t - (1/L)S_t(S_t^\top H_t S_t)^\dagger S_t^\top g_t$ with a sketch matrix $S_t \in \mathbb{R}^{d\times m}$. To have a fair comparison of the subspace Newton, we compute the RSN with Armijo's rule with backtracking line search (instead of $1/L$) and compute the RSN exactly as given in [52, definition 4] for generalized linear models. Also, we keep the same value of m for both NGDs and RSN. We compute the *logistic regression* with $\lambda = 10^{-5}$. We compare NGDs and RSN in Figure 5.4 for *realsim* data with $m = 2000$, Figure 5.5 for *gisette* data with $m = 250$, and Figure 5.6 for *w8a* data with $m = 30$. As shown in Figure 5.4 to 5.6, RSN is unable to outperform the proposed methods. For the *realsim* data, as shown in Figure 5.4, NGD1 outperforms all methods in terms of achieving the minimum

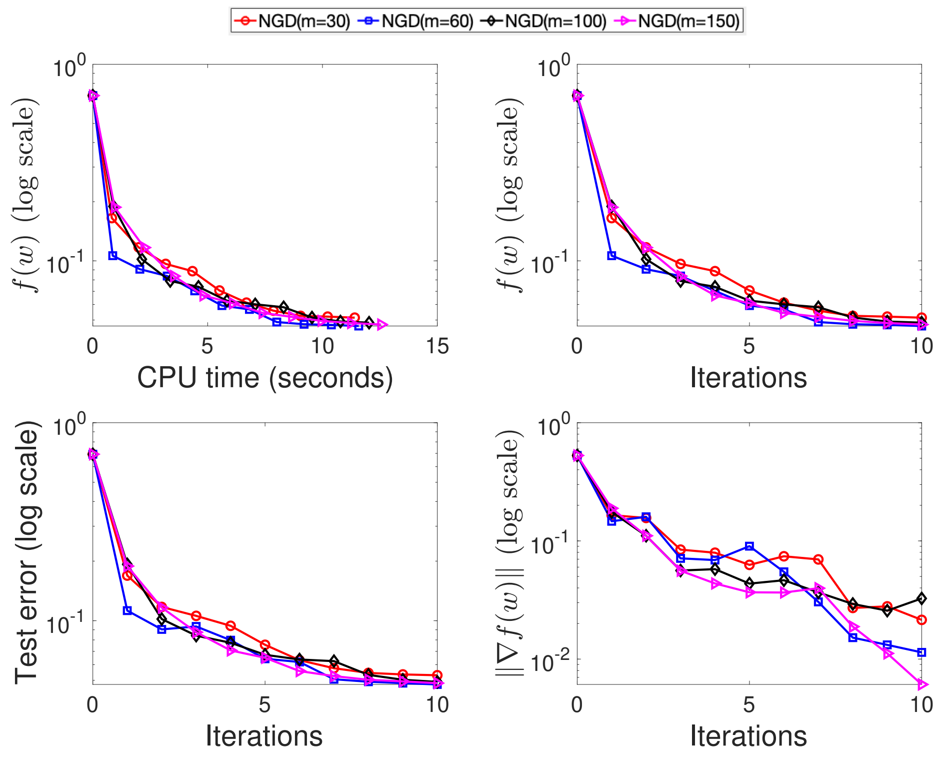


Figure 5.3: Column comparison on *w8a* dataset

cost, and NGD2 outperforms all methods in terms of test error. For the *gisette* data, in Figure 5.5, NGD1 outperforms all methods in terms of achieving minimum cost and test error. Finally, for the *w8a* dataset, Figure 5.6, NGD outperforms all methods in terms of achieving minimum cost, and NGD1 outperforms at the later stage in terms of test error. In conclusion, it is observable that the Nyström approximation is better than the approximation of RSN because RSN only captures a limited set of m^2 elements from the Hessian, whereas Nyström captures a substantially larger set of dm elements of the Hessian. This makes the Nyström approximation more comprehensive and accurate of the Hessian matrix.

5.7.3 Comparison of Newton Sketch and Nyström approximation

In this subsection, we compare the NGDs with the Newton sketch(NS) [93]. As explained in Section 5.3.2, the proposed method can be represented as the NS method with certain structure modifications. Hence, we compare the raw Nyström with the Hessian approximation of NS in terms of closeness with the Hessian. NS computes the Hessian approximation as $(\nabla^2 f(w)^{1/2})^\top P^\top P(\nabla^2 f(w)^{1/2})$, and it is important to note that the $P \in \mathbb{R}^{m\times n}$, where n is the number of samples and m is the sketch size. In this comparison, we keep the same value of m for both Nyström and NS. In Figure 5.7 we conduct numerical experiments on *w8a, realsim* and *gisette* datasets. We provide a comparison of the norm

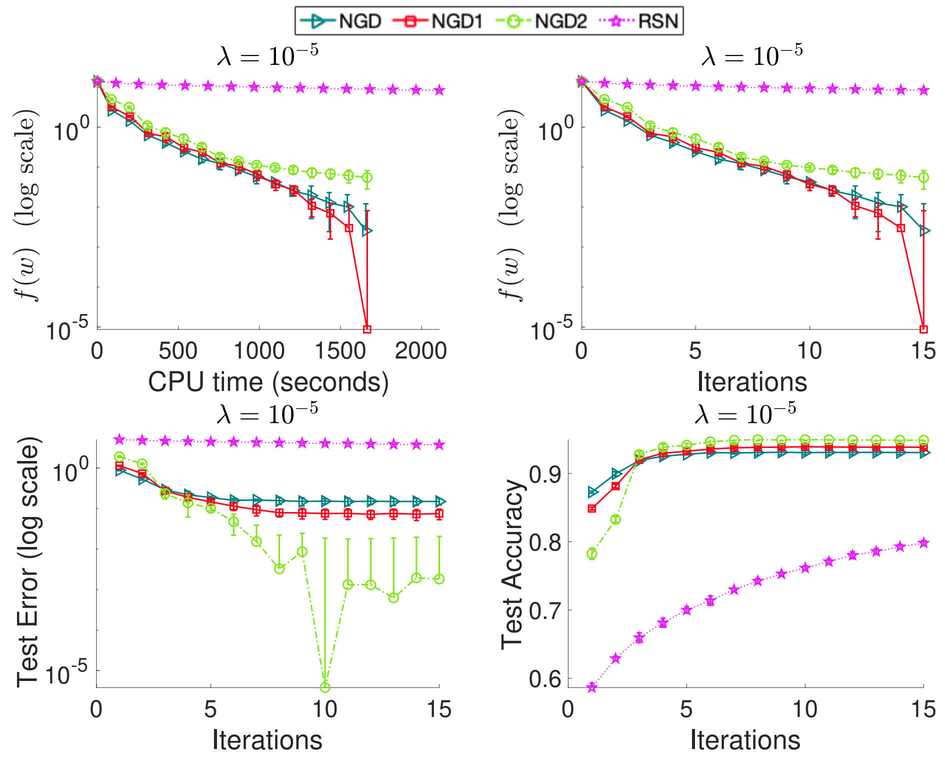


Figure 5.4: Comparison with RSN for *realsim* dataset with $m = 2000$

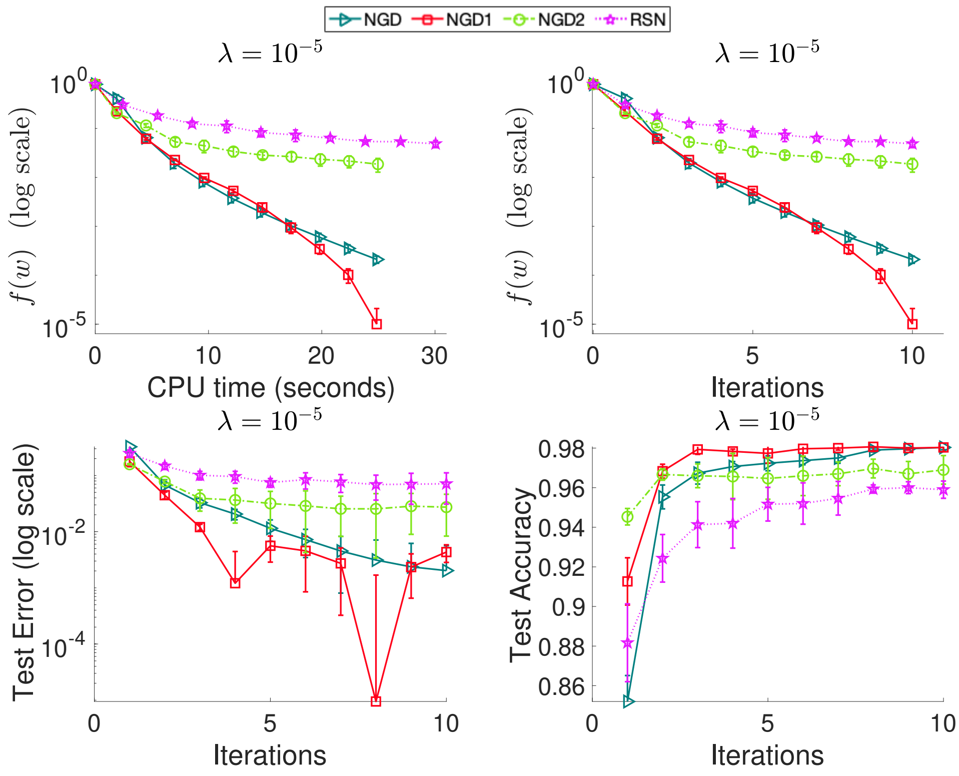


Figure 5.5: Comparison with RSN for *gisette* dataset with $m = 250$

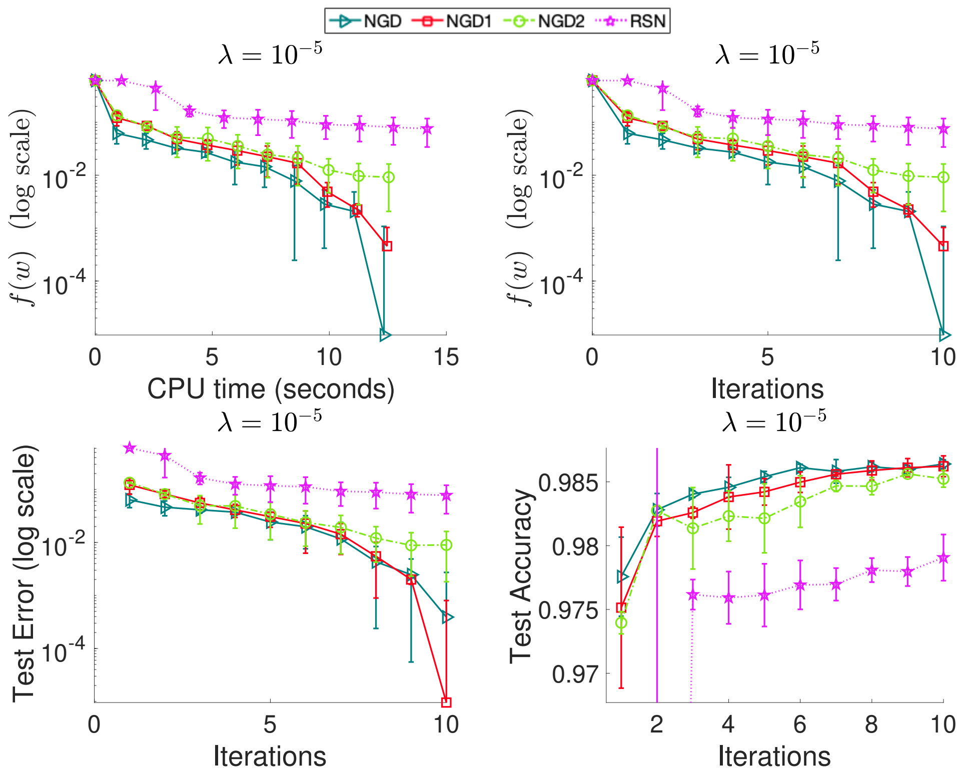


Figure 5.6: Comparison with RSN for *w8a* dataset with $m = 30$

difference with Hessian and its CPU time as m increases. We conduct these numerical experiments on the *logistic regression*. It is important to note that we use the *logistic regression* for the *realsim,* and *gisette* without ℓ_2 regularization. For the *w8a* dataset, we keep the ℓ_2-regularized *logistic regression* with $\lambda = 10^{-5}$. Hence, the rank of H for *w8a* data is full. In Figure 5.7, (a) and (d) show the performance on the *w8a* dataset, (b) and (e) show the performance on the *realsim* dataset, and (c) and (f) show the performance on the *gisette* dataset. The top row shows the CPU time of computing the Nyström approximation and Newton sketch, and the bottom row shows the distance with Hessian as m increases, where H is the Hessian.

As shown in Figure 5.7 (a) and (d), Nyström approximation outperforms the Newton sketch as m increases with the less CPU time for *w8a* in lesser CPU time compared to NS. Similarly, in Figure 5.7(b) and (e), the Nyström approximation can approach the Hessian as m increases, specifically after $m = 8000$. Since Nyström involves the inverse of the $m \times m$ matrix, it takes more CPU time after $m = 5000$. Whereas in Figure 5.7(c) and (f), the norm of distance between Hessian and Nyström decreases significantly when $m \approx 1000$ and takes more CPU time after $m \approx 1000$ compared to NS. However, we do not need to compute Nyström for large numbers of m, as we have seen in Figure 5.2 and 5.3 that about 5% to 15% of d can give sufficient decrease in the objective function. From the performance

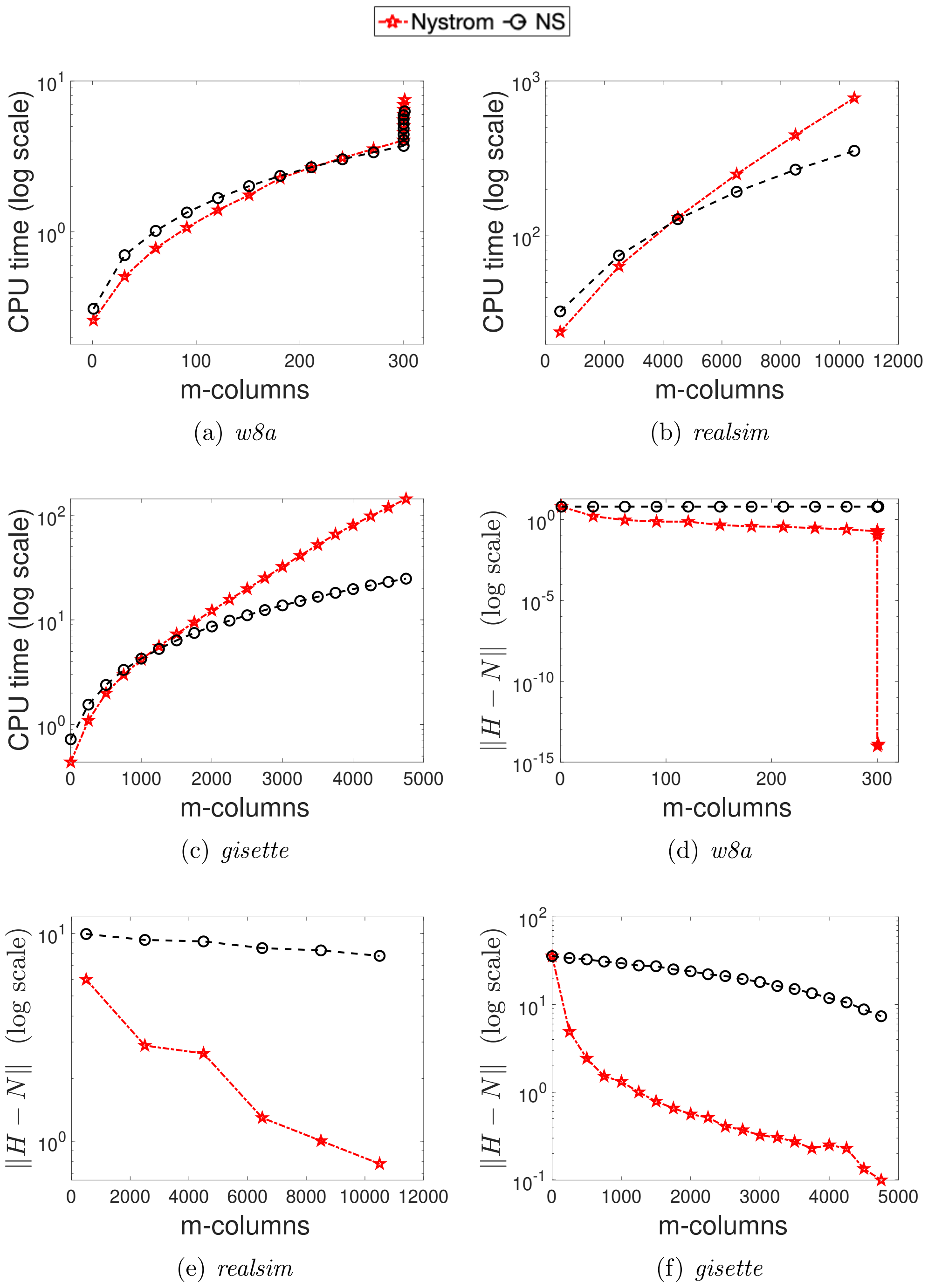


Figure 5.7: Comparison between Nyström and Newton sketch

illustrated in Figure 5.7, two significant observations can be made. Firstly, one can observe in Figure 5.7(d) that the Theorem A.1 pertaining Nyström bound of exactness holds true practically and becomes almost zero as the number of columns m covers all of the columns (*i.e.*, rank of H). Secondly, it is worth noting that the random matrix in the Newton sketch $P \in \mathbb{R}^{m\times n}$ depends on the n which is usually larger than dimension d, whereas the Newton sketch [93] usually requires a thick random matrix as compared to the thin random matrix S of Nyström approximation.

5.7.4 Comparison with existing deterministic methods

We compared the proposed methods NGD, NGD1, and NGD2 with the existing classical first-order gradient descent and the *state of the art* second-order Hessian approximation method L-BFGS method [78]. The memory used in the L-BFGS method was set to 20. We report the training cost on the training dataset and testing set (test error) for each iteration and the CPU time cost per iteration. Also, we show the norm of the gradient with respect to iterations. Figure 5.8 shows the performance of experiments on *logistic regression* with $\lambda = 10^{-5}$ on *giestte* dataset with $m = 500$. As shown in Figure 5.8, NGD1 outperforms all other methods in terms of both CPU time and iterations in terms of both training cost and the norm of gradient. L-BFGS takes more CPU time compared to all variants of NGDs at the cost of 10^{-5}. Also, GD shows improvements after the 20th iteration and outperforms in terms of the test error, and NGD2 shows some increment in the test accuracy after the 30th iteration. In Figure 5.9, we conduct the experiments on *logistic regression* with $\lambda = 10^{-5}$ on *epsilon* dataset with $m = 200$. Figure 5.9 shows that the NGDs are performing almost similarly and outperform the L-BFGS and GD in terms of the training cost, testing error, and test accuracy. Also, NGD and NG1 outperform all of the methods in terms of the norm of the gradient.

5.7.5 Numerical experiments for stochastic regularized Nyström gradient

We compare the proposed stochastic variant NSGD with stochastic gradient descent method and stochastic second order approximation optimization methods, namely, SVRG-LBFGS [73], SVRG-SQN [81], and SQN [16]. The memory used in the L-BFGS method was set to 20, which is a commonly used value [73, 16]. Figure 5.10 shows that NSGD outperforms existing methods in terms of the training cost for the *a8a* dataset. However, it could not achieve a better test error compared to SVRG-SQN and SVRG-LBFGS. Moreover, SVRG-SQN outperforms NSGD and other existing methods in terms of both training cost and test error for *epsilon* dataset.

5.7.6 Numerical experiments for non-convex function

In this subsection, we demonstrate the regularized Nyström method for non-convex function. We compare the proposed method against gradient descent, a regularized randomized

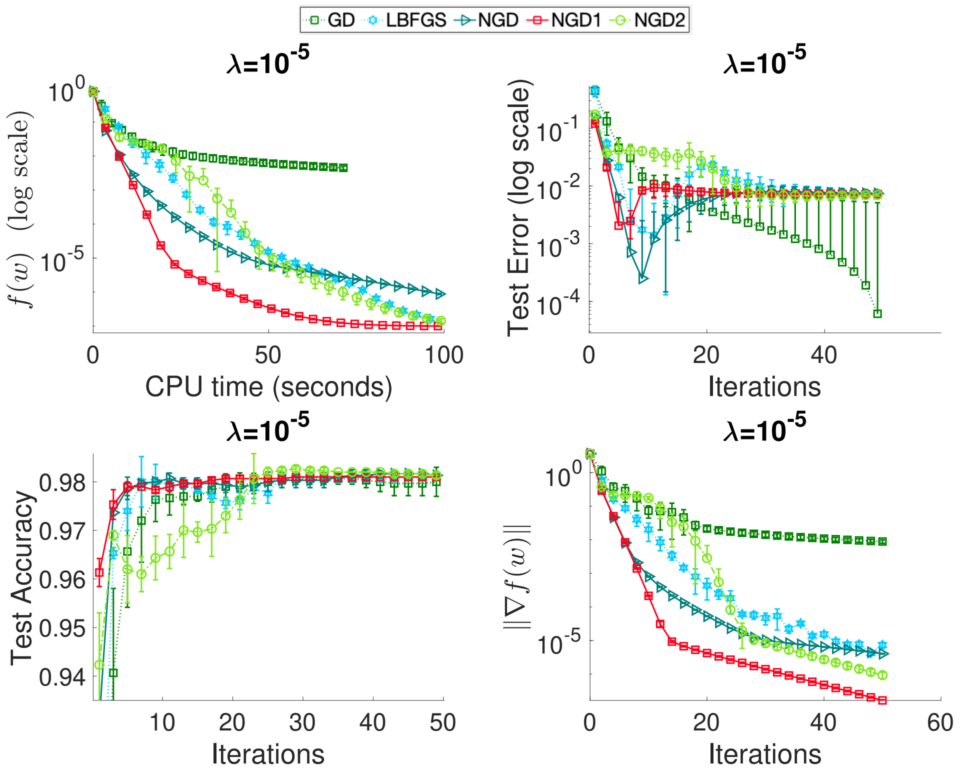


Figure 5.8: Experiments on the *gisette* dataset with $m = 500$.

subspace method(RS-RNM) [48], and regularized Newton method(RNM) [120]. We test these methods on the problem of support vector regression as follows:

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i - x_i^\top w) + \lambda\|w\|^2, \tag{5.7.1}$$

where $(x_i, y_i) \in \mathbb{R}^d \times \{0,1\}, i \in \{1,2,\ldots,n\}$ is training example. We keep ℓ_2 regularized parameter to be 10^{-5}. For the loss function, we use Geman-McClure loss function [4] as follows:

$$\ell(w) = \frac{2w^2}{w^2+4}.$$

We compute the experiments on *gisette* dataset with $c_1 = 1, c_2 = 2$, and randomly pick 500 columns m =*500* per iteration. We keep $\gamma = 1/2$ for all regularized methods, and compute regularized Newton method [120](RNM-GS) with Armijo backtracking line search with search direction:

$$p_t^{(RNM-GS)} = -(\nabla^2 f(w_t) + c_2 \max(0, -\lambda_{\min}(\nabla^2 f(w_t)) + c_1\|\nabla f(w_t)\|^\gamma)\nabla f(w_t),$$

where $\gamma = 1/2$ and GS in RNM-GS represents the gamma-square root. We compute regularized randomized subspace method [48](RS-RNM-GS) with Armijo backtracking line

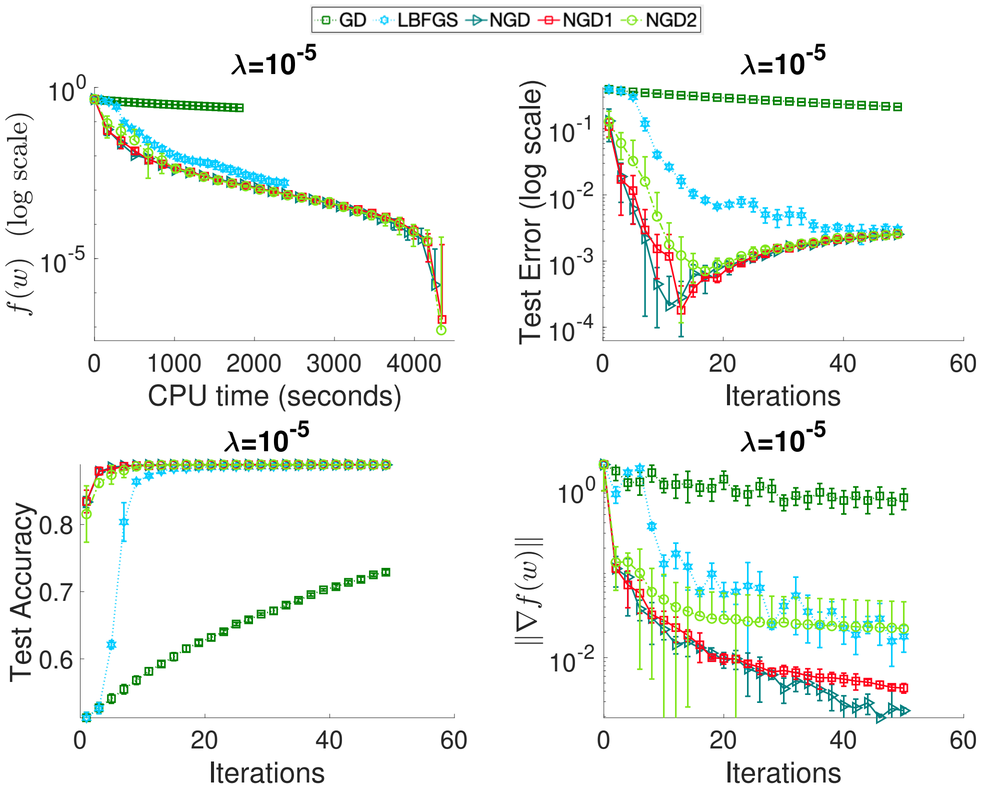


Figure 5.9: Comparison for *epsilon* dataset with $m = 200$

search with search direction:

$$p_t^{(RS-RNM-GS)} = -S_t(S_t^\top H_t S_t + c_2 \max(0, -\lambda_{\min}(S_t^\top \nabla^2 f(w_t) S_t) + c_1 \|\nabla f(w_t)\|^\gamma)^{-1} S_t \nabla f(w_t),$$

where $\gamma = 1/2$ and GS in RS-RNM-GS represents the gamma-square root. For use gradient descent(GD) with Armijo backtracking line search the search direction is given as $p_t^{(GD)} = -\nabla f(w_t)$.

In Figure 5.11, we illustrate experiments conducted on the *gisette* dataset. A comparison in the top left and top right corners reveals that RNM-GS significantly reduces the function value over iterations. However, it consumes considerably more CPU time compared to other methods. In the case of RS-RNM-GS, it demonstrates the ability to optimize the objective function, yet NGD surpasses GD and RS-RNM-GS. Similarly, NGD outperforms other methods, and RS-RNM-GD achieves satisfactory test accuracy. Finally, the norm of the gradient against iterations in the bottom right corner indicates that NGD performs well, being less efficient than RNM-GS but more efficient than RS-RNM-GS.

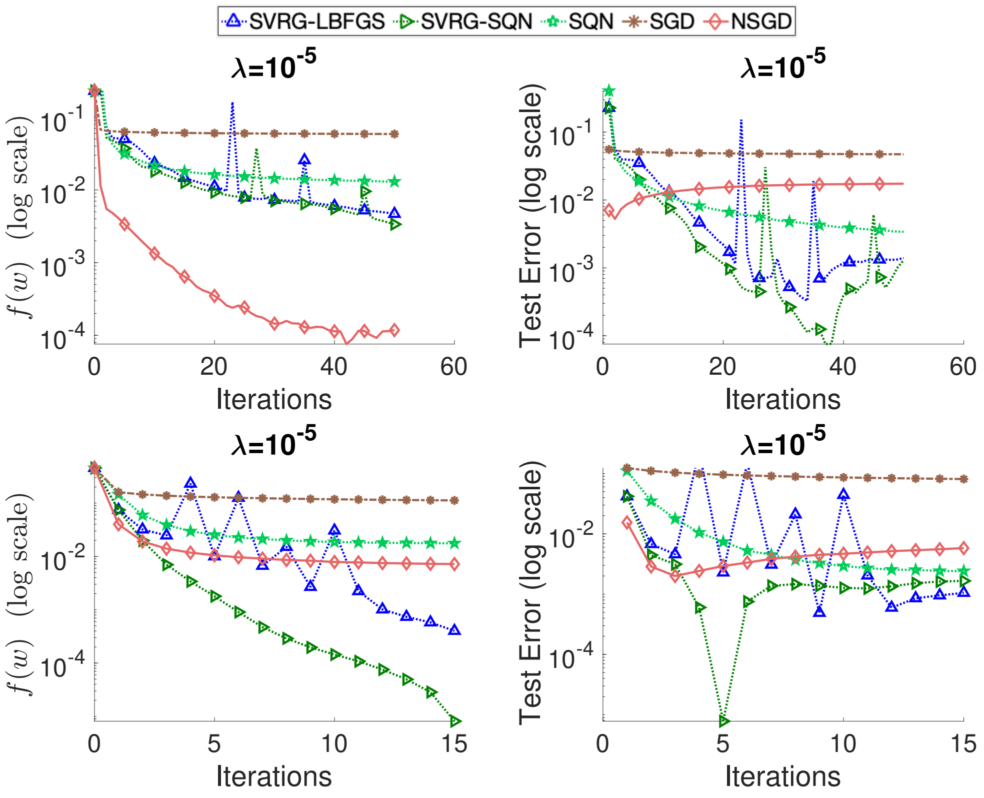


Figure 5.10: First two from left shows the experiments on *a8a* dataset and two from right shows the experiments on *epsilon* dataset

5.8 Application: Tumor detection

Brain MRI serves as the primary diagnostic tool for various brain disorders, including tumor detection. As the diagnostic landscape embraces the sophistication of deep neural networks, the preference for first-order optimizers in deep learning becomes evident. However, when confronted with limited sample sizes, training a stable and generalized model with an extensive parameter set using first-order optimizers poses challenges. Our focus lies in scrutinizing brain MRI images specifically for the purpose of brain tumor detection. This dataset comprises 253 MRI images, encompassing 155 cases with tumors and 98 instances of a healthy brain. The dimension of this dataset is 1000. We employ a transfer learning strategy for tumor detection, a technique extensively applied in brain MRI and biological scenarios characterized by a limited sample size. Within deep models, the lower layers are dedicated to generic tasks like edge detection, while the upper layers are tailored for specific tasks. Consequently, the conventional approach involves fine-tuning exclusively the upper layers. The goal is to minimize the objective function:

$$\min_{w} f(w), \quad \text{where} \quad f(w) = \sum_{i=1}^{n} f_i\,(w),$$

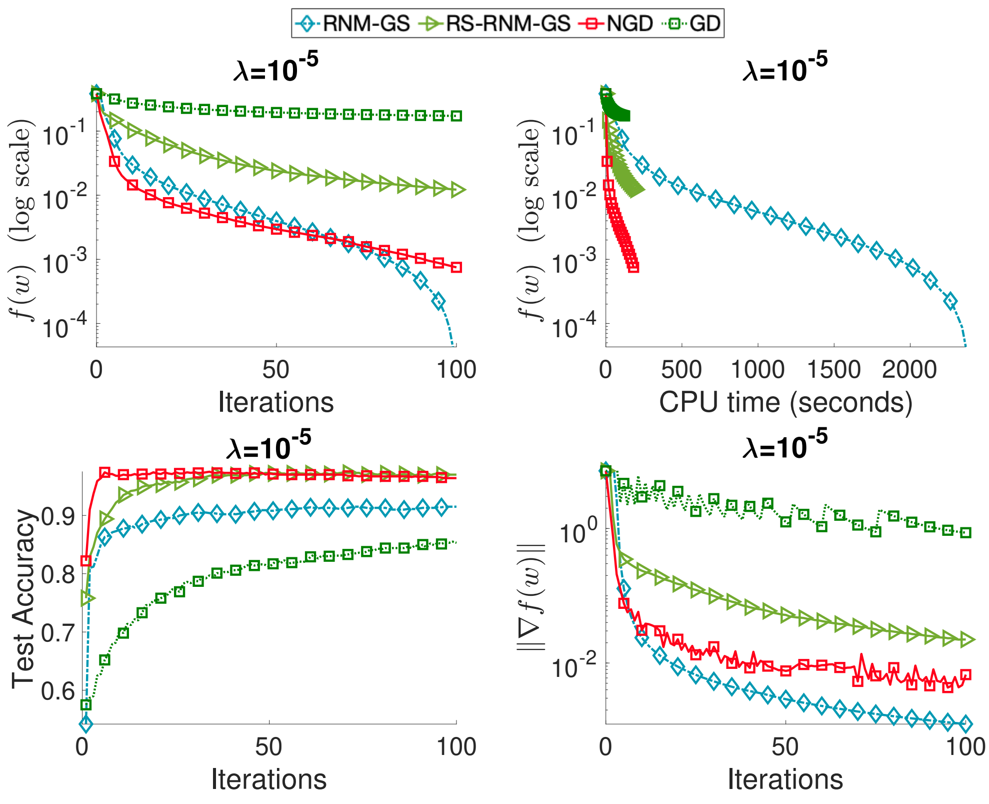


Figure 5.11: Experiments on *gisette* dataset for ℓ_2 regularized Geman-McClure loss *m=500*

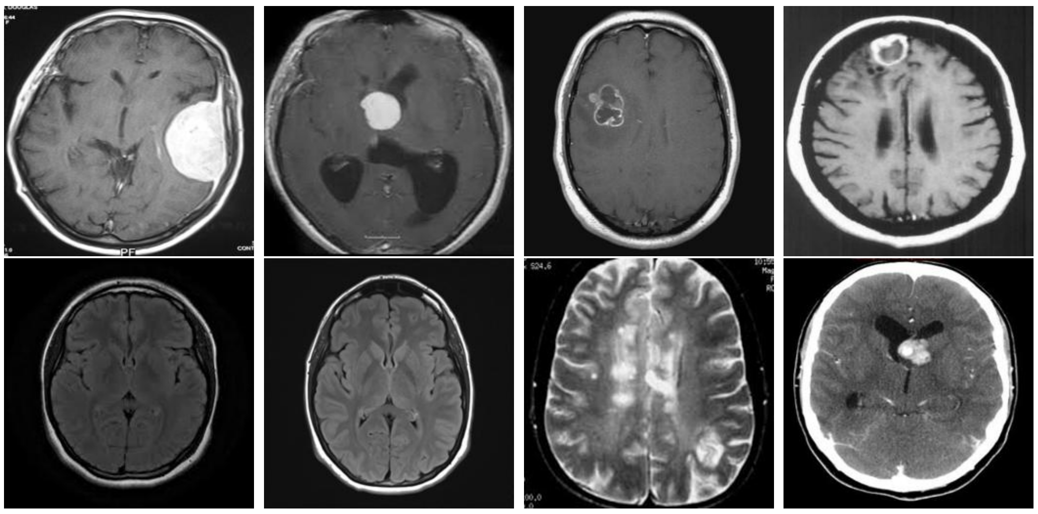


Figure 5.12: Sample Images from MRI dataset [1], Top row: Tumor, Bottom row: Healthy

where $w \in \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}$, and f_i is the loss function corresponding to i^{th} sample is the *logistic regression* for brain tumor *classification* problem. *i.e.*, The data has d dimensions and n samples. We propose an NGD algorithm for fine-tuning the top layers of pre-trained deep networks. Specifically, we compute a partial column Hessian of size $(d \times m)$ with $m \ll d$ uniformly randomly selected variables (d is the number of parameters), then use the *Nyström* *method* to approximate the full Hessian matrix.

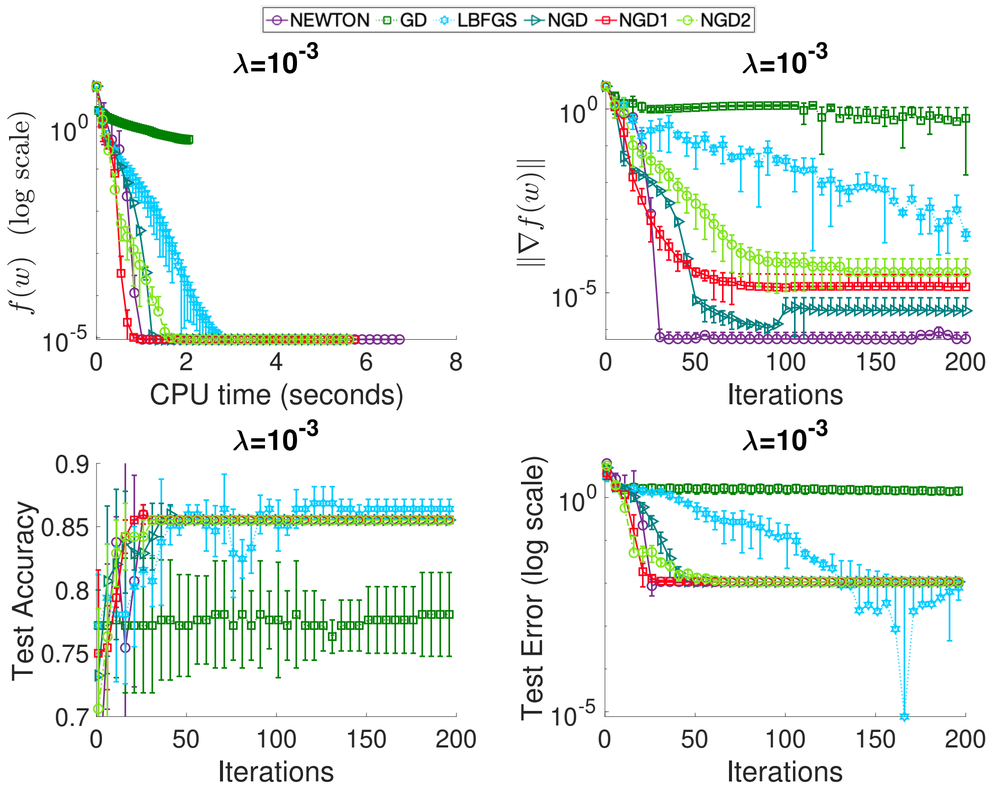


Figure 5.13: Comparison of NGDs with existing methods on MRI datase ($m = 200$)

Figure 5.13 shows that NGD1 outperforms other methods in terms of training cost in the least CPU time. Newton's method outperforms in terms of decreasing the norm of the gradient. Additionally, all NGDs are giving competitive behavior to each other in terms of the norm of gradient. GD and L-BFGS are not able to give competitive results in terms of test accuracy and test error. Also, all NGDs and Newton's method have the upper hand in achieving better test accuracy and test error.

5.9 Conclusions

In this chapter, we introduce the regularized Nyström method to approximate Hessian and propose both deterministic and stochastic optimization methods to solve the objective function. We present the comprehensive convergence analysis and certain results using the distance between the Hessian and Nyström approximations. Furthermore, we conducted extensive numerical experiments to evaluate the performance of the proposed methods with RSN [52], NS [93], and other existing first-order and quasi-Newton methods. From the numerical results, the proposed methods demonstrate robustness, efficiently approximating the Hessian by selecting approximately 5%(in high-density scenarios) and 15-20%(in high-sparsity scenarios) of the dimension. Moreover, we employ the proposed method for an application involving brain tumor detection. The results in this application highlight the promising impact of our proposed methods in real-world scenarios.

Chapter 6

Conclusions and future works

6.1 Conclusions

In this thesis, we developed diverse strategies to solve the large-scale unconstrained optimization problems for both deterministic and stochastic scenarios. We summarized chapter-wise conclusions.

In chapter 3, we addressed a scenario where the computation of a full gradient is feasible. Our proposal introduces a novel approach by simultaneously employing Wolfe line search with the regularized L-BFGS method. This innovation aims to handle the challenge of reducing the number of function evaluations while determining the optimal step size for large-scale, unconstrained optimization problems. By efficiently integrating Wolfe line search in cases where the iterate fails to satisfy the curvature condition, the proposed method facilitates the computation of larger steps. This proposition introduces a novel perspective on utilizing line search techniques, specifically through the conditional use of Wolfe line search. Numerical experiments demonstrate that RL-BFGS-SW solves a similar number of problems as RL-BFGS but exhibits enhanced capabilities in solving certain large-scale problems with fewer function evaluations.

In chapter 4, we moved to a case where the computation of the full gradient is costly. We proposed a novel idea and introduced a unique approach by employing the Barzilai-Borwein (BB) method as a Hessian approximation to further reduce the variance of the stochastic gradient in empirical risk minimization problems for strongly convex functions. Unlike similar methods that use different Hessian approximations containing information about multiple eigenvalues, the BB-method represents a constant diagonal matrix. This means the BB-approximation maintains the same constant value on the diagonal, signifying an approximation focused on a specific eigenvalue. This proposal explored the unexplored concept of using the BB method in matrix-vector multiplication with a vector of parameters. Theoretically, it is sound to use the BB method as it satisfies both strongly convexity and

Lipschitz continuity of the gradient. It has similar convergence properties as the similar existing methods. This proposal shows its efficiency for dense and high-dimensional datasets, as the BB-approximation becomes too small when the gradient is near the optimal point. We establish linear convergence not only for the proposed methods but also for SVRG-2 and SVRG-2D [53], leveraging a unique lemma. Moreover, we propose the utilization of the BB-method as a BB-step size with certain parameter modifications. Numerical experiments with these variants demonstrate performance similar to existing methods. However, SVRG-2BB exhibits exceptional and stable performance on specific benchmark datasets, particularly when dealing with dense data and high dimensions.

In chapter 5, we considered a case where partial Hessian information is affordable. Our novel approach introduces a Regularized Nyström method that leverages the Nyström method to approximate the Hessian for both deterministic and stochastic large-scale unconstrained strongly convex optimization problems. Shifting away from the conventional use of first-order methods for Hessian approximation, the Nyström method provides actual Hessian information with computational and memory costs similar to L-BFGS. This makes the deterministic regularized Nyström method robust and outperforms the existing similar methods. Theoretical analysis establishes bounds representing the distance between the proposed approximation and the Hessian, indicating the strong Hessian approximation generated by the proposed method. Numerical experiments affirm the excellent performance of the proposed method in all scenarios for the deterministic case and outperform existing *state-of-the-art* methods. In addition, regularized Nyström method outperforms the existing methods in the classification problem of tumor detection as an application for brain MRI. This shows that the proposed method can be useful in various real-world applications.

As we summarized above, we have made several contributions to solving large-scale, unconstrained optimization problems. However, there are many challenges that remain to be addressed. Finally, in the next subsection, we discuss some future work based on the current achievements and further challenges involved in the proposed methods.

6.2 Future works

In this subsection, we present a glimpse of both ongoing efforts and future developments that are somewhat related to the work presented in previous chapters of this thesis.

6.2.1 Unstable Nyström approximation and its Levenberg-Marquardt variant

In the regularized Nyström method of **Chapter 5**, the Nyström approximation calculates a partial column Hessian of size $(d \times m)$ with $m \ll d$, using randomly selected column indices. Subsequently, it employs regularized Nyström to obtain a distinctive solution for the search direction. However, the search direction might face instability issues due to the low-rank structure, especially when the regularization parameter is small. To ensure a stable search direction, we are working on proposing a Levenberg-Marquardt-type method for the Nyström approximation.

- Regularized Nyström explained in subsection 5.2.1 of **Chapter 5** solves the system of linear equations to find search direction:

$$(\boldsymbol{Z}\boldsymbol{Z}^T + \rho I)\boldsymbol{d} = -\boldsymbol{g},$$

  where $\boldsymbol{Z}\boldsymbol{Z}^T$ is Nyström approximation given in (5.2.6), $\rho > 0$ is a regularized parameter, and $\boldsymbol{g}$ is a gradient.

- Note that when ρ is small, the solution d can be unstable.

- One solution is to use the Levenberg-Marquardt (LM) technique as a solution to the following subproblem:

$$\boldsymbol{d} \in \text{ argmin } \|\boldsymbol{Z}\boldsymbol{Z}^T\boldsymbol{d} + \boldsymbol{g}\|^2 + \rho\|\boldsymbol{d}\|^2,$$

- which gives a stable solution:

$$(\boldsymbol{Z}\boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{Z}^T + \rho\boldsymbol{I})\boldsymbol{d} = -\boldsymbol{Z}\boldsymbol{Z}^T\boldsymbol{g}.$$

- Finally, a Nyström-LM variant is given as:

$$\boldsymbol{d} = -(\boldsymbol{Z}\boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{Z}^T + \rho\boldsymbol{I})^{-1}\boldsymbol{Z}\boldsymbol{Z}^T\boldsymbol{g}.$$

- In order to make sure the direction is decent, we add the parameter $\delta > 0$ as follows:

$$\boldsymbol{d} = -(\boldsymbol{Z}\boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{Z}^T + \rho\boldsymbol{I})^{-1}(\boldsymbol{Z}\boldsymbol{Z}^T + \delta\boldsymbol{I})\boldsymbol{g}.$$

- Finally, Nyström-LM can be obtained as:

$$\boldsymbol{B} = (\boldsymbol{Z}\boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{Z}^T + \rho\boldsymbol{I})^{-1}(\boldsymbol{Z}\boldsymbol{Z}^T + \delta\boldsymbol{I}), \tag{6.2.1}$$

  and it can be incorporated into certain gradient methods to obtain a quasi-Newton-type update.

6.2.2 Modifying the initial matrix of L-BFGS with Nyström like methods

While L-BFGS performs effectively with large-scale convex optimization problems, the Hessian approximation using the L-BFGS matrix can exhibit significant changes based on the initial matrix. Given the successful performance of both regularized L-BFGS and Nyström methods in large-scale unconstrained problems, a proposal is made to leverage the Nyström method for initializing the L-BFGS matrix. This integration is explored within the framework of regularized L-BFGS, accompanied by suitable modifications.

The L-BFGS-SW explained in **Chapter 1** in subsection 1.2.3 computes the approximated Hessian at iteration x_k using the last m memory vector pairs (s_k, y_k) for $i = k-m, \dots, k-1$ and the initial matrix $H_k^0 \; = \gamma_k I$ where

$$\gamma_k \; = \frac{s_{k-1}^\top y_{k-1}}{y_{k-1}^\top y_{k-1}}, \tag{6.2.2}$$

is an initial matrix with $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. The idea is to employ a robust initial matrix such as the Nyström approximation

$$H_k^0 = H_k S_k (S_k^\top H_k S_k)^\dagger S_k^\top H_k, \tag{6.2.3}$$

or randomized subspace Newton

$$H_k^0 = S_k (S_k^\top H_k S_k)^\dagger S_k^\top, \tag{6.2.4}$$

instead of γ_k, where $S_k \in \mathbb{R}^{d \times m}$ is a random matrix. Once we have modified the initial matrix, we may employ it in various techniques that use the L-BFGS method.

6.2.3 Tackling heterogeneous data challenges: Future directions into distributed, federated, and multi-objective optimization

Future work in the realm of distributed, federated, and multi-objective optimization is paramount, as these areas hold crucial significance. Distributed optimization serves as a pivotal tool with diverse applications in fields such as machine learning, autonomous systems, and networking. The exponential growth in data and the prevalence of cloud computing and edge devices underscore the necessity of employing distributed optimization for the efficient utilization of computational resources. Addressing this imperative, there is a substantial need to propose distributed variants of optimization methods tailored to the unique challenges posed by different application domains.

The distributed nature of the optimization process allows for parallel computation, enabling each computing node to optimize its local objective function based on its specific dataset and problem context. However, the emergence of heterogeneous data poses a significant challenge to this process. Consequently, there is a motivation to explore solutions that address the relationships between the diverse distributions present in the data across various devices or nodes by incorporating novel concepts like optimal transport.

Federated learning, conceptualized as a privacy-preserving form of distributed optimization, is a noteworthy area of focus. As part of my future work, I intend to investigate the application of Nyström-type approximation to resolve federated learning problems. This approach not only contributes to solving the challenge of heterogeneity but also aids in reducing communication between the central server and local devices.

A recent study [62] discovered that shared attributes, such as decentralized and parallel computation, between distributed optimization and multi-objective optimization indicate potential synergies and possibilities for their integration. This observation opens avenues for exploring the convergence of these approaches, emphasizing the need for further research and development in this dynamic and interdisciplinary field.

Appendices

Appendix: Theoretical properties of a regularized Nyström method of Chapter 5

A Exactness of Nyström approximation

Here, we present a result to obtain the distance between Hessian and Nyström approximation based on the size of the number of columns m or rank of M. [75] showed a stronger result in the following theorem:

Theorem A.1. *[75, Theorem 3] Suppose $\boldsymbol{H} \in \mathbb{R}^{d\times d}$ is positive semi-definite matrix and rank$(\boldsymbol{H}) = r \leq d$. Consider the Nyström approximation $\boldsymbol{N} = \boldsymbol{C}\boldsymbol{M}^{\dagger}\boldsymbol{C}^{\top}$ and rank$(\boldsymbol{M}) = r \leq m \leq d$, where m is the number of columns picked randomly. Then the Nyström approximation is exact. i.e.,*

$$\|\boldsymbol{H} - \boldsymbol{N}\|_F = 0,$$

where $\|.\|_F$ is the Frobenious norm.

Note that $\|\boldsymbol{A}\|_2 \leq \|\boldsymbol{A}\|_F$ for any matrix $\boldsymbol{A}$. From the above theorem, it is easy to see that Nyström approximation produces the exactly same singular values when rank$(\boldsymbol{M}) = r$. Hence, we can expect to achieve the same convergence as the Newton's method or superlinear convergence, at least when the number of columns chosen is $m \geq r$ and rank$(\boldsymbol{H}) =$ rank$(\boldsymbol{M}) = r$. Moreover, it tells us that when rank$(\boldsymbol{M}) < r$, we cannot achieve quadratic convergence since the distance between Hessian and Nyström is bounded from above and not exactly zero.

Remark A.1. *To have the least possible value of m (i.e., $m = r$) that satisfies the above theorem, we need to choose exactly those r independent columns of $\boldsymbol{H}$ which is difficult due to the randomness involved in choosing m. In short, when rank$(\boldsymbol{H}) = r$, it becomes a feature selection problem to choose the r independent columns that will form a Nyström approximation. The usual convergence gives a probabilistic convergence due to the* *randomness involved in m and the convergence rate depends on the size of the number of randomly chosen columns $m = |\Omega|$.*

B Bound on the difference between NGD's search direction and Newton's direction

Lemma B.1. *Suppose that Assumption 5.4.1 holds. Let $\{\boldsymbol{w}\}$ be a sequence generated by Algorithm 5. If*

$$m > 64k\vartheta/\varepsilon^4,$$

then

$$\|\boldsymbol{p}_t - \boldsymbol{p}_t^N\| \leq \frac{1}{\lambda}(U_{Nys} + c_1\|\boldsymbol{g}_t\|^\gamma)\|\boldsymbol{p}_t\|,$$

with probability at least $1 - \varrho$, where U_{Nys} is an upper bound of $\|\boldsymbol{H}_t - \boldsymbol{N}_t\|$ given in Theorem 5.2.1. Moreover, if $\operatorname{rank}(\boldsymbol{M}) = \operatorname{rank}(\boldsymbol{H})$, then

$$\frac{\|\boldsymbol{p}_t - \boldsymbol{p}_t^N\|}{\|\boldsymbol{p}_t\|} \leq \frac{c_1}{\lambda}\|\boldsymbol{g}_t\|^\gamma,$$

with probability at least $1 - \varrho$ given in Theorem 5.2.1.

Proof. Let the direction of the Newton's method be $\boldsymbol{p}_t^N = -\nabla^2 f(\boldsymbol{w}_t)^{-1}\boldsymbol{g}_t$ and regularized Nyström direction is $\boldsymbol{p}_t = -(\boldsymbol{N}_t + \rho_t\boldsymbol{I})^{-1}\boldsymbol{g}_t$. Since f is strongly convex, $\lambda_{\min}(\nabla^2 f(\boldsymbol{w})) \geq \lambda$, let $\nabla^2 f(\boldsymbol{w}) = \boldsymbol{H}$. Then we have $\|\boldsymbol{H}_t^{-1}\| \leq \frac{1}{\lambda}$ for $t > 0$. Next, the distance between the directions can be given as:

$$\begin{aligned}
\|\boldsymbol{p}_t - \boldsymbol{p}_t^N\| &= \|\boldsymbol{H}_t^{-1}\left(\boldsymbol{H}_t\boldsymbol{p}_t + \boldsymbol{g}_t\right)\| \\
&= \|\boldsymbol{H}_t^{-1}\left(\boldsymbol{H}_t - (\boldsymbol{N}_t + \rho_t\boldsymbol{I})\right)\boldsymbol{p}_t\| \\
&\leq \|\boldsymbol{H}_t^{-1}\|\|\left(\boldsymbol{H}_t - (\boldsymbol{N}_t + \rho_t\boldsymbol{I})\right)\boldsymbol{p}_t\| \\
&= \|\boldsymbol{H}_t^{-1}\|\|\left(\boldsymbol{H}_t - \boldsymbol{N}_t\right)\boldsymbol{p}_t - (\rho_t\boldsymbol{I})\boldsymbol{p}_t\| \\
&\leq \|\boldsymbol{H}_t^{-1}\|\|\left(\boldsymbol{H}_t - \boldsymbol{N}_t\right)\boldsymbol{p}_t\| + \|\boldsymbol{H}_t^{-1}\|\|\rho_t\boldsymbol{p}_t\| \\
&\leq \|\boldsymbol{H}_t^{-1}\|\|\boldsymbol{H}_t - \boldsymbol{N}_t\|\|\boldsymbol{p}_t\| + c_1\|\boldsymbol{H}_t^{-1}\|\|\boldsymbol{g}_t\|^\gamma\|\boldsymbol{p}_t\|. \qquad \text{(B.1)}
\end{aligned}$$

- **case a)** In this case, we discuss the distance $\|\boldsymbol{p}_t - \boldsymbol{p}_t^N\|$, when $m > 64k\vartheta/\varepsilon^4$ (Theorem 5.2.1) or $\operatorname{rank}(\boldsymbol{M}_t) < \operatorname{rank}(\boldsymbol{H}_t)$.

  Using Theorem 5.2.1 in the (B.1), we get

$$\|\boldsymbol{p}_t - \boldsymbol{p}_t^N\| \leq \|\boldsymbol{H}^{-1}\|\|\boldsymbol{H}_t - \boldsymbol{N}_t\|\|\boldsymbol{p}_t\| + c_1\|\boldsymbol{H}_t^{-1}\|\|\boldsymbol{g}_t\|^\gamma\|\boldsymbol{p}_t\|$$

$$\leq \frac{1}{\lambda}(U_{Nys} + c_1\|\boldsymbol{g}_t\|^{\gamma})\|\boldsymbol{p}_t\|,$$

where $\|\boldsymbol{H}_t^{-1}\| \leq \frac{1}{\lambda}$, and $\|\boldsymbol{H}_t - \boldsymbol{N}_t\| \leq U_{Nys}$.

- **case b)** For this case, we obtain a result when $\operatorname{rank}(\boldsymbol{M}) = \operatorname{rank}(\boldsymbol{H})$.

  Using the Theorem A.1 in (B.1), we get

$$\begin{aligned}\|\boldsymbol{p}_t - \boldsymbol{p}_t^N\| &\leq \|\boldsymbol{H}^{-1}\|\|\boldsymbol{H}_t - \boldsymbol{N}_t\|\|\boldsymbol{p}_t\| + c_1\|\boldsymbol{H}_t^{-1}\|\|\boldsymbol{g}_t\|^{\gamma}\|\boldsymbol{p}_t\| \\ &= c_1\|\boldsymbol{H}_t^{-1}\|\|\boldsymbol{g}_t\|^{\gamma}\|\boldsymbol{p}_t\|.\end{aligned}$$

  Hence, we get

$$\frac{\|\boldsymbol{p}_t - \boldsymbol{p}_t^N\|}{\|\boldsymbol{p}_t\|} \leq c_1\|\boldsymbol{H}^{-1}\|\|\boldsymbol{g}_t\|^{\gamma} \leq \frac{c_1}{\lambda}\|\boldsymbol{g}_t\|^{\gamma},$$

  where $\|\boldsymbol{H}_t^{-1}\| \leq \frac{1}{\lambda}$.

This completes the proof. ■

Remark B.1. *$\boldsymbol{H}$ may not be the full rank matrix if f is not a strongly convex function. Then disregarding Assumption 5.4.1 for case (b) in above lemma holds for $d = m$ if f strongly convex function and may be $m < d$ if f not strongly convex function.*

C Closeness to the Hessian inverse

In this subsection, we discuss the closeness of the inverse of regularized Nyström approximation with the Hessian inverse. Let $\boldsymbol{H}$ be the Hessian of the objective function (5.1.1) and we consider the regularized Newton's method regularized by any $\rho > 0$. Then, the inverse of Hessian is given by $(\boldsymbol{H} + \rho\boldsymbol{I})_{\boldsymbol{w}}^{-1} = (\nabla^2 f(\boldsymbol{w}) + \rho\boldsymbol{I})^{-1}$ at $\boldsymbol{w}$. Let the regularized Nyström at $\boldsymbol{w}$ be given by $(\boldsymbol{Z}_{\boldsymbol{w}}\boldsymbol{Z}_{\boldsymbol{w}}^{\top} + \rho\boldsymbol{I})^{-1}$. The distance of the regularized inverse matrix is then given as

$$\|(\boldsymbol{Z}_{\boldsymbol{w}}\boldsymbol{Z}_{\boldsymbol{w}}^{\top} + \rho\boldsymbol{I})^{-1} - (\boldsymbol{H} + \rho\boldsymbol{I})_{\boldsymbol{w}}^{-1}\| \leq \frac{\|\boldsymbol{J}_{\boldsymbol{w}}\|}{\rho(\|\boldsymbol{J}_{\boldsymbol{w}}\| + \rho)}, \tag{C.1}$$

where $0 < \|\boldsymbol{J}_{\boldsymbol{w}}\| = \|\boldsymbol{H} - \boldsymbol{Z}_{\boldsymbol{w}}\boldsymbol{Z}_{\boldsymbol{w}}^{\top}\| \leq \|\boldsymbol{H} - \boldsymbol{H}_k\| + \varepsilon\sum_{i=1}^{d}(\boldsymbol{H}_{ii})^2$; which follows from (5.2.3), whereas (C.1) follows from [46, Proposition 3.1].

Note that the rank of Hessian can be possibly r when the objective function is not ℓ_2 regularized.

Bibliography

[1] *Dataset : Brain MRI images for brain tumor detection*, https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection, (2020).

[2] N. Agarwal, B. Bullins, and E. Hazan, *Second-order stochastic optimization for machine learning in linear time*, Journal of Machine Learning Research, JMLR, 18 (2017), pp. 116:1–116:40.

[3] R. Baraniuk, *A lecture on compressive sensing*, IEEE signal processing magazine, 24 (2007).

[4] J. T. Barron, *A general and adaptive robust loss function*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4331–4339.

[5] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, *Convexity, classification, and risk bounds*, Journal of the American Statistical Association, 101 (2006), pp. 138–156.

[6] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA Journal of Numerical Analysis, 8 (1988), pp. 141–148.

[7] D. Bertsekas, A. Nedic, and A. Ozdaglar, *Convex analysis and optimization*, vol. 1, Athena Scientific, 2003.

[8] D. P. Bertsekas, *Nonlinear programming*, Journal of the Operational Research Society, 48 (1997), pp. 334–334.

[9] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, Springer, 2006.

[10] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.

[11] C. G. Broyden, *The convergence of a class of double-rank minimization algorithms: 2. the new algorithm*, IMA Journal of Applied Mathematics, 6 (1970), pp. 222–231.

[12] A. M. Bruckstein, D. L. Donoho, and M. Elad, *From sparse solutions of systems of equations to sparse modeling of signals and images*, SIAM review, 51 (2009), pp. 34–81.

[13] O. Burdakov, L. Gong, S. Zikrin, and Y.-x. Yuan, *On efficiently combining limited-memory and trust-region techniques*, Mathematical Programming Computation, 9 (2017), pp. 101–134.

[14] J. V. Burke and A. Wiegmann, *Notes on limited memory BFGS updating in a trust-region framework*, Preprint, Department of Mathematics, University of Washington, Seattle, WA, (1996).

[15] J. V. Burke, A. Wiegmann, and L. Xu, *Limited memory BFGS updating in a trust-region framework*, SIAM Journal on Optimization, 17 (2008).

[16] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, *A stochastic quasi-newton method for large-scale optimization*, SIAM Journal on Optimization, 26 (2016), pp. 1008–1031.

[17] R. H. Byrd, J. Nocedal, and R. B. Schnabel, *Representations of quasi-newton matrices and their use in limited memory methods*, Mathematical Programming, 63 (1994), pp. 129–156.

[18] R. H. Byrd, J. Nocedal, and Y.-X. Yuan, *Global convergence of a cass of quasi-newton methods on convex problems*, SIAM Journal on Numerical Analysis, 24 (1987), pp. 1171–1190.

[19] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on information theory, 52 (2006), pp. 489–509.

[20] E. J. Candès and M. B. Wakin, *An introduction to compressive sampling*, IEEE signal processing magazine, 25 (2008), pp. 21–30.

[21] E. J. Candes, M. B. Wakin, and S. P. Boyd, *Enhancing sparsity by reweighted ℓ_1 minimization*, Journal of Fourier analysis and applications, 14 (2008), pp. 877–905.

[22] A. Cauchy, *Méthode générale pour la résolution des systemes d'équations simultanées*, Comptes rendus de l'Académie des Sciences Paris, 25 (1847), pp. 536–538.

[23] S. L. Cessie and J. V. Houwelingen, *Ridge estimators in logistic regression*, Journal of the Royal Statistical Society Series C: Applied Statistics, 41 (1992), pp. 191–201.

[24] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, 2 (2011), pp. 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[25] A. R. Conn, N. I. Gould, and P. L. Toint, *Testing a class of methods for solving minimization problems with simple bounds on the variables*, Mathematics of Computation, 50 (1988), pp. 399–430.

[26] A. R. Conn, N. I. Gould, and P. L. Toint, *Convergence of quasi-newton matrices generated by the symmetric rank one update*, Mathematical Programming, 50 (1991), pp. 177–195.

[27] W. C. Davidon, *Variable metric method for minimization*, Technical Report ANLC - 5990 (Revised), (1959).

[28] W. C. Davidon, *Variable metric method for minimization*, SIAM Journal on Optimization, 1 (1991), pp. 1–17.

[29] D. W. Decker, H. B. Keller, and C. T. Kelley, *Convergence rates for Newton's method at singular points*, SIAM Journal on Numerical Analysis, 20 (1983), pp. 296–314.

[30] D. W. Decker and C. T. Kelley, *Newton's method at singular points. I*, SIAM Journal on Numerical Analysis, 17 (1980), pp. 66–70.

[31] 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, NeurIPS, (2014), pp. 1646–1654.

[32] F. Dellaert, M. Kaess, et al., *Factor graphs for robot perception*, Foundations and Trends® in Robotics, 6 (2017), pp. 1–139.

[33] J. E. Dennis, Jr and J. J. Moré, *Quasi-newton methods, motivation and theory*, SIAM review, 19 (1977), pp. 46–89.

[34] J. E. Dennis Jr and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, SIAM, 1996.

[35] M. Derezinski, R. Khanna, and M. W. Mahoney, *Improved guarantees and a multiple-descent curve for column subset selection and the Nyström method*, Advances in Neural Information Processing Systems, NeurIPS, 33 (2020), pp. 4953–4964.

[36] M. Derezinski, J. Lacotte, M. Pilanci, and M. W. Mahoney, *Newton-less: Sparsification without trade-offs for the sketched newton update*, Advances in Neural Information Processing Systems, NeurIPS, 34 (2021), pp. 2835–2847.

[37] P. Deuflhard, *A short history of Newton's method*, Documenta Mathematica, Optimization Stories, (2012), pp. 25–30.

[38] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213.

[39] P. Drineas and M. W. Mahoney, *On the Nyström method for approximating a gram matrix for improved kernel-based learning*, Journal of Machine Learning Research, JMLR, 6 (2005), pp. 2153–2175.

[40] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization.*, Journal of Machine Learning Research, JMLR, 12 (7), (2011).

[41] D. K. Faddeev, V. N. Faddeeva, and R. C. Williams, *Computational methods of linear algebra*, WH Freeman San Francisco, 1963.

[42] C. Fang, C. J. Li, Z. Lin, and T. Zhang, *Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator*, Advances in Neural Information Processing Systems, NeurIPS, 31 (2018).

[43] R. Fletcher, *A new approach to variable metric algorithms*, The Computer Journal, 13 (1970), pp. 317–322.

[44] ——, *Practical methods of optimization*, John Wiley & Sons, 2013.

[45] R. Fletcher and C. M. Reeves, *Function minimization by conjugate gradients*, The Computer Journal, 7 (1964), pp. 149–154.

[46] Z. Frangella, J. A. Tropp, and M. Udell, *Randomized Nyström preconditioning*, arXiv preprint arXiv:2110.02820, (2021).

[47] J. Friedman, T. Hastie, and R. Tibshirani, *A note on the group lasso and a sparse group lasso*, arXiv preprint arXiv:1001.0736, (2010).

[48] T. Fuji, P.-L. Poirion, and A. Takeda, *Randomized subspace regularized Newton method for unconstrained non-convex optimization*, arXiv preprint arXiv:2209.04170, (2022).

[49] A. Gittens, *The spectral norm error of the naive nystrom extension*, arXiv preprint arXiv:1110.5305, (2011).

[50] D. Goldfarb, *A family of variable-metric methods derived by variational means*, Mathematics of Computation, 24 (1970), pp. 23–26.

[51] N. I. Gould, D. Orban, and P. L. Toint, *Cuter and sifdec: A constrained and unconstrained testing environment, revisited*, ACM Transactions on Mathematical Software (TOMS), 29 (2003), pp. 373–394.

[52] R. Gower, D. Kovalev, F. Lieder, and P. Richtárik, *Rsn: randomized subspace newton*, Advances in Neural Information Processing Systems, NeurIPS, 32 (2019).

[53] R. Gower, N. L. Roux, and F. Bach, *Tracking the gradients using the hessian: A new look at variance reducing stochastic methods*, in Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, AISTATS, vol. 84, Proceedings of Machine Learning Research, 09–11 Apr (2018), pp. 707–715.

[54] R. M. Gower and P. Richtárik, *Randomized quasi-newton updates are linearly convergent matrix inversion algorithms*, SIAM Journal on Matrix Analysis and Applications, 38 (2017), pp. 1380–1409.

[55] L. Grippo, F. Lampariello, and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM Journal on Numerical Analysis, 23 (1986), pp. 707–716.

[56] Y. Guo, *Supervised exponential family principal component analysis via convex optimization*, Advances in Neural Information Processing Systems, NeurIPS, 21 (2008).

[57] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2, Springer, 2009.

[58] M. R. Hestenes, *Conjugate direction methods in optimization*, vol. 12, Springer Science & Business Media, 2012.

[59] M. R. Hestenes, E. Stiefel, et al., *Methods of conjugate gradients for solving linear systems*, Journal of research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[60] G. Hinton, N. Srivastava, and K. Swersky, *Neural networks for machine learning, Lecture 6a : overview of mini-batch gradient descent*, University of Torronto, (2012).

[61] A. E. Hoerl and R. W. Kennard, *Ridge regression: applications to nonorthogonal problems*, Technometrics, 12 (1970), pp. 69–82.

[62] Z. Hu, K. Shaloudegi, G. Zhang, and Y. Yu, *Federated learning meets multi-objective optimization*, IEEE Transactions on Network Science and Engineering, 9 (2022), pp. 2039–2051.

[63] J. Huang, T. Zhang, and D. Metaxas, *Learning with structured sparsity*, in Proceedings of the International Conference on Machine Learning ICML, 2009, pp. 417–424.

[64] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, Advances in Neural Information Processing Systems, NeurIPS, 26 (2013).

[65] L. Kantorovitch, *The method of successive approximation for functional equations*, Acta Mathematica, 71 (1939), pp. 63–97.

[66] C. Kanzow and D. Steck, *Regularization of limited memory quasi-newton methods for large-scale nonconvex minimization*, Mathematical Programming Computation, (2023), pp. 1–28.

[67] H. Kasai, *Sgdlibrary: A MATLAB library for stochastic optimization algorithms*, Journal of Machine Learning Research, JMLR, 18 (2018), pp. 1–5.

[68] U. Kenji, *A regularized Newton method without line search for unconstrained optimization*, tech. rep., Technical Report, 2009-007, Department of Applied Mathematics and Physics, Kyoto University, 2009.

[69] H. F. Khalfan, R. H. Byrd, and R. B. Schnabel, *A theoretical and experimental study of the symmetric rank-one update*, SIAM Journal on Optimization, 3 (1993), pp. 1–24.

[70] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, *An interior-point method for large-scale ℓ_1-regularized least squares*, IEEE Journal of Selected Topics in Signal Processing, 1 (2007), pp. 606–617.

[71] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in Proceedings of the International Conference on Learning Representations, ICLR, Y. Bengio and Y. LeCun, eds., 2015.

[72] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, *The unconstrained binary quadratic programming problem: a survey*, Journal of Combinatorial Optimization, 28 (2014), pp. 58–81.

[73] R. Kolte, M. Erdogdu, and A. Ozgur, *Accelerating SVRG via second-order information*, in Advances in Neural Information Processing Systems, workshop on optimization for machine learning, NeurIPS, 2015.

[74] M. Kowalski and B. Torrésani, *Structured sparsity: from mixed norms to structured shrinkage*, in SPARS'09-Signal Processing with Adaptive Sparse Structured Representations, 2009.

[75] S. Kumar, M. Mohri, and A. Talkwalkar, *On sampling-based approximate spectral decomposition*, in International Conference on Machine Learning, ICML, 2009.

[76] J. Lacotte, Y. Wang, and M. Pilanci, *Adaptive Newton sketch: Linear-time optimization with quadratic convergence and effective hessian dimensionality*, in Proceedings of the 38th International Conference on Machine Learning, ICML, M. Meila and T. Zhang, eds., vol. 139, 2021, pp. 5926–5936.

[77] D.-H. Li, M. Fukushima, L. Qi, and N. Yamashita, *Regularized Newton methods for convex minimization problems with singular solutions*, Computational Optimization and Applications, 28 (2004), pp. 131–147.

[78] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Mathematical Programming, 45 (1989), pp. 503–528.

[79] D. G. Luenberger and Y. Ye, *Linear and nonlinear programming*, Springer, 3 ed., 2008.

[80] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM Journal on Scientific and Statistical Computing, 4 (1983), pp. 553–572.

[81] P. Moritz, R. Nishihara, and M. Jordan, *A linearly-convergent stochastic L-BFGS algorithm*, in Proceedings of the International Conference on Artificial Intelligence and Statistics, AISTATS, 2016, pp. 249–258.

[82] S. G. Nash and A. Sofer, *Linear and nonlinear programming*, McGraw-Hill Science, Engineering & Mathematics, 1996.

[83] Y. Nesterov, *Introductory Lectures on Convex Optimization*, Springer US, 2004.

[84] Y. E. Nesterov, *A method for solving the convex programming problem with convergence rate $O(1/k^2)$*, in Dokl. akad. nauk Sssr, vol. 269, 1983, pp. 543–547.

[85] A. Y. Ng, *Feature selection, l 1 vs. l 2 regularization, and rotational invariance*, in Proceedings of the International Conference on Machine Learning, ICML, 2004, p. 78.

[86] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, in Proceedings of the International Conference on Machine Learning, ICML, 2017, pp. 2613–2621.

[87] J. Nocedal, *Updating quasi-newton matrices with limited storage*, Mathematics of Computation, 35 (1980), pp. 773–782.

[88] J. Nocedal, *Software for large-scale unconstrained optimization: L-BFGS distribution*, 2014.

[89] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[90] R. Overbeek, N. Larsen, G. D. Pusch, M. D'Souza, E. S. Jr, N. Kyrpides, M. Fonstein, N. Maltsev, and E. Selkov, *Wit: integrated system for high-throughput genome sequence analysis and metabolic reconstruction*, Nucleic Acids Research, 28 (2000), pp. 123–125.

[91] J. D. Pearson, *Variable metric methods of minimisation*, The Computer Journal, 12 (1969), pp. 171–178.

[92] D. L. Phillips, *A technique for the numerical solution of certain integral equations of the first kind*, Journal of the ACM, JACM, 9 (1962), pp. 84–97.

[93] M. Pilanci and M. J. Wainwright, *Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence*, SIAM Journal on Optimization, 27 (2017), pp. 205–245.

[94] B. T. Polyak, *Some methods of speeding up the convergence of iteration methods*, Ussr Computational Mathematics and Mathematical Physics, 4 (1964), pp. 1–17.

[95] R. A. Polyak, *Regularized newton method for unconstrained convex optimization*, Mathematical Programming, 120 (2009), pp. 125–145.

[96] M. J. Powell, *Some global convergence properties of a variable metric algorithm for minimization without exact line searches*, Nonlinear Programming, 9 (1976), pp. 53–72.

[97] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM Journal on Optimization, 7 (1997), pp. 26–33.

[98] H. Robbins and S. Monro, *A stochastic approximation method*, The Annals of Mathematical Statistics, 22 (1951), pp. 400–407.

[99] N. Roux, M. Schmidt, and F. Bach, *A stochastic gradient method with an exponential convergence rate for finite training sets*, Advances in Neural Information Processing Systems, NeurIPS, 25 (2012), pp. 2663–2671.

[100] M. Schmidt, N. Le Roux, and F. Bach, *Minimizing finite sums with the stochastic average gradient*, Mathematical Programming, 162 (2017), pp. 83–112.

[101] N. N. Schraudolph, J. Yu, and S. Günter, *A stochastic quasi-newton method for online convex optimization*, in Artificial Intelligence and Statistics, AISTATS, Proceedings of Machine Learning Research, (2007), pp. 436–443.

[102] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, Cambridge University Press, (2014).

[103] S. Shalev-Shwartz and T. Zhang, *Stochastic dual coordinate ascent methods for regularized loss minimization*, Journal of Machine Learning Research, JMLR, 14(16) (2013), pp. 567–599.

[104] D. F. Shanno, *Conditioning of quasi-newton methods for function minimization*, Mathematics of Computation, 24 (1970), pp. 647–656.

[105] D. F. Shanno and K. H. Phua, *Matrix conditioning and nonlinear optimization*, Mathematical Programming, 14 (1978), pp. 149–160.

[106] S. Sugimoto and N. Yamashita, *A regularized limited-memory BFGS method for unconstrained minimization problems*, Technical report 2014-001. Kyoto University, Department of Applied Mathematics and Physics, (2014).

[107] W. Sun, *Nonmonotone trust region method for solving optimization problems*, Applied Mathematics and Computation, 156 (2004), pp. 159–174.

[108] A. Talwalkar, *Matrix approximation for large-scale learning*, PhD thesis, New York University, 2010.

[109] A. Talwalkar and A. Rostamizadeh, *Matrix coherence and the Nyström method*, arXiv preprint arXiv:1408.2044, (2014).

[110] C. Tan, S. Ma, Y.-H. Dai, and Y. Qian, *Barzilai-borwein step size for stochastic gradient descent*, in Advances in Neural Information Processing Systems, NeurIPS, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds., vol. 29, Curran Associates, Inc., (2016), pp. 685–693.

[111] H. Tankaria, D. Singh, and M. Yamada, *Nys-Curve: Nyström-approximated curvature for stochastic optimization*, arXiv preprint arXiv:2110.08577v1, (2021).

[112] H. Tankaria, S. Sugimoto, and N. Yamashita, *A regularized limited memory BFGS method for large-scale unconstrained optimization and its efficient implementations*, Computational Optimization and Applications, 82 (2022), pp. 61–88.

[113] D. A. Tarzanagh and M. R. Peyghami, *A new regularized limited memory BFGS-type method based on modified secant conditions for unconstrained optimization problems*, Journal of Global Optimization, 63 (2015), pp. 709–728.

[114] H. L. Taylor, S. C. Banks, and J. F. McCoy, *Deconvolution with the l 1 norm*, Geophysics, 44 (1979), pp. 39–52.

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

[116] A. N. Tikhonov and V. I. Arsenin, *Solutions of Ill-posed Problems: Andrey N. Tikhonov and Vasiliy Y. Arsenin. Translation Editor Fritz John*, Wiley, 1977.

[117] A. N. Tikhonov, A. Goncharsky, V. V. Stepanov, and A. G. Yagola, *Numerical methods for the solution of ill-posed problems*, vol. 328, Springer Science & Business Media, 1995.

[118] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Fixed-rank approximation of a positive-semidefinite matrix from streaming data*, Advances in Neural Information Processing Systems, NeurIPS, 30 (2017).

[119] K. Ueda, *Studies on regularized Newton-type methods for unconstrained minimization problems and their global complexity bounds*, Kyoto University, (2012).

[120] K. Ueda and N. Yamashita, *Convergence properties of the regularized Newton method for the unconstrained nonconvex optimization*, Applied Mathematics and Optimization, 62 (2010), pp. 27–46.

[121] ——, *A regularized Newton method without line search for unconstrained optimization*, Computational Optimization and Applications, 59 (2014), pp. 321–351.

[122] R. Vershynin, *High-dimensional probability: An introduction with applications in data science*, vol. 47, Cambridge University Press, 2018.

[123] Y. Wang, Z. Lü, F. Glover, and J.-K. Hao, *Path relinking for unconstrained binary quadratic programming*, European Journal of Operational Research, 223 (2012), pp. 595–604.

[124] T. T. Wu and K. Lange, *Coordinate descent algorithms for lasso penalized regression*, The Annals of Applied Statistics, 2 (2008), pp. 224 – 244.

[125] L. Xiao and T. Zhang, *A proximal stochastic gradient method with progressive variance reduction*, SIAM Journal on Optimization, 24 (2014), pp. 2057–2075.

[126] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for l1-minimization with applications to compressed sensing*, SIAM Journal on Imaging Sciences, LIST OF FIGURES, (2008), pp. 143–168.

[127] L. Zhang, M. Mahdavi, and R. Jin, *Linear convergence with condition number independent access of full gradients*, Advances in Neural Information Processing Systems, NeurIPS, 26 (2013), pp. 980–988.

[128] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67 (2005), pp. 301–320.