國立臺灣大學電機資訊學院資訊工程學系
碩士論文
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master's Thesis

隨機擬牛頓方法用於深度神經網路的訓練
Stochastic Quasi-Newton Methods for Training Deep Neural Networks

黃子軒
Zih-Syuan Huang

指導教授：林智仁博士
Advisor: Chih-Jen Lin, Ph.D.

中華民國 114 年 9 月
September, 2025

doi:10.6342/NTU202504473

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

隨機擬牛頓方法用於深度神經網路的訓練

Stochastic Quasi-Newton Methods for Training Deep Neural Networks

本論文係黃子軒君（學號 R11922210）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 114 年 09 月 08 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department of Computer Science and Information Engineering on 8 September 2025 have examined a Master's thesis entitled above presented by ZIH-SYUAN HUANG (student ID: R11922210) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

林智仁 _____ _____

（指導教授 Advisor）

_____ _____ _____

_____ _____ _____

系主任/所長 Director: 陳祝嵩

doi:10.6342/NTU202504473

誌謝

感謝林智仁教授的指導，亦感謝李靜沛副教授與李育杰教授的寶貴建議。

doi:10.6342/NTU202504473

摘要

擬牛頓法因能利用二階資訊而無需實際計算黑森矩陣，在最佳化中已被證明相當有效。然而，將其應用於隨機最佳化，特別是在深度學習情境中，仍面臨挑戰，原因在於梯度估計存在雜訊以及目標函數具非凸性。在本文中，我們提出了一種專為訓練深度神經網路設計的隨機有限記憶 BFGS（LBFGS）方法。我們的方法引入了一種新穎的曲率選擇策略，透過更新頻率機制來挑選曲率最大的曲率對，有效解決隨機性與非凸性問題。此外，我們結合了動量方法，以進一步提升收斂速度。實驗結果顯示，我們的方法在標準的凸與非凸影像分類基準資料集，不僅顯著優於某一個現有的隨機 LBFGS（oLBFGS）方法，還能與廣泛使用的深度學習最佳化方法，如動量隨機梯度下降 (SGDM)、Adam 與 Shampoo 表現相似。

關鍵字： 深度學習、隨機最佳化、擬牛頓法

doi:10.6342/NTU202504473

Abstract

Quasi-Newton methods have proven to be effective for optimization due to their use of second-order information without explicitly computing Hessian matrices. However, their adaptation to stochastic optimization, particularly in deep learning contexts, remains challenging due to noisy gradient estimates and nonconvex objectives. In this thesis, we propose a stochastic limited-memory BFGS (LBFGS) optimizer designed specifically for training deep neural networks. Our method introduces a novel curvature selection strategy that utilizes an update frequency mechanism to select curvature pairs exhibiting the highest curvature, effectively addressing stochastic and nonconvex issues. Additionally, we integrate momentum to speed up the convergence. Experimental results demonstrate that our approach significantly outperforms the existing stochastic LBFGS method (oLBFGS) and remains competitive with widely used deep learning optimizers such as SGD with momentum (SGDM), Adam, and Shampoo on standard convex and non-convex image classification benchmarks.

Keywords: Deep Learning, Stochastic Optimization, Quasi-Newton Methods

doi:10.6342/NTU202504473

Contents

doi:10.6342/NTU202504473

doi:10.6342/NTU202504473

List of Figures

doi:10.6342/NTU202504473

List of Tables

doi:10.6342/NTU202504473

Chapter 1

Introduction

In traditional machine learning applications, the task of training a model is to solve an unconstrained optimization problem:

$$\arg\min_{x\in\mathbb{R}^d} f(x), \tag{1.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a twice-differentiable, nonconvex objective function, and $x \in \mathbb{R}^d$ denotes the model parameters.

To define the objective function, we first specify the training dataset. Let the dataset D consist of n training data points: $D = \{d_i\}_{i=1}^n$, where each d_i denotes the i-th data point. The objective function f takes the form of the average of individual loss values:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_{d_i}(x), \tag{1.2}$$

where $f_{d_i}(x)$ denotes the loss associated with data point d_i. The literature commonly refers to this structure of f as a *finite-sum problem*.

An optimization algorithm (or optimizer) generates a sequence of iterates $\{x_t\}_{t=1}^T$ by minimizing an approximation of the objective function f at each step. We let t denote the iteration index and T be the total number of iterations. A natural question is how to choose an appropriate approximation of f. Motivated by the second-order Taylor expansion, we approximate f at the current iterate x_t using a quadratic model $g_t : \mathbb{R}^d \to \mathbb{R}$:

doi:10.6342/NTU202504473

$$f(x_t + p) \approx g_t(p) = f(x_t) + p^\top \nabla f(x_t) + \frac{1}{2} p^\top B_t p, \tag{1.3}$$

where $f(x_t) \in \mathbb{R}$ denotes the function value at x_t, $\nabla f(x_t) \in \mathbb{R}^d$ denotes the gradient at x_t, $B_t \in \mathbb{R}^{d \times d}$ is a matrix required to be designed, and $p \in \mathbb{R}^d$ is a candidate step direction. The gradient $\nabla f(x_t)$ is

$$\nabla f(x_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_{d_i}(x_t). \tag{1.4}$$

The search direction is the minimizer of the quadratic model $g_t(p)$. If B_t is positive definite, we can compute the minimizer as

$$p_t = -B_t^{-1} \nabla f(x_t) = -H_t \nabla f(x_t). \tag{1.5}$$

Here, H_t denotes the inverse of B_t. We explain positive definiteness in Definition 1.

Definition 1 (Positive Definiteness)**.** *A matrix $A \in \mathbb{R}^{d \times d}$ is* positive definite *if*

$$v^\top A v > 0, \quad \forall v \in \mathbb{R}^d \setminus \{\mathbf{0}\}. \tag{1.6}$$

With the obtained direction p_t, the update rule at the current iterate x_t is

$$x_{t+1} = x_t + \alpha_t p_t, \tag{1.7}$$

where α_t is the step size. The search direction determines the direction to move from x_t, while the step size controls the magnitude of the movement.

If B_t is the identity matrix, the optimization method is first-order; if B_t equals the Hessian matrix $\nabla^2 f(x_t)$ or its approximation, the method is second-order (Newton).

The Newton method is well known for its fast theoretical convergence [1]. Due to this advantage, traditional machine learning libraries, such as LIBLINEAR [2], utilize the Newton method or its variants to solve optimization problems efficiently. In addition to the Newton method, quasi-Newton methods also offer strong theoretical guarantees [1; 3]. Among them, the BFGS method [4; 5; 6; 7; 1] is particularly popular. However, BFGS is

doi:10.6342/NTU202504473

memory-intensive. Therefore, we focus on its limited-memory variant (LBFGS) [8; 1].

Meanwhile, deep neural networks have demonstrated exceptional performance across a wide range of tasks, including computer vision, speech, and natural language processing [9; 10; 11; 12; 13; 14; 15]. Currently, stochastic first-order methods, more precisely methods that use only diagonal information of the Hessian approximation [16; 17; 18], are the most widely used optimizers for training deep neural networks. In comparison, second-order methods, such as Newton and quasi-Newton methods, have not been widely adopted, despite having faster theoretical convergence. Furthermore, empirical evidence suggests that methods going beyond diagonal Hessian approximations can lead to improved model performance [15]. Motivated by these insights, we aim to revisit LBFGS in the context of training deep neural networks. However, applying LBFGS is challenging due to the stochastic nature of deep learning optimization.

We summarize our main results as follows:

1. We propose a novel stochastic LBFGS method tailored for training deep neural networks. The method integrates an update frequency mechanism that strategically selects curvature pairs exhibiting the highest curvature. This design enhances the robustness and performance of LBFGS in stochastic and nonconvex optimization settings.

2. We incorporate cautious updates and momentum into our stochastic LBFGS framework, significantly enhancing its convergence performance and stability in practical deep learning settings.

3. Empirical evaluations on diverse optimization tasks, ranging from convex to nonconvex image classification problems, demonstrate the superiority of our proposed method compared to the existing stochastic LBFGS variant (oLBFGS [19; 20; 21; 22]). Our method achieves competitive performance relative to state-of-the-art deep learning optimizers, including SGDM [23; 24; 25; 26; 27], Adam [17], and Shampoo [28; 29].

doi:10.6342/NTU202504473

Chapter 2

Quasi-Newton Methods

To understand quasi-Newton methods, we first introduce the *secant equation*, which at $(t+1)$-th iteration is

$$y_t = B_t s_t \quad \text{or} \quad H_t y_t = s_t, \tag{2.1}$$

where $s_t \coloneqq x_t - x_{t-1}$ and $y_t \coloneqq \nabla f(x_t) - \nabla f(x_{t-1})$. The pair (s_t, y_t) is referred to as a *curvature pair*.

There are two main motivations for the secant equation:

1. **First-order Taylor approximation of ∇f**

$$\nabla f(x_{t-1}) \approx \nabla f(x_t) + \nabla^2 f(x_t)(x_{t-1} - x_t) \tag{2.2}$$

$$\iff \nabla f(x_{t-1}) - \nabla f(x_t) \approx \nabla^2 f(x_t)(x_{t-1} - x_t) \tag{2.3}$$

$$\iff y_t \approx \nabla^2 f(x_t) s_t. \tag{2.4}$$

This implies that the Hessian matrix approximately satisfies the secant equation.

doi:10.6342/NTU202504473

2. **Fundamental Theorem of Calculus:**

$$\nabla f(x_t) - \nabla f(x_{t-1}) = \int_0^1 \nabla^2 f(x_{t-1} + z(x_t - x_{t-1}))\,(x_t - x_{t-1})\,dz \tag{2.5}$$

$$\iff \nabla f(x_t) - \nabla f(x_{t-1}) = \int_0^1 \nabla^2 f(x_{t-1} + z(x_t - x_{t-1}))\,dz\,(x_t - x_{t-1}) \tag{2.6}$$

$$\iff y_t = \int_0^1 \nabla^2 f(x_{t-1} + z\alpha_{t-1}p_{t-1})\,dz\,s_t, \tag{2.7}$$

where the integral represents the *average Hessian* matrix between x_{t-1} and x_t[1].

This shows that the average Hessian matrix satisfies the secant equation exactly.

Therefore, we aim to construct a matrix that satisfies the secant equation and, ideally, has a relationship with the Hessian matrix. A *quasi-Newton method* is any method that uses the secant equation to construct either a Hessian approximation B_t or an inverse Hessian approximation H_t.

A quasi-Newton method may impose the following conditions to uniquely determine the update of the Hessian approximation $B_t \in \mathbb{R}^{d\times d}$:

1. The updated matrix B_t should be as close as possible to the previous approximation $B_{t-1} \in \mathbb{R}^{d\times d}$.
2. The updated matrix B_t must be symmetric and positive definite.
3. The updated matrix B_t must satisfy the secant equation.

Formally, this leads to the following optimization problem:

$$\begin{aligned} \underset{B\in\mathbb{R}^{d\times d}}{\arg\min} \quad & \|B - B_{t-1}\|_W \\ \text{subject to} \quad & B = B^\top, \quad Bs_t = y_t, \end{aligned} \tag{2.8}$$

where the norm $\|\cdot\|_W$ evaluates how close the updated matrix B_t is to the previous matrix B_{t-1} by the *weighted Frobenius norm*, defined in Definition 2.

[1] Average Hessian is the terminology used by [1].

doi:10.6342/NTU202504473

Definition 2 (Weighted Frobenius Norm)**.** *Let $A \in \mathbb{R}^{d\times d}$ be a matrix and $W \in \mathbb{R}^{d\times d}$ be a positive definite matrix. The weighted Frobenius norm of A with respect to W is*

$$\|A\|_W \coloneqq \left\|W^{1/2}AW^{1/2}\right\|_F, \tag{2.9}$$

where the Frobenius norm $\|C\|_F$ for a matrix $C = [c_{ij}] \in \mathbb{R}^{d\times d}$ is

$$\|C\|_F \coloneqq \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^2}. \tag{2.10}$$

The definition of a symmetric matrix is in Definition 3.

Definition 3 (Symmetry)**.** *A matrix $A \in \mathbb{R}^{d\times d}$ is symmetric if $A = A^\top$.*

The constraint $B = B^\top$ in Eq. (2.8) ensures that B is symmetric. Although we do not explicitly impose a positive definiteness constraint in Eq. (2.8), the resulting solution will be positive definite under certain conditions. We will discuss this important property later. The reason for considering a positive definite B_t is to guarantee that the search direction p_t, defined in Eq. (1.5), is a descent direction. Definition 4 presents the definition of a descent direction.

Definition 4 (Descent Direction)**.** *A vector $p_t \in \mathbb{R}^d$ is a descent direction at $(t+1)$-th iteration if $p_t^\top \nabla f(x_t) < 0$.*

To understand why the search direction p_t is reasonable, recall that the quadratic model $g_t(p)$ approximates $f(x_t + p)$. By substituting p_t into Eq. (1.3), we obtain

$$f(x_t + p_t) \approx f(x_t) - \tfrac{1}{2}\nabla f(x_t)^\top H_t \nabla f(x_t). \tag{2.11}$$

Since $\nabla f(x_t)^\top H_t \nabla f(x_t) > 0$ due to the positive definiteness of H_t (which holds since B_t is positive definite, and its inverse $H_t = B_t^{-1}$ is therefore also positive definite), it is highly likely that $f(x_t + p_t) < f(x_t)$.

To derive the solution to Eq. (2.8), we apply the theorem stated in Corollary 4.3 of Dennis and Schnabel [30].

doi:10.6342/NTU202504473

Theorem 1. *Let $s', y' \in \mathbb{R}^d$, let $A \in \mathbb{R}^{d\times d}$ be symmetric, and let $\bar{A} \in \mathbb{R}^{d\times d}$ be symmetric and positive definite. Then, the unique solution to*

$$\begin{aligned} \underset{A_+\in\mathbb{R}^{d\times d}}{\arg\min} \quad & \|A_+ - A\|_{\bar{A}} \\ \textit{subject to} \quad & A_+ = A_+^\top, \quad A_+ s' = y' \end{aligned} \tag{2.12}$$

is

$$A_+ = A + \frac{u'v'^\top + v'u'^\top}{v'^\top s'} - \frac{s'^\top u'}{(v'^\top s')^2} v'v'^\top, \tag{2.13}$$

where $u' = y' - As'$ and $v' = \bar{A}^{-1}s'$.

For applying Theorem 1 to Eq. (2.8), we choose a symmetric and positive definite matrix W that satisfies the secant equation, such that $y_t = W^{-1}s_t$. Let $\bar{A} = W$, $s' = s_t$, $y' = y_t$, $A = B_{t-1}$, and $A_+ = B_t$. Then,

$$\begin{aligned} B_t &= B_{t-1} + \frac{(y_t - B_{t-1}s_t)y_t^\top + y_t(y_t - B_{t-1}s_t)^\top}{y_t^\top s_t} - \frac{s_t^\top(y_t - B_{t-1}s_t)}{(y_t^\top s_t)^2} y_t y_t^\top \\ &= B_{t-1} + \frac{y_t y_t^\top - B_{t-1}s_t y_t^\top + y_t y_t^\top - y_t s_t^\top B_{t-1}}{y_t^\top s_t} - \frac{s_t^\top y_t - s_t^\top B_{t-1}s_t}{(y_t^\top s_t)^2} y_t y_t^\top \\ &= B_{t-1} + \frac{y_t y_t^\top}{y_t^\top s_t} - \frac{B_{t-1}s_t y_t^\top}{y_t^\top s_t} + \frac{y_t y_t^\top}{y_t^\top s_t} - \frac{y_t s_t^\top B_{t-1}}{y_t^\top s_t} - \frac{s_t^\top y_t}{(y_t^\top s_t)^2} y_t y_t^\top + \frac{s_t^\top B_{t-1}s_t}{(y_t^\top s_t)^2} y_t y_t^\top \\ &= B_{t-1} - \frac{B_{t-1}s_t y_t^\top}{y_t^\top s_t} - \frac{y_t s_t^\top B_{t-1}}{y_t^\top s_t} + \frac{y_t s_t^\top B_{t-1}s_t y_t^\top}{(y_t^\top s_t)^2} + \frac{y_t y_t^\top}{y_t^\top s_t} \\ &= \left(B_{t-1} - \frac{y_t s_t^\top B_{t-1}}{y_t^\top s_t}\right)\left(I - \frac{s_t y_t^\top}{y_t^\top s_t}\right) + \frac{y_t y_t^\top}{y_t^\top s_t} \\ &= \left(I - \frac{y_t s_t^\top}{y_t^\top s_t}\right) B_{t-1} \left(I - \frac{s_t y_t^\top}{y_t^\top s_t}\right) + \frac{y_t y_t^\top}{y_t^\top s_t}. \end{aligned} \tag{2.14}$$

Finally, Eq. (2.14) yields the *Davidon–Fletcher–Powell (DFP)* update formula [1]:

$$\begin{aligned} B_t^{\text{DFP}} &= \left(I - \rho_t y_t s_t^\top\right) B_{t-1}^{\text{DFP}} \left(I - \rho_t s_t y_t^\top\right) + \rho_t y_t y_t^\top \\ \rho_t &= \frac{1}{y_t^\top s_t}. \end{aligned} \tag{2.15}$$

Likewise, if we want an approximation of the inverse Hessian H_t, the problem be-

doi:10.6342/NTU202504473

comes:

$$\begin{aligned} \underset{H\in\mathbb{R}^{d\times d}}{\arg\min} \quad & \|H - H_{t-1}\|_W \\ \text{subject to} \quad & H = H^\top, \quad Hy_t = s_t. \end{aligned} \tag{2.16}$$

Similarly, to apply Theorem 1, we choose a symmetric and positive definite matrix W that satisfies the secant equation, such that $s_t = W^{-1}y_t$. Let $\bar{A} = W$, $s' = y_t$, $y' = s_t$, $A = H_{t-1}$, and $A_+ = H_t$. Then,

$$\begin{aligned} H_t &= H_{t-1} + \frac{(s_t - H_{t-1}y_t)s_t^\top + s_t(s_t - H_{t-1}y_t)^\top}{s_t^\top y_t} - \frac{y_t^\top(s_t - H_{t-1}y_t)}{(s_t^\top y_t)^2}s_t s_t^\top \\ &= H_{t-1} + \frac{s_t s_t^\top - H_{t-1}y_t s_t^\top + s_t s_t^\top - s_t y_t^\top H_{t-1}}{s_t^\top y_t} - \frac{y_t^\top s_t - y_t^\top H_{t-1}y_t}{(s_t^\top y_t)^2}s_t s_t^\top \\ &= H_{t-1} + \frac{s_t s_t^\top}{s_t^\top y_t} - \frac{H_{t-1}y_t s_t^\top}{s_t^\top y_t} + \frac{s_t s_t^\top}{s_t^\top y_t} - \frac{s_t y_t^\top H_{t-1}}{s_t^\top y_t} - \frac{y_t^\top s_t}{(s_t^\top y_t)^2}s_t s_t^\top + \frac{y_t^\top H_{t-1}y_t}{(s_t^\top y_t)^2}s_t s_t^\top \\ &= H_{t-1} - \frac{H_{t-1}y_t s_t^\top}{s_t^\top y_t} - \frac{s_t y_t^\top H_{t-1}}{s_t^\top y_t} + \frac{s_t y_t^\top H_{t-1}y_t s_t^\top}{(s_t^\top y_t)^2} + \frac{s_t s_t^\top}{s_t^\top y_t} \\ &= \left(H_{t-1} - \frac{s_t y_t^\top H_{t-1}}{s_t^\top y_t}\right)\left(I - \frac{y_t s_t^\top}{s_t^\top y_t}\right) + \frac{s_t s_t^\top}{s_t^\top y_t} \\ &= \left(I - \frac{s_t y_t^\top}{s_t^\top y_t}\right) H_{t-1} \left(I - \frac{y_t s_t^\top}{s_t^\top y_t}\right) + \frac{s_t s_t^\top}{s_t^\top y_t}. \end{aligned} \tag{2.17}$$

Finally, Eq. (2.17) gives the *Broyden–Fletcher–Goldfarb–Shanno (BFGS)* update formula [1]:

$$\begin{aligned} H_t^{\text{BFGS}} &= \left(I - \rho_t s_t y_t^\top\right) H_{t-1}^{\text{BFGS}} \left(I - \rho_t y_t s_t^\top\right) + \rho_t s_t s_t^\top \\ \rho_t &= \frac{1}{y_t^\top s_t}. \end{aligned} \tag{2.18}$$

Note that we derive both the DFP and BFGS updates using the derivation in Dennis and Schnabel [30], which is different from the original derivation in Dennis and Moré [3].

One important property of the matrices generated by BFGS and DFP is that they remain positive definite if the curvature condition is satisfied:

$$s_t^\top y_t > 0. \tag{2.19}$$

We will discuss how to ensure Eq. (2.19) later. We first examine the positive definiteness

doi:10.6342/NTU202504473

of the matrix produced by BFGS:

$$
\begin{aligned}
v^\top H_t^{\text{BFGS}} v &= v^\top \left(I - \frac{s_t y_t^\top}{y_t^\top s_t}\right) H_{t-1}^{\text{BFGS}} \left(I - \frac{y_t s_t^\top}{y_t^\top s_t}\right) v + v^\top \frac{s_t s_t^\top}{y_t^\top s_t} v \\
&= u^\top H_{t-1}^{\text{BFGS}} u + \frac{a^2}{y_t^\top s_t}, \quad \forall v \in \mathbb{R}^d \setminus \{\mathbf{0}\}, \\
&\text{where } u = \left(I - \frac{y_t s_t^\top}{y_t^\top s_t}\right) v \in \mathbb{R}^d \text{ and } a = v^\top s_t \in \mathbb{R}.
\end{aligned} \tag{2.20}
$$

Since H_{t-1}^{BFGS} is positive definite, we have $w^\top H_{t-1}^{\text{BFGS}} w > 0, \quad \forall w \in \mathbb{R}^d \setminus \{\mathbf{0}\}$. This property implies $u^\top H_{t-1}^{\text{BFGS}} u > 0$. Moreover, $a^2 > 0$ and we assume that Eq. (2.19) holds. Therefore, $v^\top H_t^{\text{BFGS}} v > 0, \quad \forall v \in \mathbb{R}^d \setminus \{\mathbf{0}\}$, which implies that the updated inverse Hessian approximation H_t^{BFGS} remains positive definite.

We now examine the positive definiteness of the matrix produced by DFP:

$$
\begin{aligned}
v^\top B_t^{\text{DFP}} v &= v^\top \left(I - \frac{y_t s_t^\top}{y_t^\top s_t}\right) B_{t-1}^{\text{DFP}} \left(I - \frac{s_t y_t^\top}{y_t^\top s_t}\right) v + v^\top \frac{y_t y_t^\top}{y_t^\top s_t} v \\
&= u^\top B_{t-1}^{\text{DFP}} u + \frac{a^2}{y_t^\top s_t}, \quad \forall v \in \mathbb{R}^d \setminus \{\mathbf{0}\}, \\
&\text{where } u = \left(I - \frac{s_t y_t^\top}{y_t^\top s_t}\right) v \in \mathbb{R}^d \text{ and } a = v^\top y_t \in \mathbb{R}.
\end{aligned} \tag{2.21}
$$

By a similar argument, we conclude that the Hessian approximation B_t^{DFP} in the DFP update remains positive definite.

DFP and BFGS are also known as rank-two update methods. The proof is in Appendix A. Definition 5 provides the definition of a rank-k update.

Definition 5 (Rank-k)**.** *A matrix $B \in \mathbb{R}^{d\times d}$ is a rank-k update of $A \in \mathbb{R}^{d\times d}$ if*

$$
B = A + \sum_{i=1}^{k} c_i v_i v_i^\top, \tag{2.22}
$$

where each $c_i \in \mathbb{R}$ is a scalar and each $v_i \in \mathbb{R}^d$ is a vector.

Unlike rank-two update methods such as DFP and BFGS, which guarantee that the updated matrix remains positive definite, the *symmetric-rank-one (SR1)* method employs

doi:10.6342/NTU202504473

a rank-one update to maintain B_t^{SR1} [1]:

$$B_t^{\text{SR1}} = B_{t-1}^{\text{SR1}} + \underbrace{\frac{(y_t - B_{t-1}^{\text{SR1}} s_t)(y_t - B_{t-1}^{\text{SR1}} s_t)^\top}{(y_t - B_{t-1}^{\text{SR1}} s_t)^\top s_t}}_{\text{rank-1}}. \tag{2.23}$$

However, the scalar denominator $(y_t - B_{t-1}^{\text{SR1}} s_t)^\top s_t$ can be negative. Therefore, the positive definiteness of B_t^{SR1} is not guaranteed. To enforce positive definiteness, one can apply Hessian modifications such as eigenvalue or shift adjustments [1]. Nevertheless, these operations are computationally expensive [31].

Therefore, we focus on quasi-Newton methods that inherently preserve positive definiteness. Among them, the BFGS method is the most widely used. In the following, we denote H_t^{BFGS} and B_t^{BFGS} simply as H_t and B_t, respectively.

doi:10.6342/NTU202504473

Chapter 3

Limited-memory BFGS Methods

Since the BFGS method maintains a matrix $H \in \mathbb{R}^{d\times d}$, its space complexity is $\mathcal{O}(d^2)$, which makes it highly memory-intensive for large-scale optimization problems. The definition of Big-O notation is in Definition 6

Definition 6 (Big-O Notation)**.** *Let $f, g : \mathbb{R} \to \mathbb{R}$ be functions. The notation $f(x) = \mathcal{O}(g(x))$ means that there exist a constant $c > 0$ and a point $x_0 \in \mathbb{R}$ such that*

$$|f(x)| \leq c \cdot |g(x)| \quad \forall x \geq x_0. \tag{3.1}$$

To mitigate the issue of $\mathcal{O}(d^2)$ space, one can store all past curvature pairs (s_i, y_i) for all $i = 1, \ldots, t$ instead of retaining the full matrix H. Recall that the BFGS approximation of the inverse Hessian approximation H_t is

$$\begin{aligned} H_t &= V_t^\top H_{t-1} V_t + \rho_t s_t s_t^\top \\ \rho_t &= \frac{1}{y_t^\top s_t} \quad V_t = I - \rho_t y_t s_t^\top. \end{aligned} \tag{3.2}$$

doi:10.6342/NTU202504473

By recursively applying Eq. (3.2), we obtain the following expansion:

$$
\begin{aligned}
H_t = &\left(V_t^\top \cdots V_1^\top\right) H_0 \left(V_1 \cdots V_t\right) \\
&+ \rho_1 \left(V_t^\top \cdots V_2^\top\right) s_1 s_1^\top \left(V_2 \cdots V_t\right) \\
&+ \rho_2 \left(V_t^\top \cdots V_3^\top\right) s_2 s_2^\top \left(V_3 \cdots V_t\right) \\
&+ \cdots \\
&+ \rho_{t-2} \left(V_t^\top V_{t-1}^\top\right) s_{t-2} s_{t-2}^\top \left(V_{t-1} V_t\right) \\
&+ \rho_{t-1} V_t^\top s_{t-1} s_{t-1}^\top V_t \\
&+ \rho_t s_t s_t^\top .
\end{aligned} \tag{3.3}
$$

We can initialize the inverse Hessian approximation H_0 as a scaled identity matrix, i.e., $H_0 = cI$ with $c > 0$. For this approach, storing all previous curvature pairs incurs a space complexity of $\mathcal{O}(td)$, as there are t curvature pairs and each pair consists of two vectors in $\mathbb{R}^d$. Since t denotes the $(t+1)$-th iteration, the space complexity grows linearly with the number of iterations, which can become prohibitively large in practice.

To alleviate this issue, Liu and Nocedal [8] propose the Limited-memory BFGS (LBFGS) method, which stores only the most recent h curvature pairs. Formally, the limited-memory construction of H_t is:

$$
\begin{aligned}
H_t = &\left(V_t^\top \cdots V_{t-h+1}^\top\right) H_t^0 \left(V_{t-h+1} \cdots V_t\right) \\
&+ \rho_{t-h+1} \left(V_t^\top \cdots V_{t-h+2}^\top\right) s_{t-h+1} s_{t-h+1}^\top \left(V_{t-h+2} \cdots V_t\right) \\
&+ \rho_{t-h+2} \left(V_t^\top \cdots V_{t-h+3}^\top\right) s_{t-h+2} s_{t-h+2}^\top \left(V_{t-h+3} \cdots V_t\right) \\
&+ \cdots \\
&+ \rho_{t-2} \left(V_t^\top V_{t-1}^\top\right) s_{t-2} s_{t-2}^\top \left(V_{t-1} V_t\right) \\
&+ \rho_{t-1} V_t^\top s_{t-1} s_{t-1}^\top V_t \\
&+ \rho_t s_t s_t^\top .
\end{aligned} \tag{3.4}
$$

doi:10.6342/NTU202504473

The initial inverse Hessian approximation is

$$H_t^0 = \frac{y_t^\top s_t}{y_t^\top y_t} I, \tag{3.5}$$

as suggested by Nocedal and Wright [1]. Storing only the most recent h curvature pairs requires $\mathcal{O}(d)$ space, since h is a fixed constant[1]—typically between 5 and 20—and each curvature pair consists of two vectors in $\mathbb{R}^d$. Thus, the space complexity is generally acceptable. To apply the LBFGS method in practice, we do not explicitly construct H_t using Eq. (3.4), since our goal is to calculate the search direction defined in Eq. (1.5). Instead, an efficient algorithm that relies solely on vector operations is applicable to obtain the search direction. Algorithm 1 presents details of the procedure, which is commonly referred to as the two-loop recursion [1]. Subsequently, we derive this two-loop procedure for calculating $r = H_t v$ for any given $v \in \mathbb{R}^d$. From Algorithm 1, the first part of the two-loop recursion iteratively updates q by:

$$q \leftarrow q - a_i y_i = q - \rho_i s_i^\top q y_i = q - \rho_i y_i s_i^\top q = V_i q. \tag{3.6}$$

Based on the above recursion, at iteration index i of the first loop, we have

$$q = V_i \cdots V_t v. \tag{3.7}$$

After the first loop, the last q is $V_{t-h+1} \cdots V_t v$. From Eq. (3.4), this q will be used in the

[1] https://github.com/hjmshi/PyTorch-LBFGS

doi:10.6342/NTU202504473

first term of $H_t v$. For the second loop at the i-th iteration, we establish:

$$r \leftarrow r + s_i(a_i - b) = r - s_i b + s_i a_i = r - \rho_i s_i y_i^\top r + s_i a_i = V_i^\top r + s_i a_i \tag{3.8}$$

$$\leftarrow V_i^\top (V_{i-1}^\top r + s_{i-1} a_{i-1}) + s_i a_i = V_i^\top V_{i-1}^\top r + V_i^\top s_{i-1} a_{i-1} + s_i a_i \tag{3.9}$$

$$\cdots$$

$$\leftarrow V_i^\top \cdots V_{t-h+1}^\top r + V_i^\top \cdots V_{t-h+2}^\top s_{t-h+1} a_{t-h+1} + V_i^\top \cdots V_{t-h+3}^\top s_{t-h+2} a_{t-h+2}$$

$$+ \cdots$$

$$+ V_i^\top V_{i-1}^\top s_{i-2} a_{i-2} + V_i^\top s_{i-1} a_{i-1} + s_i a_i \tag{3.10}$$

$$= V_i^\top \cdots V_{t-h+1}^\top \underbrace{H_t^0 \overbrace{V_{t-h+1} \cdots V_t v}^{\text{last } q}}_{\text{initial } r}$$

$$+ V_i^\top \cdots V_{t-h+2}^\top s_{t-h+1} \underbrace{\rho_{t-h+1} s_{t-h+1}^\top V_{t-h+2} \cdots V_t v}_{a_{t-h+1}}$$

$$+ V_i^\top \cdots V_{t-h+3}^\top s_{t-h+2} \underbrace{\rho_{t-h+2} s_{t-h+2}^\top V_{t-h+3} \cdots V_t v}_{a_{t-h+2}}$$

$$+ \cdots$$

$$+ V_i^\top V_{i-1}^\top s_{i-2} \underbrace{\rho_{i-2} s_{i-2}^\top V_{i-1} \cdots V_t v}_{a_{i-2}}$$

$$+ V_i^\top s_{i-1} \underbrace{\rho_{i-1} s_{i-1}^\top V_i \cdots V_t v}_{a_{i-1}}$$

$$+ s_i \underbrace{\rho_i s_i^\top V_{i+1} \cdots V_t v}_{a_i}. \tag{3.11}$$

We derive Eq. (3.8), which then recursively generates Eq. (3.9), and so on, until we obtain Eq. (3.10). Recall that in the first loop, we have $a_i = \rho_i s_i^\top q$. Hence, Eq. (3.10) and Eq. (3.6) together imply Eq. (3.11). When $i = t$ in the second loop, Eq. (3.11) and Eq. (3.4) imply that the resulting vector r is in fact $H_t v$. Through the summation, we also see the need to maintain a_i for any $i = t - h + 1, \ldots, t$ from the first loop.

Now we have obtained the search direction p_t, so the remaining task is to select the step size α_t. In addition to ensuring a sufficient decrease in the function value, recall in Eq. (2.19), we must also ensure the positive definiteness of the inverse Hessian approxi-

doi:10.6342/NTU202504473

Algorithm 1 LBFGS_two_loop_recursion(H_{t-1}^0, S, Y, h, v)

Input: Initial inverse Hessian approximation H_{t-1}^0, curvature pair lists $S = \{s_j\}$ and $Y = \{y_j\}$, history size h, and vector $v \in \mathbb{R}^d$
$k \leftarrow |S|$
if $h > k$ **then**
    $h \leftarrow k$
end if
$t \leftarrow k$
$q \leftarrow v$
for $i = t, t-1, \ldots, t-h+1$ **do**
    $a_i \leftarrow \rho_i s_i^\top q$
    $q \leftarrow q - a_i y_i$
end for
$r \leftarrow H_{t-1}^0 q$
for $i = t-h+1, t-h+2, \ldots, t$ **do**
    $b \leftarrow \rho_i y_i^\top r$
    $r \leftarrow r + s_i (a_i - b)$
end for
Output: r

mation H_t. Practitioners traditionally enforce this condition using the Wolfe line search method [1].

Wolfe line search is an inexact line search strategy that imposes two conditions on the step size (or learning rate) α_t:

1. **Armijo condition:**

$$f(x_t + \alpha_t p_t) \leq f(x_t) + c_1 \alpha_t \nabla f(x_t)^\top p_t. \tag{3.12}$$

2. **Curvature condition**[2]**:**

$$\nabla f(x_t + \alpha_t p_t)^\top p_t \geq c_2 \nabla f(x_t)^\top p_t, \tag{3.13}$$

with $0 < c_1 < c_2 < 1$.

[2]This terminology may cause confusion with Eq. (2.19).

doi:10.6342/NTU202504473

Algorithm 2 The traditional LBFGS method

Input: Initial point x_0, history size h, number of iterations T
$S \leftarrow [\,], \quad Y \leftarrow [\,]$
for $t = 1, \ldots, T$ **do**
  $g_{t-1} \leftarrow \nabla f(x_{t-1})$
  if $t > 1$ **then**
    $s_{t-1} \leftarrow x_{t-1} - x_{t-2}, \quad y_{t-1} \leftarrow g_{t-1} - g_{t-2}$
    $S \leftarrow S \cup \{s_{t-1}\}, \quad Y \leftarrow Y \cup \{y_{t-1}\}$
    $H^0_{t-1} \leftarrow \frac{y_{t-1}^\top s_{t-1}}{y_{t-1}^\top y_{t-1}} I$
    $p_{t-1} \leftarrow -\text{LBFGS\_two\_loop\_recursion}(H^0_{t-1}, S, Y, h, g_{t-1})$
  else
    $p_{t-1} \leftarrow -g_{t-1}$
  end if
  Compute the step size α_{t-1} using the Wolfe line search procedure
  $x_t \leftarrow x_{t-1} + \alpha_{t-1} p_{t-1}$
end for
Output: x_T

We show that Eq. (3.13) can lead to the needed condition in Eq. (2.19):

$$\begin{aligned}
&\nabla f(x_t + \alpha_t p_t)^\top p_t \geq c_2 \nabla f(x_t)^\top p_t \\
\iff &\nabla f(x_t + \alpha_t p_t)^\top \alpha_t p_t \geq c_2 \nabla f(x_t)^\top \alpha_t p_t \\
\iff &\nabla f(x_t + \alpha_t p_t)^\top s_t \geq c_2 \nabla f(x_t)^\top \alpha_t p_t \\
\iff &\nabla f(x_t + \alpha_t p_t)^\top s_t - \nabla f(x_t)^\top s_t \geq c_2 \nabla f(x_t)^\top \alpha_t p_t - \nabla f(x_t)^\top \alpha_t p_t \\
\iff &y_t^\top s_t \geq (c_2 - 1)\alpha_t \nabla f(x_t)^\top p_t.
\end{aligned} \tag{3.14}$$

Therefore, if Eq. (3.13) holds, because p_t is a descent direction and $c_2 < 1$, we have $(c_2 - 1)\alpha_t \nabla f(x_t)^\top p_t > 0$ and obtain Eq. (2.19) through the last inequality of Eq. (3.14). The literature collectively refers to the Armijo and curvature conditions as the *Wolfe conditions*. Moreover, under certain assumptions, Lemma 3.1 of Nocedal and Wright [1] shows the existence of intervals that contain step sizes satisfying the Wolfe condition. We summarize the traditional LBFGS method in Algorithm 2.

doi:10.6342/NTU202504473

Chapter 4

Stochastic LBFGS Methods

With the rise of modern deep learning techniques such as self-supervised learning [32] and data augmentation [33], datasets have become enormously large. As a result, the finite-sum structure no longer holds [34], and computing the full gradient in Eq. (1.4) becomes computationally infeasible. We therefore reformulate the deterministic optimization problem in Eq. (1.1) as a stochastic optimization problem. Specifically, the goal becomes minimizing the expected function value over a data distribution:

$$\arg\min_{x\in\mathbb{R}^d} f\left(x\right) = \mathbb{E}_{\xi\sim\mathcal{D}}\left[f_\xi\left(x\right)\right], \tag{4.1}$$

where x denotes the model parameters, $\mathcal{D}$ is a data distribution over a space Ω, f_ξ is twice differentiable almost everywhere and nonconvex for all $\xi \in \Omega$.

We study whether LBFGS is applicable to solve Eq. (4.1). Our discussion indicates that the LBFGS method constructs curvature pairs (s, y) using the full gradient $\nabla f(x)$. Because only stochastic gradients $\nabla f_\xi(x)$ are available now, the inverse Hessian approximation H may be inaccurate and the optimization procedure may diverge [19]. Furthermore, classical line search methods, which select the step size α_t to ensure a sufficient decrease of the function value, require access to the full gradient $\nabla f(x)$ and the exact function value $f(x)$. The line search procedure in the BFGS method further guarantees the satisfaction of the curvature condition in Eq. (2.19). However, in stochastic optimization settings, the full gradient $\nabla f(x)$ and the exact function value $f(x)$ are generally unavailable, making

doi:10.6342/NTU202504473

traditional line search inapplicable.

In summary, to apply the LBFGS method to solve the stochastic optimization problem in Eq. (4.1), two major challenges are:

Challenge 1. *Inaccurate H from stochastic gradients*

Challenge 2. *The satisfaction of the curvature condition in Eq.* (2.19)

We will show that Challenge 2 is relatively easier to address by replacing line search with effective methods to enforce Eq. (2.19). On the other hand, Challenge 1 is more difficult. We review previous works that attempt to address these issues and highlight their limitations. We refer to LBFGS applied in the stochastic optimization setting as *Stochastic LBFGS*.

Line search methods, including their stochastic variants such as stochastic line search [22; 35; 36], are rarely adopted in stochastic LBFGS methods [19; 37; 38; 39; 40; 41; 20; 42; 21]. A common alternative is *learning rate scheduling*, which defines a function $\alpha(t) : \mathbb{R} \to \mathbb{R}$ to control the step size or learning rate at $(t+1)$-th iteration. One popular example is cosine learning rate scheduling [43]:

$$\alpha(t) = \alpha_{\text{min}} + \frac{1}{2}(\alpha_{\text{max}} - \alpha_{\text{min}}) \left(1 + \cos\left(\frac{t+1}{T}\pi\right)\right), \tag{4.2}$$

where α_{min} is the minimum learning rate, α_{max} is the initial learning rate, and T denotes the total number of iterations. Because learning rate schedules do not depend on the function value $f(x)$ or the full gradient $\nabla f(x)$, they are particularly well suited for stochastic optimization. As a result, stochastic LBFGS methods typically replace traditional line search procedures with learning rate schedules.

One classical approach to address Challenge 2 is *Powell's damping* [44], which replaces s_t with $\tilde{s}_t$:

$$\tilde{s}_t = \theta s_t + (1-\theta) H_{t-1} y_t, \quad \text{where } \theta = \begin{cases} \frac{(1-\epsilon)\, y_t^\top H_{t-1} y_t}{y_t^\top H_{t-1} y_t - s_t^\top y_t}, & \text{if } y_t^\top s_t < \epsilon\, y_t^\top H_{t-1} y_t, \\ 1, & \text{if } y_t^\top s_t \geq \epsilon\, y_t^\top H_{t-1} y_t, \end{cases} \tag{4.3}$$

doi:10.6342/NTU202504473

where $\epsilon > 0$ is a predefined hyperparameter. When $y_t^\top s_t < \epsilon\, y_t^\top H_{t-1} y_t$, it follows that:

$$\begin{aligned} y_t^\top \tilde{s}_t &= \theta\, y_t^\top s_t + (1-\theta)\, y_t^\top H_{t-1} y_t \\ &= \theta \left(y_t^\top s_t - y_t^\top H_{t-1} y_t \right) + y_t^\top H_{t-1} y_t \\ &= \frac{(1-\epsilon)\, y_t^\top H_{t-1} y_t}{y_t^\top H_{t-1} y_t - s_t^\top y_t} \left(y_t^\top s_t - y_t^\top H_{t-1} y_t \right) + y_t^\top H_{t-1} y_t \\ &= -(1-\epsilon)\, y_t^\top H_{t-1} y_t + y_t^\top H_{t-1} y_t \\ &= \epsilon\, y_t^\top H_{t-1} y_t \\ &> y_t^\top s_t. \end{aligned} \tag{4.4}$$

Note that since H_{t-1} is positive definite,

$$y_t^\top \tilde{s}_t = \epsilon\, y_t^\top H_{t-1} y_t > 0. \tag{4.5}$$

Moreover, if $y_t^\top s_t \geq \epsilon\, y_t^\top H_{t-1} y_t$, then

$$y_t^\top \tilde{s}_t = y_t^\top s_t \geq \epsilon\, y_t^\top H_{t-1} y_t > 0. \tag{4.6}$$

Thus, Powell's damping ensures that the resulting curvature $y_t^\top \tilde{s}_t$ is always greater than or equal to the original curvature $y_t^\top s_t$ and strictly positive. Furthermore, one can always choose $\epsilon \geq \frac{(c_2-1)\,\alpha_t \nabla f(x_t)^\top p_t}{y_t^\top H_{t-1} y_t}$ such that the curvature condition in Eq. (3.13) is satisfied. Specifically,

$$\epsilon \geq \frac{(c_2-1)\,\alpha_t \nabla f(x_t)^\top p_t}{y_t^\top H_{t-1} y_t} \tag{4.7}$$

$$\iff \epsilon\, y_t^\top H_{t-1} y_t \geq (c_2-1)\,\alpha_t \nabla f(x_t)^\top p_t \tag{4.8}$$

$$\implies y_t^\top \tilde{s}_t \geq (c_2-1)\,\alpha_t \nabla f(x_t)^\top p_t, \tag{4.9}$$

as shown in Eq. (4.5) and Eq. (4.6).

Despite its advantage, Powell's damping requires tuning the hyperparameter ϵ, which can be difficult in practice. Large values introduce excessive noise into the inverse Hessian approximation H, while small values may fail to adequately safeguard the curvature con-

doi:10.6342/NTU202504473

dition, potentially causing divergence. Therefore, we do not adopt Powell's damping.

Another approach to address Challenge 2 is the *cautious update* strategy [45]. Proposed initially to address convergence issues in the BFGS method for nonconvex optimization, the key idea is to skip the inverse Hessian approximation update when the curvature $y^\top s$ is too small. Formally, we update the inverse Hessian approximation H, or construct a new curvature pair (s, y), only if

$$y^\top s \geq \epsilon \|s\|^2, \tag{4.10}$$

where $\epsilon > 0$ is a predetermined hyperparameter. The cautious update effectively stabilizes the optimization process in cases where the curvature condition in Eq. (2.19) may not hold [21; 22]. For this reason, we incorporate the cautious update into our method.

In summary, we replace line search with learning rate scheduling and integrate the cautious update in Eq. (4.10) into our stochastic LBFGS method in Algorithm 3 to handle Challenge 2 robustly.

We review previous works that attempt to address Challenge 1. To mitigate this issue, Schraudolph et al. [19] propose the *data consistency* scheme, which aims to reduce the noise in y introduced by stochastic gradients due to inconsistent sampling. This scheme ensures that y_t uses the same data sample across two consecutive iterates:

$$y_t = \nabla f_{\xi_t}(x_t) - \nabla f_{\xi_t}(x_{t-1}), \tag{4.11}$$

where ξ_t represents the identical data samples across two successive iterations. Mokhtari and Ribeiro [42]; Bordes et al. [46]; Mokhtari and Ribeiro [20]; Berahas et al. [21]; Bollapragada et al. [22] adopt this approach. In addition to employing data consistency to address the noise introduced by stochastic gradients, another critical challenge in stochastic LBFGS is the presence of extremely small eigenvalues in the inverse Hessian approximation H_t. This problem can cause the optimizer to diverge. To mitigate this difficulty, Mokhtari and Ribeiro [42]; Schraudolph et al. [19] introduce a hyperparameter specifically designed to control the smallest eigenvalue. Specifically, they add a scaled identity

doi:10.6342/NTU202504473

Algorithm 3 Cautious stochastic LBFGS method

Input: Initial point x_0, learning rate schedule $\alpha(\cdot)$, history size h, threshold ϵ, number of iterations T
$S \leftarrow [\,], \quad Y \leftarrow [\,]$
$k \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  Sample $\xi_{t-1} \sim \mathcal{D}$
  $g_{t-1} \leftarrow \nabla f_{\xi_{t-1}}(x_{t-1})$
  if $t > 1$ **then**
    $s_{k+1} \leftarrow x_{t-1} - x_{t-2}, \quad y_{k+1} \leftarrow g_{t-1} - g_{t-2}$
    if $y_{k+1}^\top s_{k+1} > \epsilon s_{k+1}^\top s_{k+1}$ **then**
      $S \leftarrow S \cup \{s_{k+1}\}, \quad Y \leftarrow Y \cup \{y_{k+1}\}$
      $k \leftarrow k + 1$
    end if
  end if
  if $k > 0$ **then**
    $H_{t-1}^0 \leftarrow \frac{y_k^\top s_k}{y_k^\top y_k} I$
    $p_{t-1} \leftarrow -\text{LBFGS\_two\_loop\_recursion}(H_{t-1}^0, S, Y, h, g_{t-1})$
  else
    $p_{t-1} \leftarrow -g_{t-1}$
  end if
  $x_t \leftarrow x_{t-1} + \alpha(t-1)p_{t-1}$
end for
Output: x_T

matrix to H_t. This technique is also known as Hessian modification [1]. However, tuning this additional hyperparameter is nontrivial. The difficulty is analogous to the challenge of selecting ϵ in Powell's damping Eq. (4.3), where one must balance safeguarding curvature conditions with preserving second-order information. Moreover, the data consistency scheme incurs an additional computational cost for computing the stochastic gradient of the previous iterate $\nabla f_{\xi_t}(x_{t-1})$ at each iteration. More importantly, past works have shown that the data consistency scheme is ineffective in deep learning [22]. For these reasons, we do not adopt either the data consistency scheme or the Hessian modification strategy for controlling the eigenvalues of the inverse Hessian approximation.

Besides data consistency, another research direction involves approximating y using the first-order Taylor expansion of ∇f:

$$y_t \approx \nabla^2 f(x_t)(x_t - x_{t-1}) \approx \nabla^2 f_{\xi_t^H}(x_t) s_t, \tag{4.12}$$

doi:10.6342/NTU202504473

where ξ_t^H denotes the data sample used to compute the stochastic Hessian matrix $\nabla^2 f_{\xi_t^H}(x_t)$. Byrd et al. [37] first propose this approach (SQN), and Moritz et al. [38] extend SQN by applying SVRG [47], a popular method that reduces the variance of stochastic gradient, to the stochastic gradient computation. Although computing the stochastic Hessian matrix can be expensive, SQN mitigates this cost by employing an update frequency strategy: updating curvature pairs once every L iterations, thereby amortizing the per-iteration cost. Furthermore, SQN leverages Hessian-vector products to reduce memory usage, as the stochastic Hessian matrix is never explicitly constructed. Owing to the Hessian-free property of the logistic regression problem, SQN can compute these products efficiently. In the context of deep learning, Dagréou et al. [48] compute Hessian-vector products within acceptable time and memory constraints. However, Eq. (2.19) may not be satisfied when using Eq. (4.12). As a result, applying the first-order Taylor expansion of ∇f to approximate y in nonconvex problems poses the substantial challenge of ensuring the positive definiteness of the inverse Hessian approximation H, representing a critical limitation of this method in deep learning. Thus, we do not adopt this approach in our proposed stochastic LBFGS method.

doi:10.6342/NTU202504473

Chapter 5

Proposed Stochastic LBFGS Method

We propose a solution to address Challenge 1 discussed in Chapter 4, motivated by two key ideas:

1. **Update frequency L:** Moritz et al. [38] have pointed out the advantages of introducing update frequency into stochastic LBFGS methods. More recently, Niu et al. [40] also incorporate update frequency in their development of stochastic LBFGS algorithms. Following this insight, we attempt to construct a new curvature pair (s, y) once every L iterations for updating the inverse Hessian approximation H.

2. **High curvature value $y^\top s$:** Eq. (4.10) emphasizes the importance of setting a threshold to ensure that the curvature remains sufficiently large. To this end, we aim to construct curvature pairs that yield the largest possible value of $y^\top s$.

Motivated by these insights, we propose integrating these two strategies by selecting the curvature pair (s, y) with the highest curvature value $y^\top s$ among the iterates collected over L steps. Specifically, the update frequency mechanism suggests constructing curvature pairs from the iterates $x_{t-L}, \ldots, x_t$ and their corresponding stochastic gradients $\nabla f_{\xi_{t-L}}(x_{t-L}), \ldots, \nabla f_{\xi_t}(x_t)$. However, this results in multiple candidate curvature pairs (s, y), each constructed from different combinations of two iterates within the L-step window—for example, $s \leftarrow x_t - x_{t-L}$ and $y \leftarrow \nabla f_{\xi_t}(x_t) - \nabla f_{\xi_{t-L}}(x_{t-L})$. To determine which pair to use for updating the inverse Hessian approximation H, we adopt the

doi:10.6342/NTU202504473

high-curvature criterion, selecting the pair that makes the curvature value $y^\top s$ as high as possible. Formally, the procedure is as follows:

1. Choose an anchor point index A.
2. Select the index with the highest curvature relative to the anchor point: $J \leftarrow \arg\max_{j \in t-L,\ldots,t-1} (x_j - x_A)^\top (\nabla f_{\xi_j}(x_j) - \nabla f_{\xi_A}(x_A))$.
3. Construct the curvature pair: $s \leftarrow x_J - x_A, \quad y \leftarrow \nabla f_{\xi_J}(x_J) - \nabla f_{\xi_A}(x_A))$.
4. Apply Eq. (4.10) to check if the curvature pair should be used.

In other words, among the L sampled iterates, we select the one with the largest $y^\top s$ value to construct the curvature pair. In this sense, the update frequency L serves as the sample size for curvature selection. Thus, the proposed method incorporates both the update frequency mechanism and the selection of the highest curvature. Additionally, we apply the safeguard condition in Eq. (4.10) to the selected curvature pair. If the selected curvature pair fails to satisfy the condition in Eq. (4.10), we discard it and proceed to the next L-iteration window.

Another distinction between prior works and our approach lies in the update frequency utilized. Earlier studies typically employ much smaller update frequencies, whereas our method uses significantly larger frequencies to enhance the stability of the optimizer. However, increasing the update frequency introduces delays in updating the inverse Hessian approximation, which may slow the convergence. Gupta et al. [28]; Anil et al. [49] explore this trade-off in the context of Shampoo, a recently recognized second-order optimization method [50; 51]. Their findings demonstrate that update frequencies ranging from approximately 100 to $2,000$ have a minimal impact on the convergence speed, thereby validating our use of larger update frequencies. Moreover, their results suggest that delaying inverse Hessian approximation updates over a substantial interval is acceptable, further motivating the use of cautious updates over Powell's damping.

To further enhance the performance of our method, we incorporate a well-established

doi:10.6342/NTU202504473

acceleration technique, the heavy ball method [24]:

$$x_{t+1} = x_t - \alpha_t \nabla f(x_t) + \beta(x_t - x_{t-1}), \tag{5.1}$$

where $\alpha_t > 0$ denotes the learning rate and $\beta > 0$ is a momentum hyperparameter. This method, later popularized as the momentum method, can be adapted to the stochastic form as follows:

$$x_{t+1} = x_t - \alpha_t \nabla f_{\xi_t}(x_t) + \beta(x_t - x_{t-1}), \tag{5.2}$$

where ξ_t represents a randomly selected sample. This formula is the same as

$$m_t = \beta m_{t-1} + \nabla f_{\xi_t}(x_t) \tag{5.3}$$

$$x_{t+1} = x_t - \alpha_t m_t. \tag{5.4}$$

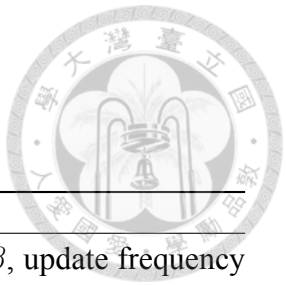
As noted by Gupta et al. [28]; Yao et al. [52]; Xie et al. [53]; Chen et al. [54]; Liu et al. [55], the following formulation is equivalent to the momentum method, and the deep learning community adopts this version:

$$m_t = \beta m_{t-1} + (1 - \beta)\nabla f_{\xi_t}(x_t) \tag{5.5}$$

$$x_{t+1} = x_t - \alpha_t m_t. \tag{5.6}$$

Sutskever et al. [56] emphasize the significance of momentum in deep learning optimization. Although traditional second-order optimization methods typically do not incorporate momentum, recent developments by Gupta et al. [28]; Yao et al. [52]; Liu et al. [55] successfully integrate momentum into their second-order algorithms. Inspired by these advancements, we also explore the incorporation of momentum into our stochastic LBFGS approach to enhance the performance. The proposed stochastic LBFGS method is in Algorithm 4.

doi:10.6342/NTU202504473



Algorithm 4 The proposed stochastic LBFGS method

Input: Initial point x_0, learning rate schedule $\alpha(\cdot)$, momentum β, update frequency L, history size h, threshold ϵ, number of iterations T
$S \leftarrow [\,], \quad Y \leftarrow [\,]$
$k \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  Sample $\xi_{t-1} \sim \mathcal{D}$
  $g_{t-1} \leftarrow \nabla f_{\xi_{t-1}}(x_{t-1})$
  if $t = 1$ **then**
    $J \leftarrow 1$
    Set the anchor: $x_A \leftarrow x_J, \quad g_A \leftarrow g_J$
  end if
  if $t \bmod L = 0$ **then**
    Select the index with the highest curvature: $J \leftarrow \arg\max_{j \in t-L,\ldots,t-1} (x_j - x_A)^\top (g_j - g_A)$
    $s_{k+1} \leftarrow x_J - x_A, \quad y_{k+1} \leftarrow g_J - g_A$
    if $y_{k+1}^\top s_{k+1} > \epsilon s_{k+1}^\top s_{k+1}$ **then**
      $S \leftarrow S \cup \{s_{k+1}\}, \quad Y \leftarrow Y \cup \{y_{k+1}\}$
      $k \leftarrow k + 1$
    end if
    Set the anchor: $x_A \leftarrow x_J, \quad g_A \leftarrow g_J$
  end if
  if $t > 1$ **then**
    $m_{t-1} \leftarrow \beta m_{t-2} + (1 - \beta) g_{t-1}$
  else
    $m_{t-1} \leftarrow g_{t-1}$
  end if
  if $k > 0$ **then**
    $H_{t-1}^0 \leftarrow \frac{y_k^\top s_k}{y_k^\top y_k} I$
    $p_{t-1} \leftarrow -\text{LBFGS\_two\_loop\_recursion}(H_{t-1}^0, S, Y, h, m_{t-1})$
  else
    $p_{t-1} \leftarrow -m_{t-1}$
  end if
  $x_t \leftarrow x_{t-1} + \alpha(t-1) p_{t-1}$
end for
Output: x_T

doi:10.6342/NTU202504473

Chapter 6

Experiments

We compare our stochastic LBFGS method against several baseline optimizers:

1. **SGD with Momentum (SGDM)** [23; 24; 25; 26; 27]: A widely-used stochastic first-order optimizer. The corresponding pseudocode is in Algorithm 5.

2. **Adam** [17]: A popular stochastic first-order optimizer with a diagonal approximation of the Hessian matrix. The pseudocode is in Algorithm 6.

3. **Shampoo** [28; 29]: An stochastic optimizer with a Kronecker product approximation of the Hessian matrix. We adopt the Adam grafting Shampoo[1]. The pseudocode is overly complex. Therefore, we omit it here.

4. **oLBFGS** [19]: A stochastic variant of LBFGS that utilizes data consistency to reduce the noise in stochastic gradients [20]. We also incorporate the cautious update strategy from Eq. (4.10) to address curvature issues in nonconvex optimization [21; 22]. The pseudocode is in Algorithm 7.

We evaluate the optimizers on various image classification tasks, covering both convex and nonconvex problems:

1. Logistic Regression (Lin) [57] on MNIST [26]

2. VGG11[2] [58] on CIFAR10 [59]

[1] https://github.com/facebookresearch/optimizers
[2] https://github.com/weiaicunzai/pytorch-cifar100/blob/master/models/vgg.py

doi:10.6342/NTU202504473

Algorithm 5 SGD with momentum

Input: Initial point x_0, learning rate schedule $\alpha(\cdot)$, momentum β, number of iterations T
for $t = 1, \ldots, T$ **do**
  Sample $\xi_{t-1} \sim \mathcal{D}$
  $g_{t-1} \leftarrow \nabla f_{\xi_{t-1}}(x_{t-1})$
  if $t > 1$ **then**
    $m_{t-1} \leftarrow \beta m_{t-2} + (1-\beta) g_{t-1}$
  else
    $m_{t-1} \leftarrow g_{t-1}$
  end if
  $x_t \leftarrow x_{t-1} - \alpha(t-1) m_{t-1}$
end for
Output: x_T

Algorithm 6 Adam

Input: Initial point x_0, learning rate schedule $\alpha(\cdot)$, exponential moving average parameters β_1, β_2, constant for numerical stability ϵ, number of iterations T
$m_0 \leftarrow 0, \quad v_0 \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  Sample $\xi_{t-1} \sim \mathcal{D}$
  $g_{t-1} \leftarrow \nabla f_{\xi_{t-1}}(x_{t-1})$
  $m_{t-1} \leftarrow \beta_1 m_{t-2} + (1-\beta_1) g_{t-1}$
  $v_{t-1} \leftarrow \beta_2 v_{t-2} + (1-\beta_2) g_{t-1}^2$
  $\hat{m}_{t-1} \leftarrow \frac{1}{1-\beta_1^t}$
  $\hat{v}_{t-1} \leftarrow \frac{1}{1-\beta_2^t}$
  $x_t \leftarrow x_{t-1} - \alpha(t-1) \frac{\hat{m}_{t-1}}{\sqrt{\hat{v}_{t-1}} + \epsilon}$
end for
Output: x_T

3. ResNet18[3] [60] on CIFAR100 [59]

We obtain all datasets via `torchvision` [61] and split them into training and validation sets. However, we use only the training sets for both model training and hyperparameter tuning.

We use the final training objective[4] to evaluate the performance of optimizers, as the goal of optimization is to minimize the function value as much as possible. Therefore, we adopt the final training objective for evaluation. To quantify performance, we define the

[3] `https://github.com/weiaicunzai/pytorch-cifar100/blob/master/models/resnet.py`

[4] Formally, for the VGG11/CIFAR10 and ResNet18/CIFAR100 tasks, we use the stochastic final training objective $f_\xi(x_T)$, where x_T denotes the final model parameters. Due to data augmentation, computing the exact final training objective is computationally prohibitive.

doi:10.6342/NTU202504473

Algorithm 7 oLBFGS

Input: Initial point x_0, learning rate schedule $\alpha(\cdot)$, history size h, threshold ϵ, number of iterations T
$S \leftarrow [\,], \quad Y \leftarrow [\,]$
$k \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  Sample $\xi_{t-1} \sim \mathcal{D}$
  $g_{t-1} \leftarrow \nabla f_{\xi_{t-1}}(x_{t-1})$
  $\tilde{g}_{t-1} \leftarrow \nabla f_{\xi_{t-1}}(x_{t-2})$
  if $t > 1$ **then**
    $s_{k+1} \leftarrow x_{t-1} - x_{t-2}, \quad y_{k+1} \leftarrow g_{t-1} - \tilde{g}_{t-1}$
    if $y_{k+1}^\top s_{k+1} > \epsilon s_{k+1}^\top s_{k+1}$ **or** $f_{\xi_{t-1}}$ is convex **then**
      $S \leftarrow S \cup \{s_{k+1}\}, \quad Y \leftarrow Y \cup \{y_{k+1}\}$
      $k \leftarrow k + 1$
    end if
  end if
  if $k > 0$ **then**
    $H_{t-1}^0 \leftarrow \frac{y_k^\top s_k}{y_k^\top y_k} I$
    $p_{t-1} \leftarrow -\text{LBFGS\_two\_loop\_recursion}(H_{t-1}^0, S, Y, h, g_{t-1})$
  else
    $p_{t-1} \leftarrow -g_{t-1}$
  end if
  $x_t \leftarrow x_{t-1} + \alpha(t-1)p_{t-1}$
end for
Output: x_T

relative performance as the relative difference from the best-performing algorithm:

$$\frac{\text{training objective} - \text{best training objective}}{\text{best training objective}}. \tag{6.1}$$

We train all problems for $100,000$ iterations. We set the batch size to $1,000$ for all optimizers, as the LBFGS method typically requires a relatively large batch size [22]. We employ cosine learning rate scheduling, as defined in Eq. (4.2), for all experiments, and set ϵ in Eq. (4.10) to 0.01 following Bollapragada et al. [22]. Additionally, we perform a grid search to tune the initial learning rate (for all optimizers) and the update frequency (for our stochastic LBFGS method and Shampoo), while keeping all other hyperparameters fixed. We also investigate the effects of different update frequencies and the use of momentum to identify the optimal strategy for our stochastic LBFGS method.

Our first experiment is to examine the effect of varying update frequencies on the per-

doi:10.6342/NTU202504473

formance of our stochastic LBFGS method, both with momentum ($\beta = 0.9$) and without momentum ($\beta = 0.0$). As shown in Table 6.1 and Table 6.2, varying the update frequency within the range of $1,000$ to $10,000$ iterations has no significant impact on the performance.

Next, we evaluate the impact of incorporating momentum into our stochastic LBFGS method. Because classical LBFGS methods do not employ momentum, it is essential to check the effect after incorporating this technique. As shown in Table 6.3, LBFGS with momentum achieves performance comparable to, and in some cases better than, its non-momentum counterpart. Based on this result, we adopt the momentum variant in subsequent experiments, as the setting aligns with modern practices in deep learning optimization.

Note that Shampoo introduces a similar update frequency to mitigate the computational cost of matrix inversion. We investigate whether Shampoo's performance degrades as the update frequency increases from 100 to $1,000$. As shown in Table 6.4, Shampoo maintains stable performance across different update frequencies.

Recall that oLBFGS incorporates the data consistency scheme to address the issue of data inconsistency. To assess its effectiveness, we compare oLBFGS with the cautious stochastic LBFGS (cLBFGS) in Algorithm 3, which differs from oLBFGS in Algorithm 7 only in the construction of y:

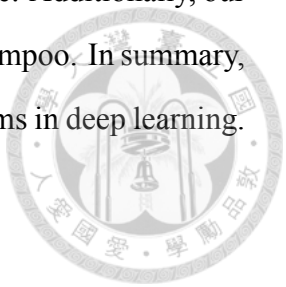
$$y \leftarrow \nabla f_{\xi_{t-1}}(x_{t-1}) - \nabla f_{\xi_{t-1}}(x_{t-2}). \tag{6.2}$$

As shown in Table 6.5, the stochastic LBFGS method without data consistency (cLBFGS) performs significantly worse, highlighting the critical role of data consistency in achieving stable and effective optimization.

Finally, we perform a comprehensive comparison between our proposed stochastic LBFGS method and baseline optimizers, as presented in Table 6.6 and Fig. 6.1. We detail the hyperparameter settings used in Table 6.6 in Appendix B. The results demonstrate that our method significantly outperforms oLBFGS. Therefore, even without imposing the scheme in oLBFGS to address the data inconsistency, our strategy of utilizing update

doi:10.6342/NTU202504473

frequency to select the pair with the highest curvature is highly effective. Additionally, our stochastic LBFGS method is competitive with SGDM, Adam, and Shampoo. In summary, we have made LBFGS a viable option for solving optimization problems in deep learning.



doi:10.6342/NTU202504473

Table 6.1: Effect of the update frequency in our stochastic LBFGS method with momentum.

Update Frequency	Training Objective	Relative Performance
	Lin/MNIST	
10000	2.05e-01	0.51%
5000	2.05e-01	0.47%
2500	2.04e-01	0.22%
1000	2.04e-01	0.05%
500	2.04e-01	0.00%
250	2.05e-01	0.73%
100	2.08e-01	1.81%
50	2.13e-01	4.70%
25	2.20e-01	7.88%
10	2.28e-01	12.08%
	VGG11/CIFAR10	
10000	5.03e-06	348.30%
5000	1.86e-06	65.59%
2500	1.12e-06	0.00%
1000	1.76e-05	1464.76%
500	1.19e-04	10508.34%
250	5.15e-04	45811.69%
100	4.00e-03	356565.91%
50	1.03e-02	914342.01%
25	2.22e-01	19778894.21%
10	7.50e-01	66905531.44%
	ResNet18/CIFAR100	
10000	2.59e-04	5.44%
5000	2.62e-04	6.49%
2500	2.67e-04	8.47%
1000	2.68e-04	9.13%
500	2.46e-04	0.00%
250	2.72e-04	10.78%
100	3.73e-04	51.63%
50	1.10e-03	348.71%
25	6.90e-03	2704.45%
10	2.74e-01	111136.59%

doi:10.6342/NTU202504473

Table 6.2: Effect of the update frequency in our stochastic LBFGS method without momentum.

Update Frequency	Training Objective	Relative Performance
	Lin/MNIST	
10000	2.08e-01	1.62%
5000	2.06e-01	0.79%
2500	2.06e-01	0.64%
1000	2.05e-01	0.04%
500	2.05e-01	0.27%
250	2.05e-01	0.00%
100	2.08e-01	1.38%
50	2.15e-01	5.17%
25	2.26e-01	10.27%
10	2.45e-01	19.82%
	VGG11/CIFAR10	
10000	5.76e-07	0.00%
5000	2.03e-06	252.86%
2500	3.99e-05	6815.23%
1000	2.44e-05	4129.33%
500	1.03e-04	17834.69%
250	4.00e-04	69232.21%
100	1.41e-03	244398.03%
50	1.38e-02	2386403.99%
25	2.62e-02	4549734.50%
10	2.80e-01	48530316.24%
	ResNet18/CIFAR100	
10000	2.67e-04	0.00%
5000	2.68e-04	0.21%
2500	2.92e-04	9.30%
1000	3.17e-04	18.65%
500	4.67e-04	74.72%
250	4.97e-04	86.17%
100	9.63e-04	260.43%
50	3.30e-03	1135.33%
25	3.46e-03	1194.70%
10	3.02e-03	1030.63%

doi:10.6342/NTU202504473

Table 6.3: Comparison of our stochastic LBFGS method with and without momentum across different problems.

Method	Training Objective	Relative Performance
	Lin/MNIST	
with momentum	2.04e-01	0.00%
w/o momentum	2.05e-01	0.42%
	VGG11/CIFAR10	
with momentum	1.12e-06	94.58%
w/o momentum	5.76e-07	0.00%
	ResNet18/CIFAR100	
with momentum	2.46e-04	0.00%
w/o momentum	2.67e-04	8.60%

Table 6.4: Effect of update frequency on the training objective of Shampoo.

Update Frequency	Training Objective	Relative Performance
	Lin/MNIST	
1000	2.00e-01	0.00%
500	2.00e-01	0.00%
250	2.00e-01	0.01%
100	2.00e-01	0.02%
	VGG11/CIFAR10	
1000	1.74e-07	10.69%
500	2.32e-07	48.05%
250	1.75e-07	11.68%
100	1.57e-07	0.00%
	ResNet18/CIFAR100	
1000	2.42e-04	0.06%
500	2.45e-04	1.05%
250	2.42e-04	0.00%
100	2.42e-04	0.16%

doi:10.6342/NTU202504473

Table 6.5: Comparison of cLBFGS and oLBFGS.

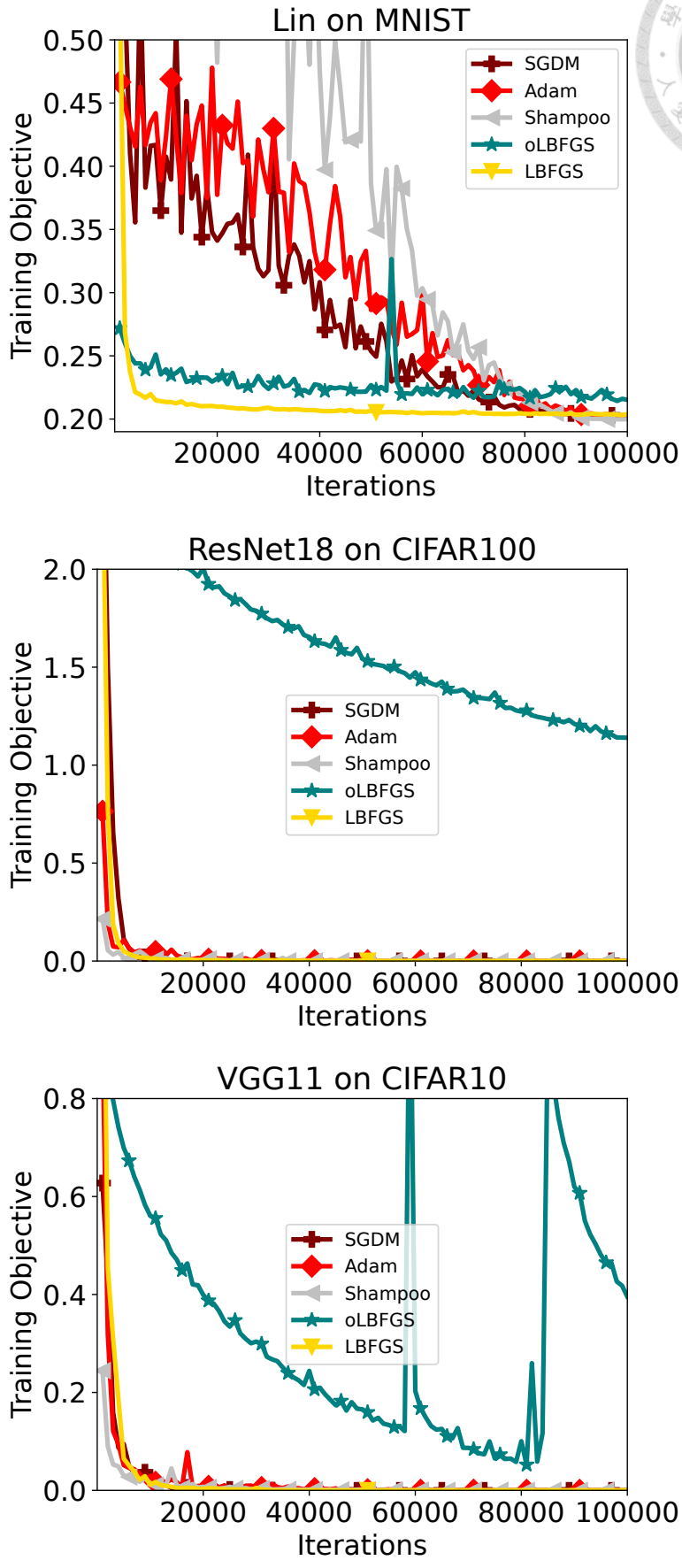
Method	Training Objective	Relative Performance
	Lin/MNIST	
cLBFGS	5.53e-01	156.91%
oLBFGS	2.15e-01	0.00%
	VGG11/CIFAR10	
cLBFGS	2.01e+00	408.76%
oLBFGS	3.96e-01	0.00%
	ResNet18/CIFAR100	
cLBFGS	2.07e+02	18081.66%
oLBFGS	1.14e+00	0.00%

Table 6.6: Comparison of optimizers across different tasks.

Method	Training objective	Relative Performance
	Lin/MNIST	
SGDM	2.03e-01	1.52%
Adam	2.02e-01	1.01%
Shampoo	2.00e-01	0.00%
oLBFGS	2.15e-01	7.68%
Ours	2.04e-01	2.01%
	VGG11/CIFAR10	
SGDM	4.82e-07	16434.35%
Adam	2.92e-09	0.00%
Shampoo	1.57e-07	5282.18%
oLBFGS	3.96e-01	13567789744.01%
Ours	1.12e-06	38365.49%
	ResNet18/CIFAR100	
SGDM	2.69e-04	11.13%
Adam	2.46e-04	1.60%
Shampoo	2.42e-04	0.00%
oLBFGS	1.14e+00	470767.86%
Ours	2.46e-04	1.61%

doi:10.6342/NTU202504473

Figure 6.1: Comparison of training curves for optimizers across different tasks, showing training objectives versus iterations.



doi:10.6342/NTU202504473

Bibliography

[1] Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, 1999.

[2] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(61):1871–1874, 2008.

[3] John E Dennis, Jr and Jorge J Moré. Quasi-newton methods, motivation and theory. *SIAM Review*, 19(1):46–89, 1977.

[4] Charles G Broyden. A new double-rank minimization algorithm. *Notices of the American Mathematical Society*, 16(4):670, 1969.

[5] Roger Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 1970.

[6] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

[7] David F Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656, 1970.

[8] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

doi:10.6342/NTU202504473

[10] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y Ng. Deep speech: Scaling up end-to-end speech recognition. Technical report, 2014.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference for Learning Representations*, 2015.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[13] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. Technical report, 2024.

[14] OpenAI. Gpt-4 technical report. Technical report, 2023.

[15] Gemini Team Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. Technical report, 2024.

[16] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):2121–2159, 2011.

[17] P Kingma Diederik and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference for Learning Representations*, 2019.

[19] Nicol N Schraudolph, Jin Yu, and Simon Günter. A stochastic quasi-newton method for online convex optimization. In *Artificial Intelligence and Statistics*, 2007.

doi:10.6342/NTU202504473

[20] Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory bfgs. *Journal of Machine Learning Research*, 16(1):3151–3181, 2015.

[21] Albert S Berahas, Jorge Nocedal, and Martin Takác. A multi-batch l-bfgs method for machine learning. In *Advances in Neural Information Processing Systems*, 2016.

[22] Raghu Bollapragada, Jorge Nocedal, Dheevatsa Mudigere, Hao-Jun Shi, and Ping Tak Peter Tang. A progressive batching l-bfgs method for machine learning. In *International Conference on Machine Learning*, 2018.

[23] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[24] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[25] Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, 2007.

[28] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, 2018.

[29] Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel pytorch implementation of the distributed shampoo optimizer for training neural networks at-scale. Technical report, 2023.

[30] John E Dennis, Jr and Robert B Schnabel. Least change secant updates for quasi-newton methods. *Siam Review*, 21(4):443–459, 1979.

doi:10.6342/NTU202504473

[31] Betty Shea and Mark Schmidt. Don't be so positive: Negative step sizes in second-order methods. Technical report, 2024.

[32] Jie Gui, Tuo Chen, Jing Zhang, Qiong Cao, Zhenan Sun, Hao Luo, and Dacheng Tao. A survey on self-supervised learning: Algorithms, applications, and future trends. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12): 9052–9071, 2024.

[33] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[34] Aaron Defazio and Léon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. In *Advances in Neural Information Processing Systems*, 2019.

[35] Maren Mahsereci and Philipp Hennig. Probabilistic line searches for stochastic optimization. *Journal of Machine Learning Research*, 18(119):1–59, 2017.

[36] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *Advances in Neural Information Processing Systems*, 2019.

[37] Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[38] Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic l-bfgs algorithm. In *Artificial Intelligence and Statistics*, 2016.

[39] Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27 (2):927–956, 2017.

[40] Yue Niu, Zalan Fabian, Sunwoo Lee, Mahdi Soltanolkotabi, and Salman Avestimehr.

doi:10.6342/NTU202504473

ml-BFGS: A momentum-based l-BFGS for distributed large-scale neural network optimization. *Transactions on Machine Learning Research*, 2023.

[41] Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-newton methods for training deep neural networks. In *Advances in Neural Information Processing Systems*, 2020.

[42] Aryan Mokhtari and Alejandro Ribeiro. Res: Regularized stochastic bfgs algorithm. *IEEE Transactions on Signal Processing*, 62(23):6089–6104, 2014.

[43] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference for Learning Representations*, 2017.

[44] Michael JD Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 14:224–248, 1978.

[45] Dong-Hui Li and Masao Fukushima. On the global convergence of the bfgs method for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 11(4):1054–1064, 2001.

[46] Antoine Bordes, Léon Bottou, and Patrick Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10(59):1737–1754, 2009.

[47] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, 2013.

[48] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. How to compute hessian-vector products? Technical report, 2024.

[49] Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. Technical report, 2020.

doi:10.6342/NTU202504473

[50] Wu Lin, Felix Dangel, Runa Eschenhagen, Juhan Bae, Richard E Turner, and Alireza Makhzani. Can we remove the square-root in adaptive gradient methods? a second-order perspective. In *International Conference on Machine Learning*, 2024.

[51] Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham Kakade, and Lucas Janson. A new perspective on shampoo's preconditioner. In *International Conference on Learning Representations*, 2025.

[52] Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[53] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9508–9520, 2024.

[54] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems*, 2023.

[55] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International Conference on Learning Representations*, 2024.

[56] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, 2013.

[57] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[58] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference for Learning Representations*, 2015.

doi:10.6342/NTU202504473

[59] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

[61] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

[62] Nicholas J Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.

[63] Richard H Byrd, Jorge Nocedal, and Robert B Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1):129–156, 1994.

[64] Jennifer B Erway, Vibhor Jain, and Roummel F Marcia. Shifted limited-memory dfp systems. In *Asilomar Conference on Signals, Systems and Computers*, 2013.

doi:10.6342/NTU202504473

Appendix

A Proof of Rank-Two Update Form for BFGS and DFP

To demonstrate that the DFP and BFGS methods perform rank-two updates, we first introduce the Sherman–Morrison–Woodbury formula [62]:

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1}+VA^{-1}U\right)^{-1}VA^{-1}. \tag{A.1}$$

Based on the compact representation of the BFGS method [63], the update rule for the inverse Hessian approximation H_t^{BFGS} is:

$$H_t^{\text{BFGS}} = \left(I - \frac{s_t y_t^\top}{y_t^\top s_t}\right) H_{t-1}^{\text{BFGS}} \left(I - \frac{y_t s_t^\top}{y_t^\top s_t}\right) + \frac{s_t s_t^\top}{y_t^\top s_t} \tag{A.2}$$

$$= H_{t-1}^{\text{BFGS}} + \begin{bmatrix} s_t & H_{t-1}^{\text{BFGS}} y_t \end{bmatrix} \begin{bmatrix} \frac{s_t^\top y_t + y_t^\top H_{t-1}^{\text{BFGS}} y_t}{y_t^\top s_t s_t^\top y_t} & -\frac{1}{y_t^\top s_t} \\ -\frac{1}{s_t^\top y_t} & 0 \end{bmatrix} \begin{bmatrix} s_t^\top \\ y_t^\top H_{t-1}^{\text{BFGS}} \end{bmatrix}. \tag{A.3}$$

By applying the Sherman–Morrison–Woodbury formula to Eq. (A.3), we derive the

doi:10.6342/NTU202504473

following:

$$
\begin{aligned}
B_t^{\text{BFGS}} &= H_{t-1}^{\text{BFGS}^{-1}} - H_{t-1}^{\text{BFGS}^{-1}} \begin{bmatrix} s_t & H_{t-1}^{\text{BFGS}} y_t \end{bmatrix} \\
&\left(\begin{bmatrix} \frac{s_t^\top y_t + y_t^\top H_{t-1}^{\text{BFGS}} y_t}{y_t^\top s_t s_t^\top y_t} & -\frac{1}{y_t^\top s_t} \\ -\frac{1}{s_t^\top y_t} & 0 \end{bmatrix}^{-1} + \begin{bmatrix} s_t^\top \\ y_t^\top H_{t-1}^{\text{BFGS}} \end{bmatrix} H_{t-1}^{\text{BFGS}^{-1}} \begin{bmatrix} s_t & H_{t-1}^{\text{BFGS}} y_t \end{bmatrix} \right)^{-1} \\
&\begin{bmatrix} s_t^\top \\ y_t^\top H_{t-1}^{\text{BFGS}} \end{bmatrix} H_{t-1}^{\text{BFGS}^{-1}} \\
&= B_{t-1}^{\text{BFGS}} - \begin{bmatrix} B_{t-1}^{\text{BFGS}} s_t & y_t \end{bmatrix} \\
&\left(\begin{bmatrix} \frac{s_t^\top y_t + y_t^\top H_{t-1}^{\text{BFGS}} y_t}{y_t^\top s_t s_t^\top y_t} & -\frac{1}{y_t^\top s_t} \\ -\frac{1}{s_t^\top y_t} & 0 \end{bmatrix}^{-1} + \begin{bmatrix} s_t^\top B_{t-1}^{\text{BFGS}} s_t & s_t^\top y_t \\ y_t^\top s_t & y_t^\top H_{t-1}^{\text{BFGS}} y_t \end{bmatrix} \right)^{-1} \begin{bmatrix} s_t^\top B_{t-1}^{\text{BFGS}} \\ y_t^\top \end{bmatrix} \\
&= B_{t-1}^{\text{BFGS}} - \begin{bmatrix} B_{t-1}^{\text{BFGS}} s_t & y_t \end{bmatrix} \\
&\left(\begin{bmatrix} 0 & -s_t^\top y_t \\ -y_t^\top s_t & -s_t^\top y_t - y_t^\top H_{t-1}^{\text{BFGS}} y_t \end{bmatrix} + \begin{bmatrix} s_t^\top B_{t-1}^{\text{BFGS}} s_t & s_t^\top y_t \\ y_t^\top s_t & y_t^\top H_{t-1}^{\text{BFGS}} y_t \end{bmatrix} \right)^{-1} \begin{bmatrix} s_t^\top B_{t-1}^{\text{BFGS}} \\ y_t^\top \end{bmatrix} \\
&= B_{t-1}^{\text{BFGS}} - \begin{bmatrix} B_{t-1}^{\text{BFGS}} s_t & y_t \end{bmatrix} \begin{bmatrix} s_t^\top B_{t-1}^{\text{BFGS}} s_t & 0 \\ 0 & -s_t^\top y_t \end{bmatrix}^{-1} \begin{bmatrix} s_t^\top B_{t-1}^{\text{BFGS}} \\ y_t^\top \end{bmatrix} \\
&= B_{t-1}^{\text{BFGS}} - \begin{bmatrix} B_{t-1}^{\text{BFGS}} s_t & y_t \end{bmatrix} \begin{bmatrix} \frac{1}{s_t^\top B_{t-1}^{\text{BFGS}} s_t} & 0 \\ 0 & -\frac{1}{s_t^\top y_t} \end{bmatrix} \begin{bmatrix} s_t^\top B_{t-1}^{\text{BFGS}} \\ y_t^\top \end{bmatrix} \\
&= B_{t-1}^{\text{BFGS}} - \underbrace{\frac{B_{t-1}^{\text{BFGS}} s_t s_t^\top B_{t-1}^{\text{BFGS}}}{s_t^\top B_{t-1}^{\text{BFGS}} s_t}}_{\text{rank-1}} + \underbrace{\frac{y_t y_t^\top}{s_t^\top y_t}}_{\text{rank-1}}.
\end{aligned}
\tag{A.4}
$$

Therefore, the BFGS method applies a rank-two update to the Hessian approximation B_{t-1}^{BFGS}.

Similarly, based on the compact representation of the DFP method [64], the update

doi:10.6342/NTU202504473

rule for the Hessian approximation B_t^{DFP} is:

$$B_t^{\text{DFP}} = \left(I - \frac{y_t s_t^\top}{y_t^\top s_t}\right) B_{t-1}^{\text{DFP}} \left(I - \frac{s_t y_t^\top}{y_t^\top s_t}\right) + \frac{y_t y_t^\top}{y_t^\top s_t} \tag{A.5}$$

$$= B_{t-1}^{\text{DFP}} + \begin{bmatrix} y_t & B_{t-1}^{\text{DFP}} s_t \end{bmatrix} \begin{bmatrix} \frac{y_t^\top s_t + s_t^\top B_{t-1}^{\text{DFP}} s_t}{s_t^\top y_t y_t^\top s_t} & -\frac{1}{s_t^\top y_t} \\ -\frac{1}{y_t^\top s_t} & 0 \end{bmatrix} \begin{bmatrix} y_t^\top \\ s_t^\top B_{t-1}^{\text{DFP}} \end{bmatrix}. \tag{A.6}$$

Applying the Sherman–Morrison–Woodbury formula to Eq. (A.6) yields:

$$\begin{aligned}
H_t^{\text{DFP}} &= B_{t-1}^{\text{DFP}^{-1}} - B_{t-1}^{\text{DFP}^{-1}} \begin{bmatrix} y_t & B_{t-1}^{\text{DFP}} s_t \end{bmatrix} \\
&\left(\begin{bmatrix} \frac{y_t^\top s_t + s_t^\top B_{t-1}^{\text{DFP}} s_t}{s_t^\top y_t y_t^\top s_t} & -\frac{1}{s_t^\top y_t} \\ -\frac{1}{y_t^\top s_t} & 0 \end{bmatrix}^{-1} + \begin{bmatrix} y_t^\top \\ s_t^\top B_{t-1}^{\text{DFP}} \end{bmatrix} B_{t-1}^{\text{DFP}^{-1}} \begin{bmatrix} y_t & B_{t-1}^{\text{DFP}} s_t \end{bmatrix} \right)^{-1} \\
&\begin{bmatrix} y_t^\top \\ s_t^\top B_{t-1}^{\text{DFP}} \end{bmatrix} B_{t-1}^{\text{DFP}^{-1}} \\
&= H_{t-1}^{\text{DFP}} - \begin{bmatrix} H_{t-1}^{\text{DFP}} y_t & s_t \end{bmatrix} \\
&\left(\begin{bmatrix} \frac{y_t^\top s_t + s_t^\top B_{t-1}^{\text{DFP}} s_t}{s_t^\top y_t y_t^\top s_t} & -\frac{1}{s_t^\top y_t} \\ -\frac{1}{y_t^\top s_t} & 0 \end{bmatrix}^{-1} + \begin{bmatrix} y_t^\top H_{t-1}^{\text{DFP}} y_t & y_t^\top s_t \\ s_t^\top y_t & s_t^\top B_{t-1}^{\text{DFP}} s_t \end{bmatrix} \right)^{-1} \begin{bmatrix} y_t^\top H_{t-1}^{\text{DFP}} \\ s_t^\top \end{bmatrix} \\
&= H_{t-1}^{\text{DFP}} - \begin{bmatrix} H_{t-1}^{\text{DFP}} y_t & s_t \end{bmatrix} \\
&\left(\begin{bmatrix} 0 & -y_t^\top s_t \\ -s_t^\top y_t & -y_t^\top s_t - s_t^\top B_{t-1}^{\text{DFP}} s_t^\top \end{bmatrix} + \begin{bmatrix} y_t^\top H_{t-1}^{\text{DFP}} y_t & y_t^\top s_t \\ s_t^\top y_t & s_t^\top B_{t-1}^{\text{DFP}} s_t \end{bmatrix} \right)^{-1} \begin{bmatrix} y_t^\top H_{t-1}^{\text{DFP}} \\ s_t^\top \end{bmatrix} \\
&= H_{t-1}^{\text{DFP}} - \begin{bmatrix} H_{t-1}^{\text{DFP}} y_t & s_t \end{bmatrix} \begin{bmatrix} y_t^\top H_{t-1}^{\text{DFP}} y_t & 0 \\ 0 & -y_t^\top s_t \end{bmatrix}^{-1} \begin{bmatrix} y_t^\top H_{t-1}^{\text{DFP}} \\ s_t^\top \end{bmatrix} \\
&= H_{t-1}^{\text{DFP}} - \begin{bmatrix} H_{t-1}^{\text{DFP}} y_t & s_t \end{bmatrix} \begin{bmatrix} \frac{1}{y_t^\top H_{t-1}^{\text{DFP}} y_t} & 0 \\ 0 & -\frac{1}{y_t^\top s_t} \end{bmatrix} \begin{bmatrix} y_t^\top H_{t-1}^{\text{DFP}} \\ s_t^\top \end{bmatrix} \\
&= H_{t-1}^{\text{DFP}} - \underbrace{\frac{H_{t-1}^{\text{DFP}} y_t y_t^\top H_{t-1}^{\text{DFP}}}{y_t^\top H_{t-1}^{\text{DFP}} y_t}}_{\text{rank-1}} + \underbrace{\frac{s_t s_t^\top}{y_t^\top s_t}}_{\text{rank-1}}.
\end{aligned} \tag{A.7}$$

Hence, the DFP method also performs a rank-two update on the inverse Hessian approxi-

doi:10.6342/NTU202504473

mation H_{t-1}^{DFP}.

B Hyperparameter Settings

We present the hyperparameter settings of Table 6.6 in Table 7, Table 8, and Table 9.

doi:10.6342/NTU202504473

Table 7: Hyperparameter settings for Lin on MNIST

Lin/MNIST	
SGDM	
batch size	1000
number of iterations	100000
learning rate	5.0
beta	0.9
weight decay	0.0
Adam	
batch size	1000
number of iterations	100000
learning rate	0.05
betas	(0.9, 0.999)
epsilon	1×10^{-8}
weight decay	0.0
Shampoo	
batch size	1000
number of iterations	100000
learning rate	0.25
betas	(0.9, 0.999)
beta3	0.9
epsilon	1×10^{-12}
max preconditioner dim	8192
grafting config	Adam with epsilon=1×10^{-8} and beta2=0.999
update frequency	500
weight decay	0.0
oLBFGS	
batch size	1000
number of iterations	100000
learning rate	1.0
history size	10
epsilon	0.01
weight decay	0.0
Ours	
batch size	1000
number of iterations	100000
learning rate	1.0
beta	0.9
history size	5
update frequency	500
epsilon	0.01
weight decay	0.0

doi:10.6342/NTU202504473

Table 8: Hyperparameter settings for VGG11 on CIFAR10

VGG11/CIFAR10	
SGDM	
batch size	1000
number of iterations	100000
learning rate	0.25
beta	0.9
weight decay	0.0
Adam	
batch size	1000
number of iterations	100000
learning rate	0.0025
betas	(0.9, 0.999)
epsilon	1×10^{-8}
weight decay	0.0
Shampoo	
batch size	1000
number of iterations	100000
learning rate	0.001
betas	(0.9, 0.999)
beta3	0.9
epsilon	1×10^{-12}
max preconditioner dim	8192
grafting config	Adam with epsilon=1×10^{-8} and beta2=0.999
update frequency	100
weight decay	0.0
oLBFGS	
batch size	1000
number of iterations	100000
learning rate	1.0
history size	10
epsilon	0.01
weight decay	0.0
Ours	
batch size	1000
number of iterations	100000
learning rate	0.5
beta	0.9
history size	5
update frequency	2500
epsilon	0.01
weight decay	0.0

doi:10.6342/NTU202504473

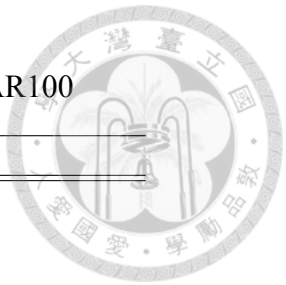


Table 9: Hyperparameter settings for ResNet18 on CIFAR100

ResNet18/CIFAR100	
SGDM	
batch size	1000
number of iterations	100000
learning rate	5.0
beta	0.9
weight decay	0.0
Adam	
batch size	1000
number of iterations	100000
learning rate	0.001
betas	(0.9, 0.999)
epsilon	1×10^{-8}
weight decay	0.0
Shampoo	
batch size	1000
number of iterations	100000
learning rate	0.001
betas	(0.9, 0.999)
beta3	0.9
epsilon	1×10^{-12}
max preconditioner dim	8192
grafting config	Adam with epsilon=1×10^{-8} and beta2=0.999
update frequency	250
weight decay	0.0
oLBFGS	
batch size	1000
number of iterations	100000
learning rate	1.0
history size	10
epsilon	0.01
weight decay	0.0
Ours	
batch size	1000
number of iterations	100000
learning rate	2.5
beta	0.9
history size	5
update frequency	500
epsilon	0.01
weight decay	0.0

doi:10.6342/NTU202504473