SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Quasi-Newton Methods for Neural Network Training in Machine Learning

av

Stefan Miletic

2021 - No K37

Quasi-Newton Methods for Neural Network Training in Machine Learning

Stefan Miletic

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Yishao Zhou

2021

Abstract

The theory of mathematical optimization provides a powerful tool in modern sciences as computational capabilities allow for fast and reliable solutions. One of those application areas is *deep learning* where different algorithms are *trained* to solve various problems within the field of *machine learning*. Typically a loss or error function is defined which needs to be minimized. For this reason, a brief introduction to optimization theory is given with focus on iterative methods incorporating *line search* and *trust region* techniques. A lot of attention will be paid to *Newton's method* and we will see how it can be further improved to handle large-scale problems which usually arise as a consequence of *network training*. In order to better understand the nature of these problems and why Newton-based iterative methods handle them well, a short introduction to *neural networks* and *network training* will be mathematically presented, where attention is paid to a special type of networks called *feed-forward* networks. Lastly, popular *quasi-Newton methods* will be introduced and explored in great detail with derivation and convergence analysis in focus. A large portion of our attention will be dedicated to some recent improvements towards *quasi-Newton methods* where *deep neural network training* performance is of importance.

Contents

1 Introduction

The general idea of *unconstrained optimization* concerns the minimization problem of a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ of an n-dimensional input variable $\boldsymbol{x}$. We call f an objective function and formulate this problem as

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}).$$

One seeks to find a *local minimizer*, which we will denote by $\boldsymbol{x}^*$, that satisfies

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \text{ for all } \boldsymbol{x} \text{ in some neighborhood of } \boldsymbol{x}^*.$$

If that neighborhood is the whole domain of f, then $\boldsymbol{x}^*$ is also said to be a *global minimizer*. This can be further extended to more delicate situations where additional constraints may be introduced. For example, we may be interested in minimization of $f(\boldsymbol{x})$ over a subset $M \subseteq \mathbb{R}^n$ for which some vector-valued function $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x}))^\top$ satisfies certain conditions on its component functions. The set M is called the *feasible set*, and this gives rise to *constrained optimization* problems

$$\min_{\boldsymbol{x} \in M} f(\boldsymbol{x}) \quad \text{subject to } \begin{cases} h_i(\boldsymbol{x}) = 0, & i \in I \\ h_j(\boldsymbol{x}) \leq 0, & j \in J, \end{cases}$$

where I and J are sets of indices such that $I \cap J = \emptyset$ and $I \cup J = \{1, \ldots, m\}$.

The notion of *convexity* as a property of an objective function together with the feasible set plays a fundamental role in optimization theory. We will define convexity in the next section and derive one important conclusion from it. Of particular interest for us will be optimization of nonlinear functions. These functions appear in a lot of different application areas, *inter alia*, the study of neural networks in deep learning where objective functions are often highly nonlinear [2, 3].

The main disposition of this paper is split into two parts, the preliminaries necessarry to understand the material (*Chapter 2*) and the main topic involving *symmetric rank-one* (SR1) and *Broyden–Fletcher–Goldfarb–Shanno* (BFGS) quasi-Newton methods (*Chapter 3*). In the preliminary part, we will introduce some general concepts within the multivariate analysis and optimization theory. We will then use those concepts to develop some standard techniques which are fundamental in the derivation of quasi-Newton methods. The covered techniques are *line search* and *trust region* methods which together fall under one broader idea called *iterative* methods. We will also derive Newton's method and see how it can be further developed into quasi-Newton methods. Since the goal of this paper is to present quasi-Newton methods' application in neural network training, we will also mathematically present neural networks and show how they are *trained*. In the main part (*Chapter 3*), we will derive standard quasi-Newton methods (SR1 and BFGS) and then extend them to varieties which are well suited for large-scale problems which naturally arise as a consequence of neural network training (*limited-memory* SR1 and BFGS). These varieties will

also be further extended to some recent studies which proved to be even more efficient and robust when it comes to *deep* neural network training. Lastly, we will show some numerical experiments in (*Chapter 4*) where a small network is trained on a personal computer using BFGS-based methods.

2 Preliminaries

2.1 General Concepts

Throughout this paper we will be dealing with a lot of terminology and it will be useful to introduce some of the key concepts.

We begin by defining convexity for sets and functions, then show one of their immediate consequences on local minimizers of convex functions.

Definition 2.1. *Let $S \subset \mathbb{R}^n$. We say that S is convex if for any two points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in S$ the line segment connecting $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ lies entirely in S. That is, S is convex if and only if for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in S$ and $t \in [0,1]$,*

$$t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2 \in S.$$

Definition 2.2. *Let $S \subset \mathbb{R}^n$ and $f : S \to \mathbb{R}$. We say that f is convex if and only if S is convex and for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in S$ and $t \in [0,1]$*

$$f(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2) \leq tf(\boldsymbol{x}_1) + (1-t)f(\boldsymbol{x}_2).$$

Theorem 2.1. *If $f : \mathbb{R}^n \to \mathbb{R}$ is convex and $\boldsymbol{x}^*$ is a local minimizer of f, then $\boldsymbol{x}^*$ is also a global minimizer of f.*

Proof. Suppose that $\boldsymbol{x}^*$ is not a global minimizer. Then there exists some $\boldsymbol{x}_0$ such that $f(\boldsymbol{x}_0) < f(\boldsymbol{x}^*)$. By convexity of f, we get

$$\begin{aligned} f(t\boldsymbol{x}_0 + (1-t)\boldsymbol{x}^*) &\leq tf(\boldsymbol{x}_0) + (1-t)f(\boldsymbol{x}^*) \\ &< tf(\boldsymbol{x}^*) + (1-t)f(\boldsymbol{x}^*) = f(\boldsymbol{x}^*), \end{aligned}$$

where $t \in (0,1]$. Since the argument of the left-hand side, $t\boldsymbol{x}_0 + (1-t)\boldsymbol{x}^*$, corresponds to a line segment connecting $\boldsymbol{x}_0$ and $\boldsymbol{x}^*$, it follows that any neighborhood of $\boldsymbol{x}^*$ will contain a subset of that line segment, hence $f(\boldsymbol{x}^*)$ cannot be a local minimizer. It follows that $\boldsymbol{x}^*$ is a global minimizer.

□

This is a very important result that can potentially save a lot of computational time. Suppose we are interested in minimization of an objective function that is not convex, where a local minimum is obtained through an iterative method. Since the objective function is not convex, there is no guarantee that the minimum will be global, hence multiple solutions need be compared, typically by running the iteration several times with different initial values.

Next, we define the gradient and the Hessian of a function, positive definiteness and semi-definiteness of a matrix and present Taylor's theorem.

Definition 2.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function of several variables $\boldsymbol{x} = (x_1, \ldots, x_n)$. Then the gradient of f is defined as*

$$\nabla f \coloneqq \nabla f(\boldsymbol{x}) \coloneqq \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^\top.$$

Definition 2.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function of several variables $\boldsymbol{x} = (x_1, \ldots, x_n)$. Then the Hessian matrix, or Hessian, of f is defined as*

$$\boldsymbol{H}f \coloneqq \nabla^2 f \coloneqq (\nabla^2 f(\boldsymbol{x}))_{ij} \coloneqq \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

Definition 2.5. *Let $\boldsymbol{M}$ be a symmetric $n \times n$ real matrix and $\boldsymbol{x} \in \mathbb{R}^n$ a real vector. We say that $\boldsymbol{M}$ is:*

positive definite if for all nonzero $\boldsymbol{x}$	$\boldsymbol{x}^\top \boldsymbol{M} \boldsymbol{x} > 0,$
positive semi-definite if for all $\boldsymbol{x}$	$\boldsymbol{x}^\top \boldsymbol{M} \boldsymbol{x} \geq 0.$

Theorem 2.2 (Taylor's theorem)**.** *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and let $\boldsymbol{p} \in \mathbb{R}^n$. Then for some $t \in (0, 1)$ it follows that*

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \boldsymbol{p}^\top \nabla f(\boldsymbol{x} + t\boldsymbol{p}).$$

Furthermore, if f is twice continuously differentiable, it follows that

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \boldsymbol{p}^\top \nabla f(\boldsymbol{x}) + \frac{1}{2} \boldsymbol{p}^\top \nabla^2 f(\boldsymbol{x} + t\boldsymbol{p}) \boldsymbol{p}$$

and

$$\nabla f(\boldsymbol{x} + \boldsymbol{p}) = \nabla f(\boldsymbol{x}) + \int_0^1 \nabla^2 f(\boldsymbol{x} + t\boldsymbol{p}) \boldsymbol{p},$$

for some $t \in (0, 1)$.

Proof. (See [8].) □

We are now ready to state the necessary and sufficient conditions for optimality which will be crucial guidelines to the most of iterative methods we will discus later.

Theorem 2.3 (First-order necessary condition)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function of several variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{x}^*$ a local minimizer of f, then*

$$\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}.$$

Proof. Since $\boldsymbol{x}^*$ is a local minimizer of f, we have for all $\boldsymbol{p} \in \mathbb{R}^n$ and τ sufficiently small

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}^* + \tau \boldsymbol{p}).$$

It follows from Taylor's theorem that

$$f(\boldsymbol{x}^* + \tau \boldsymbol{p}) = f(\boldsymbol{x}^*) + \tau \boldsymbol{p}^\top \nabla f(\boldsymbol{x}^* + t\boldsymbol{p}),$$

for some $t \in (0, \tau)$, hence

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}^*) + \tau \boldsymbol{p}^\top \nabla f(\boldsymbol{x}^* + t\boldsymbol{p}).$$

It follows for $\tau > 0$ that

$$\begin{aligned} 0 &\leq \boldsymbol{p}^\top \nabla f(\boldsymbol{x}^* + t\boldsymbol{p}) \\ &\leq \boldsymbol{p}^\top \nabla f(\boldsymbol{x}^*), \end{aligned}$$

where the last inequality follow from the fact that $\boldsymbol{x}^*$ is a minimizer, that is,

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \text{ for all } \boldsymbol{x} \text{ in some neighborhood of } \boldsymbol{x}^*.$$

Let $\boldsymbol{p} = -\nabla f(\boldsymbol{x}^*)$ and it follows that $0 = \|\nabla f(\boldsymbol{x}^*)\|^2$, thus

$$\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}.$$

□

Theorem 2.4 (Second-order sufficient condition)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function of several variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ in some neighborhood of $\boldsymbol{x}^*$ and suppose that*

$$\begin{aligned} &\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}, \\ &\nabla^2 f(\boldsymbol{x}^*) \text{ is positive definite.} \end{aligned}$$

Then $\boldsymbol{x}^*$ is a local minimizer of f.

Proof. Because f is twice continuously differentiable and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite, we can consider an open ball B centered at $\boldsymbol{x}^*$ for which $\nabla^2 f(\boldsymbol{x})$ is positive definite. Taking any nonzero vector $\boldsymbol{p}$ such that $\boldsymbol{x}^* + \boldsymbol{p} \in B$ gives in Taylor's expansion

$$f(\boldsymbol{x}^* + \boldsymbol{p}) = f(\boldsymbol{x}^*) + \boldsymbol{p}^\top \nabla f(\boldsymbol{x}^*) + \frac{1}{2} \boldsymbol{p}^\top \nabla^2 f(\boldsymbol{x}^* + t\boldsymbol{p}) \boldsymbol{p},$$

where $t \in (0, 1)$. Now, since $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ and $\boldsymbol{p}^\top \nabla^2 f(\boldsymbol{x}^* + t\boldsymbol{p})\boldsymbol{p} > 0$ as $\boldsymbol{x}^* + t\boldsymbol{p} \in B$, it follows that

$$f(\boldsymbol{x}^* + \boldsymbol{p}) > f(\boldsymbol{x}^*),$$

thus $\boldsymbol{x}^*$ is a local minimizer of f. □

Lastly, we define Lipschitz continuity and notions of convergence. If not stated differently, $\|\cdot\|$ will denote the Euclidean vector and matrix norm, where the latter one is defined by $\|\boldsymbol{A}\| \coloneqq \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \|\boldsymbol{A}\boldsymbol{x}\| / \|\boldsymbol{x}\|$.

Definition 2.6. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called Lipschitz continuous with Lipschitz constant L, shortened as L-Lipschitz, if there exists a constant $L > 0$ such that*

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \leq L \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$$

for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$*. Furthermore, we say that* f *has an L-Lipschitz continuous gradient if* f *is continuously differentiable and*

$$\|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$$

for some $L > 0$ *and all* $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$*. Lastly, we say that* f *has an L-Lipschitz continuous Hessian if* f *is twice continuously differentiable and*

$$\|\nabla^2 f(\boldsymbol{x}_1) - \nabla^2 f(\boldsymbol{x}_2)\| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$$

for some $L > 0$ *and all* $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$*.*

Definition 2.7 (Rates of convergence)**.** *Suppose that* $\{\boldsymbol{x}_k\} \subset \mathbb{R}^n$ *is a sequence that converges to* $\boldsymbol{x}^*$*. We say that it converges:*

- *Q-linearly if there is a constant* $r \in (0, 1)$ *such that for all sufficiently large* k*,*
$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq r\|\boldsymbol{x}_k - \boldsymbol{x}^*\|,$$
- *Q-superlinearly if*
$$\lim_{k\to\infty} \frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} = 0,$$
- *Q-quadratically if there is a constant* $K > 0$ *such that for all sufficinetly large* k*,*
$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq K\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2.$$

When we talk about convergence in later chapters, we will omit the letter "Q" and simply say that a method, i.e. the sequence generated by it, converges linearly, superlinearly or quadratically. The reason why "Q" is mentioned is to emphasize that successive terms form a *quotient* in these definitions, which is standard in most literature.

2.2 Iterative Methods

Iterative methods provide fast and reliable answers to many problems where direct methods fail to yield solutions within a reasonable time. A general iterative method for optimization takes an initial value as a starting point then updates it according to some scheme until convergence is reached. Since we will be dealing with neural networks, suppose we are given an error function $E(\boldsymbol{w})$ associated with some network with parameters $\boldsymbol{w}$ which needs to be minimized. The goal of an iterative method would then be to take this parameter vector $\boldsymbol{w}$ and keep updating it until it reaches or approximates a minimum of $E(\boldsymbol{w})$, satisfying a predetermined precision condition. In other words, we update $\boldsymbol{w}$ iteratively via

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \boldsymbol{p}_k, \tag{2.1}$$

where $\boldsymbol{p}_k$ is an update vector. The main difficulty then becomes choosing $\boldsymbol{p}_k$ and $\boldsymbol{w}_0$ correctly such that $\boldsymbol{w}_{k+1}$ converges to a minimum of $E(\boldsymbol{w})$.

2.2.1 Line Search Methods

Further improvements can be introduced to the iteration formula (2.1) if we impose a *step length* $\alpha_k > 0$ on $\boldsymbol{p}_k$. One then obtains a *line search* method

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \tag{2.2}$$

and the optimization becomes a two-step problem, finding the update vector $\boldsymbol{p}_k$, then finding its step length α_k such that the univariate subproblem

$$\arg\min_{\alpha_k} f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \tag{2.3}$$

is solved as efficiently as possible without making the main problem (2.2) unnecessarily slow [5, 8]. For this reason, α_k is usually an approximate solution to (2.3) and is chosen from a set of candidates. In this context we will call $\boldsymbol{p}_k$ a *search direction* and it is often required that it is a *descent direction*. Namely, we want to ensure the reduction of $f(\boldsymbol{x}_{k+1})$ by setting

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) < f(\boldsymbol{x}_k). \tag{2.4}$$

We then linearly approximate $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)$ in (2.4) around $\boldsymbol{x}_k$ using Taylor's theorem to obtain

$$\begin{aligned} f(\boldsymbol{x}_k) + \alpha_k \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) &< f(\boldsymbol{x}_k) \\ \iff \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) &< 0. \end{aligned} \tag{2.5}$$

The definition of a descent direction is then deduced from (2.5).

Definition 2.8. *For a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ a vector $\boldsymbol{p}_k$ is called a descent direction at point $\boldsymbol{x}_k$ if*

$$\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) < 0.$$

When a descent direction $\boldsymbol{p}_k$ has been chosen, the problem becomes picking a good step size α_k. One of the famous techniques for doing this is choosing α_k to satisfy the *Wolfe conditions* [8].

Definition 2.9 (Wolfe conditions)**.** *For some constants $0 < c_1 < c_2 < 1$ a step size α_k is said to satisfy the Wolfe conditions for $f : \mathbb{R}^n \to \mathbb{R}$ if*

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \leq f(\boldsymbol{x}_k) + c_1 \alpha_k \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k), \tag{2.6}$$

$$\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \geq c_2 \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k). \tag{2.7}$$

The inequalities (2.6) and (2.7) are known as the *Armijo condition* and *curvature condition* respectively. The former one ensures that α_k gives a sufficient decrease in f, while the latter one ensures that α_k is not unreasonably small. For Newton and quasi-Newton methods, one typically sets $c_1 = 10^{-4}$ and $c_2 = 0.9$ [8].

We will now show the convergence of a general line search method for smooth functions as presented in [8].

Theorem 2.5. *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded below by some scalar $\hat{f}$, and ∇f is L-Lipschitz continuous. Let $\{\boldsymbol{x}_k\}$ be a sequance of iterates generated by a general line search method (2.2) where the step size α_k satisfies the Wolfe conditions (2.6) and (2.7), and $\boldsymbol{p}_k$ is a descent direction. Define angle θ_k between $\boldsymbol{p}_k$ and $-\nabla f(\boldsymbol{x}_k)$ as*

$$\cos\theta_k \coloneqq \frac{-\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k)}{\|\boldsymbol{p}_k\| \cdot \|\nabla f(\boldsymbol{x}_k)\|}.$$

Then it follows that

$$\sum_{k=0}^{\infty} \cos^2\theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 < \infty.$$

Proof. Starting with (2.7) and noticing that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$, we have that

$$\begin{aligned} & \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_{k+1}) \geq c_2 \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) \\ \iff & \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_{k+1}) - \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) \geq c_2 \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) - \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) \\ \iff & \boldsymbol{p}_k^\top (\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)) \geq (c_2 - 1) \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k). \end{aligned}$$

It follows from the Lipschitz continuity of ∇f that

$$\begin{aligned} & \|\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)\| \leq L \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| \\ \implies & \boldsymbol{p}_k^\top (\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)) \leq \boldsymbol{p}_k^\top L(\alpha_k \boldsymbol{p}_k) = \alpha_k L \|\boldsymbol{p}_k\|^2, \end{aligned}$$

hence

$$\alpha_k \geq \frac{\boldsymbol{p}_k^\top (\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k))}{L\|\boldsymbol{p}_k\|^2} \geq \frac{(c_2 - 1)\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k)}{L\|\boldsymbol{p}_k\|^2}.$$

The Armijo condition (2.6) and definition of $\cos\theta_k$, together with the above inequality, give now

$$\begin{aligned} & f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) + c_1 \alpha_k \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) \\ \iff & f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq c_1 \alpha_k \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) \\ & \qquad \leq c_1 \frac{(c_2 - 1)\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k)}{L\|\boldsymbol{p}_k\|^2} \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) \\ & \qquad = C \frac{(\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k))^2}{\|\boldsymbol{p}_k\|^2} = C \cos^2\theta_k \|\nabla f(\boldsymbol{x}_k)\|^2, \end{aligned}$$

where $C = c_1(c_2 - 1)/L$. Summing both sides above for $k = 0, \ldots, N$ gives now

$$\begin{aligned} \sum_{k=0}^{N} C \cos^2\theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 &\geq \sum_{k=0}^{N} (f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)) \\ &= f(\boldsymbol{x}_{N+1}) - f(\boldsymbol{x}_0) \\ &\geq \hat{f} - f(\boldsymbol{x}_0), \end{aligned}$$

where $\hat{f}$ is the assumed lower bound of f. Hence we get

$$\sum_{k=0}^{N} \cos^2 \theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{f(\boldsymbol{x}_0) - \hat{f}}{C}$$

$$\implies \sum_{k=0}^{\infty} \cos^2 \theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{f(\boldsymbol{x}_0) - \hat{f}}{C} < \infty.$$

□

Theorem 2.5 has important consequences on the descent direction $\boldsymbol{p}_k$ as it can imply global convergence for the respective line search method [8]. If the angle θ_k between $\boldsymbol{p}_k$ and $-\nabla f(\boldsymbol{x}_k)$ is strictly acute for all k, it follows that $\delta \leq \cos \theta_k \leq 1$ for some $\delta > 0$, hence

$$\begin{aligned} &\sum_{k=0}^{\infty} \cos^2 \theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 < \infty \\ \implies &\cos^2 \theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 \to 0 \\ \implies &\|\nabla f(\boldsymbol{x}_k)\| \to 0. \end{aligned}$$

2.2.2 Trust Region Methods

Another standard approach for pursuing (2.1) utilizes a quadratic approximation of the objective function around the current iterate $\boldsymbol{x}_k$. Here, a search direction $\boldsymbol{p}$ is computed to minimize the quadratic model

$$m_k(\boldsymbol{x}_k + \boldsymbol{p}) = f(\boldsymbol{x}_k) + \boldsymbol{p}^\top \nabla f(\boldsymbol{x}_k) + \frac{1}{2} \boldsymbol{p}^\top \boldsymbol{B}_k \boldsymbol{p}, \tag{2.8}$$

where f is the objective function and $\boldsymbol{B}_k$ a symmetric matrix (usually Hessian or some approximation of it if it is not possible to compute it exactly). Furthermore, we introduce a *trust region* constraint by bounding the Euclidean norm of $\boldsymbol{p}$ to a ball of predetermined radius Δ_k. Thus

$$\|\boldsymbol{p}\| \leq \Delta_k \tag{2.9}$$

is imposed, in which m_k can be *trusted* to accurately approximate f. A trust region method is then defined as the constrained optimization problem

$$\min_{\boldsymbol{p} \in \mathbb{R}^n} m_k(\boldsymbol{x}_k + \boldsymbol{p}) \quad \text{subject to } \|\boldsymbol{p}\| \leq \Delta_k. \tag{2.10}$$

For trust region methods, it is necessary to redefine (2.9) in each iteration in order for $f(\boldsymbol{x}_{k+1})$ to achieve an efficient reduction. If the reduction is not achieved for a given radius Δ_k it is important to reject it and reshape the trust

region with another choice of radius. The standard way to do this is to compute a ratio of the *actual reduction* and *predicted reduction*

$$\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})}{m_k(\boldsymbol{0}) - m_k(\boldsymbol{p}_k)} \tag{2.11}$$

and look at its values [5, 8]. One way to adjust the trust region radius Δ_k is to introduce control parameters $0 \leq c_1 < c_2 < 1$, $b_1 > 1$, $b_2 \in (0, 1)$, $d \in (0, 1)$, then [1]:

- set $\Delta_{k+1} = \Delta_k$ if $\rho_k > c_2$ and $\|\boldsymbol{p}_k\| \leq d\Delta_k$,
- set $\Delta_{k+1} = b_1\Delta_k$ if $\rho_k > c_2$ and $\|\boldsymbol{p}_k\| > d\Delta_k$,
- set $\Delta_{k+1} = \Delta_k$ if $c_1 \leq \rho_k \leq c_2$,
- set $\Delta_{k+1} = b_2\Delta_k$ if $\rho_k < c_1$.

The trust region scheme is thus carried out as (2.1) if for some predetermined constant $\eta > 0$ we have $\rho_k \geq \eta$. If $\rho_k < \eta$, then the iterate is left unchanged as $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$. At the end of each iteration, ρ_k is moderated according to the above list.

We present a theorem which ensures that trust region methods converge globally under certain assumptions [5].

Theorem 2.6. *Let $\{\boldsymbol{x}_k\}$ be a sequence of iterates generated by $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}_k$, where $\boldsymbol{p}_k$ is computed by solving the trust region subproblem (2.10) for which the trust radius ρ_k is moderated by the list following its definition in (2.11). Assume that $\boldsymbol{p}_k$ satisfies*

$$m_k(\boldsymbol{0}) - m_k(\boldsymbol{p}_k) \geq \xi\|\nabla f(\boldsymbol{x}_k)\| \min\left\{\frac{\|\nabla f(\boldsymbol{x}_k)\|}{\beta_k}, \Delta_k\right\}, \tag{2.12}$$

for some $\zeta > 0$, $\beta_k > 0$, and the sequence of Hessian approximations $\{\boldsymbol{B}_k\}$ is bounded in norm. Lastly, assume that $\nabla f(\boldsymbol{x})$ is L-Lipschitz continuous. Then either one of the following three possibilities hold,

$$\begin{aligned}
&\nabla f(\boldsymbol{x}_k) = 0 \text{ for some } k,\\
&\lim_{k\to\infty} \nabla f(\boldsymbol{x}_k) = 0,\\
&f \text{ is unbounded from below.}
\end{aligned}$$

2.2.3 Newton's Method and Rise of Quasi-Newton Methods

One of the classic iterative methods for root finding is Newton's method. It is based on a linear approximation of an objective function $\boldsymbol{F}$ and can be regarded as an unconstrained optimization method if we assume Theorem 2.4 and apply the method on $\boldsymbol{F} = \nabla f(\boldsymbol{x}) = \boldsymbol{0}$. However, the iteration must start sufficiently close to the minimizer in order for it to converge to that minimizer [5, 6, 8].

Given a step increment $\Delta \boldsymbol{x} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, we can linearly approximate the gradient of a twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ around the current iterate $\boldsymbol{x}_k$ using Taylor's theorem, then evaluate it at the new iterate $\boldsymbol{x}_{k+1}$, hence

$$\nabla f(\boldsymbol{x}_{k+1}) \approx \nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)\Delta \boldsymbol{x}.$$

Newton's method is then obtain via

$$\begin{aligned} &\nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)\Delta \boldsymbol{x} = \boldsymbol{0} \\ \iff &\Delta \boldsymbol{x} = -(\nabla^2 f(\boldsymbol{x}_k))^{-1}\nabla f(\boldsymbol{x}_k), \end{aligned}$$

which gives the update scheme

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\nabla^2 f(\boldsymbol{x}_k))^{-1}\nabla f(\boldsymbol{x}_k). \tag{2.13}$$

One can show under the assumptions of Theorem 2.4 and by picking the initial $\boldsymbol{x}_0$ sufficiently close to the minimizer, which ensures the positive definiteness, that Newton's method converges quadratically [5, 6, 8]. It should be noted, however, that not every function will satisfy positive definiteness of the Hessian, and it is often required that it has some special property, e.g. being convex.

We will now prove the convergence of Newton's method as it is done in [5].

Lemma 2.7. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$, ∇f and $\nabla^2 f$ are Lipschitz continuous, $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite. Then there exists a ball $B_\delta(\boldsymbol{x}^*)$ of radius $\delta > 0$ centered at $\boldsymbol{x}^*$ such that for all $\boldsymbol{x} \in B_\delta(\boldsymbol{x}^*)$ the following statements hold:*

$$\|\nabla^2 f(\boldsymbol{x})\| \leq 2\|\nabla^2 f(\boldsymbol{x}^*)\|, \tag{2.14}$$

$$\|(\nabla^2 f(\boldsymbol{x}))^{-1}\| \leq 2\|(\nabla^2 f(\boldsymbol{x}^*))^{-1}\| \tag{2.15}$$

and

$$\frac{\|\boldsymbol{x} - \boldsymbol{x}^*\|}{2\|(\nabla^2 f(\boldsymbol{x}^*))^{-1}\|} \leq \|\nabla f(\boldsymbol{x})\| \leq 2\|\nabla^2 f(\boldsymbol{x}^*)\| \cdot \|\boldsymbol{x} - \boldsymbol{x}^*\|. \tag{2.16}$$

Proof. (See [4].) □

Theorem 2.8. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$, ∇f and $\nabla^2 f$ are Lipschitz continuous, $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite. Then there exist $K > 0$ and $\delta > 0$ such that if $\boldsymbol{x}_k \in B_\delta(\boldsymbol{x}^*)$, it follows that*

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq K\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2, \tag{2.17}$$

where $\boldsymbol{x}_{k+1}$ is given by the Newton scheme (2.13).

Proof. Choose δ such that Lemma 2.7 holds. The Newton scheme (2.13) gives

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}^* = \boldsymbol{x}_k - \boldsymbol{x}^* - (\nabla^2 f(\boldsymbol{x}_k))^{-1}\nabla f(\boldsymbol{x}_k).$$

Let $\boldsymbol{e}_{k+1} = \boldsymbol{x}_{k+1} - \boldsymbol{x}^*$, $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$. It follows from Taylor's theorem that

$$\begin{aligned}
\boldsymbol{e}_{k+1} &= (\nabla^2 f(\boldsymbol{x}_k))^{-1} \int_0^1 (\nabla^2 f(\boldsymbol{x}_k) - \nabla^2 f(\boldsymbol{x}^* + t\boldsymbol{e}_k)\boldsymbol{e}_k)dt \\
\iff \|\boldsymbol{e}_{k+1}\| &= \|(\nabla^2 f(\boldsymbol{x}_k))^{-1}\| \cdot \left\| \int_0^1 (\nabla^2 f(\boldsymbol{x}_k) - \nabla^2 f(\boldsymbol{x}^* + t\boldsymbol{e}_k)\boldsymbol{e}_k)dt \right\| \\
&\leq \|(\nabla^2 f(\boldsymbol{x}_k))^{-1}\| \cdot \int_0^1 \|(\nabla^2 f(\boldsymbol{x}_k) - \nabla^2 f(\boldsymbol{x}^* + t\boldsymbol{e}_k)\| \cdot \|\boldsymbol{e}_k\| dt.
\end{aligned}$$

Now, since f is twice L-Lipschitz continuously differentiable, we get

$$\begin{aligned}
\|\boldsymbol{e}_{k+1}\| &\leq \|(\nabla^2 f(\boldsymbol{x}_k))^{-1}\| \cdot \int_0^1 \|(\nabla^2 f(\boldsymbol{x}_k) - \nabla^2 f(\boldsymbol{x}^* + t\boldsymbol{e}_k)\| \cdot \|\boldsymbol{e}_k\| dt \\
&\leq \|(\nabla^2 f(\boldsymbol{x}_k))^{-1}\| \cdot \int_0^1 L\|\boldsymbol{x}_k - (\boldsymbol{x}^* + t\boldsymbol{e}_k)\| \cdot \|\boldsymbol{e}_k\| dt \\
&= \|(\nabla^2 f(\boldsymbol{x}_k))^{-1}\| \cdot \int_0^1 L\|\boldsymbol{e}_k\|^2 \cdot \|1 - t\| dt \\
&= \frac{L}{2}\|(\nabla^2 f(\boldsymbol{x}_k))^{-1}\| \cdot \|\boldsymbol{e}_k\|^2.
\end{aligned}$$

It follows from Lemma 2.7 that

$$\begin{aligned}
\|\boldsymbol{e}_{k+1}\| &\leq \frac{L}{2}\|(\nabla^2 f(\boldsymbol{x}_k))^{-1}\| \cdot \|\boldsymbol{e}_k\|^2 \\
&\leq \frac{L}{2} 2\|(\nabla^2 f(\boldsymbol{x}^*))^{-1}\| \cdot \|\boldsymbol{e}_k\|^2,
\end{aligned}$$

thus

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq L\|(\nabla^2 f(\boldsymbol{x}^*))^{-1}\| \cdot \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2$$

and (2.17) follows for $K = L\|(\nabla^2 f(\boldsymbol{x}^*))^{-1}\|$. □

A direct consequence of Theorem 2.8 is the following theorem which shows that Newton's method converges quadratically.

Theorem 2.9. *Suppose that the assumptions of Theorem 2.8 hold and also choose $\delta > 0$ such that $K\delta < 1$. Then Newton's method (2.13) converges quadratically to $\boldsymbol{x}^*$ if $\boldsymbol{x}_0 \in B_\delta(\boldsymbol{x}^*)$.*

Proof. It follows from Theorem 2.8 that

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq K\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2$$

if $\boldsymbol{x}_k \in B_\delta(\boldsymbol{x}^*)$. Since $\boldsymbol{x}_0 \in B_\delta(\boldsymbol{x}^*)$, we have that $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| \leq \delta$, thus

$$\|\boldsymbol{x}_1 - \boldsymbol{x}^*\| \leq K\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 \leq K\delta\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| < \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|.$$

We get that $\boldsymbol{x}_1 \in B_{K\delta^2}(\boldsymbol{x}^*) \subset B_\delta(\boldsymbol{x}^*)$, from which it follows by successive application of Theorem 2.8 and the above inequality that

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq K\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 \leq K\delta\|\boldsymbol{x}_k - \boldsymbol{x}^*\| < \|\boldsymbol{x}_k - \boldsymbol{x}^*\| \tag{2.18}$$

for all $k \geq 0$. Thus every new iterate $\boldsymbol{x}_{k+1}$ will be contained in a ball centered at $\boldsymbol{x}^*$ of radius δ and (2.18) implies that $\boldsymbol{x}_k \to \boldsymbol{x}^*$ quadratically. □

There are many ways in which Newton's method can be improved to be more robust to broader choices of the initial iterate $\boldsymbol{x}_0$. Some of them are based on the techniques we have discussed in this chapter, namely line search and trust region methods. In the context of line search, (2.2) is solved by letting the search direction $\boldsymbol{p}_k$ be the one we have obtained in (2.13), that is,

$$\boldsymbol{p}_k = -(\nabla^2 f(\boldsymbol{x}_k))^{-1} \nabla f(\boldsymbol{x}_k), \tag{2.19}$$

and the step length α_k is computed to satisfy the Wolfe (or some other) conditions. However, if $\nabla^2 f(\boldsymbol{x}_k)$ is not positive definite, we omit it and compute some positive definite approximation of it [8]. On the other hand, the trust region approach is very straight forward. The quadratic model m_k in (2.8) does not require $\boldsymbol{B}_k$ to be positive definite, thus (2.19) need not be changed and we can proceed to the trust region subproblem (2.10) directly.

Computing the Hessian of an objective function f is not always easy or possible. We have seen that Newton's method relies heavily on it and we seek to find a solution to the scenarios where it is necessary to approximate it. This problem introduces *quasi-Newton* methods where the Hessian of the objective function in Newton's update scheme (2.13) is successively approximated in each iteration [5, 8]. The new update scheme then becomes

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{B}_k^{-1} \nabla f(\boldsymbol{x}_k), \tag{2.20}$$

where $\boldsymbol{B}_k \approx \nabla^2 f(\boldsymbol{x}_k)$, and we call (2.20) a *quasi-Newton* method.

2.3 Neural Networks

Neural networks are a special type of compositions of functions used to extract or predict information from a given data set. One feeds the observed information to the network's input layer, the network then processes the information in a multiple of so called hidden layers, and lastly it returns the output values. If there are more than one hidden layers, the network is called *deep* [3].

Mathematically, we have input variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ and a parameter vector $\boldsymbol{w}$ that we use to construct a network model $y(\boldsymbol{x}, \boldsymbol{w})$ in order to approximate another function. This goal function can be anything from simple mathematical functions to various types of classifiers and image or speech recognition.

2.3.1 Feed-forward Networks

A special type of neural networks that never processes data backwards, i.e. sends data from a layer to the previous layers, is called a *feed-forward* network. These types of networks are excellent approximators of highly nonlinear functions and will therefore be of interest for us [2, 3].

Suppose we are given N_1 input variables $\boldsymbol{x} = (x_1, \ldots, x_{N_1})$ and want to build a feed-forward network. We need to define how many nodes the first hidden layer will be comprised of. Suppose it has N_2 nodes, then we can use the input variables to define the k-th node in the first layer as

$$c_k(\boldsymbol{x}) = \sum_{n=1}^{N_1} w_{kn} x_n + w_{k0}, \quad k = 1, \ldots, N_2, \tag{2.21}$$

where w_{kn} are parameters called *weights* and w_{k0} parameters called *biases*. In other words, nodes are linear combinations of weights w and input variables x with additional biases. The next step would typically include a differentiable activation function h_1 which is used to transform (2.21) into $h_1(c_k(\boldsymbol{x}))$. Thus

$$h_1(c_k(\boldsymbol{x})) = h_1\left(\sum_{n=1}^{N_1} w_{kn} x_n + w_{k0}\right) \tag{2.22}$$

is obtained as the k-th node, which for every $k = 1, \ldots, N_2$ defines the first hidden layer. Now, by treating (2.22) as our new input values we can proceed to the definition of the second hidden layer consisting of N_3 nodes as

$$c_j(\boldsymbol{x}) = \sum_{k=1}^{N_2} w_{jk} h_1(c_k(\boldsymbol{x})) + w_{j0}, \quad j = 1, \ldots, N_3. \tag{2.23}$$

Similarly, we introduce another differentiable activation function h_2 and transform (2.23) into

$$\begin{aligned} h_2(c_j(\boldsymbol{x})) &= h_2\left(\sum_{k=1}^{N_2} w_{jk} h_1(c_k(\boldsymbol{x})) + w_{j0}\right) \\ &= h_2\left(\sum_{k=1}^{N_2} w_{jk} h_1\left(\sum_{n=1}^{N_1} w_{kn} x_n + w_{k0}\right) + w_{j0}\right). \end{aligned} \tag{2.24}$$

It is not generally specified how many layers or nodes per layer a network should have and is something that should be determined in advance. One could keep linearly transforming (2.24) until final outputs $h_i(c(\boldsymbol{x}))$ are reached. We will denote these final outputs as $y_i(\boldsymbol{x}, \boldsymbol{w})$, which together form the output layer

$$\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}). \tag{2.25}$$

2.3.2 Network Training

Training a neural network is crucial to its performance and directly determines how well a certain problem the network is constructed for will be resolved. The central part in training is to find weight parameters $\boldsymbol{w}$ such that some error function $E(\boldsymbol{w})$ is minimized. This function typically involves a sum of squared residuals, coming from an important probabilistic background [2, 3].

Suppose we have a set of some observed input values $\{\boldsymbol{x}_n\}$ together with the corresponding set of outputs $\{\boldsymbol{t}_n\}$. This type of training where the output values are known is called *supervised learning*. The first step would typically concern splitting the observed values into training and testing sets, e.g. a 75%-25% split, however, further validation could be applied [2]. The training set, which we will denote by $T = \{(\boldsymbol{x}_k, \boldsymbol{t}_k)\}$, is then used on a network model (2.25) to define an error function

$$E_k(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y}(\boldsymbol{x}_k, \boldsymbol{w}) - \boldsymbol{t}_k\|^2.$$

The error function for the whole training set is then defined as

$$E(\boldsymbol{w}) = \frac{1}{M}\sum_{k=1}^{M} E_k(\boldsymbol{w}), \tag{2.26}$$

where M is the number of training pairs in T, i.e., $M = |T|$.

Minimizing $E(\boldsymbol{w})$ in (2.26) with respect to weight parameters $\boldsymbol{w}$ is what we will denote as *network training*. After the parameters have been chosen, another error is calculated for the testing set to check the network's performance on unseen data.

For a general network model $\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w})$, the error function $E(\boldsymbol{w})$ need not be convex. By using nonlinear activation functions, $E(\boldsymbol{w})$ inherits highly nonlinear dependence on the weights $\boldsymbol{w}$ [2]. What this implies in practice is that a minimizer of $E(\boldsymbol{w})$ is not always a global minimizer and we cannot use Theorem 2.1. For this reason, it is usually necessary to compare different minima obtained by using various initial values $\boldsymbol{w}_0$ in an iterative method and see which one yields the best results.

3 Quasi-Newton Methods

We saw how quasi-Newton methods arise naturally as a consequence of inability or inefficiency of computing the Hessian of an objective function. In this chapter, we will explore different approaches to the quasi-Newton update scheme (2.20) and see how each one of them approximates the Hessian. We will also see some of the recent work in this field involving substantial improvements to some of the classic quasi-Newton methods applied on neural network training [1, 7].

3.1 BFGS Method

One well established quasi-Newton method, named after its inventors, is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method [5, 8, 9].

Let us come back to the standard quasi-Newton scheme in (2.20), namely

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{B}_k^{-1}\nabla f(\boldsymbol{x}_k).$$

Here, the search direction is given by

$$\boldsymbol{p}_k = -\boldsymbol{B}_k^{-1}\nabla f(\boldsymbol{x}_k),$$

and it will be used in BFGS as a descent direction for the line search scheme

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{B}_k^{-1}\nabla f(\boldsymbol{x}_k), \tag{3.1}$$

where α_k is chosen to satisfy the Wolfe conditions. What is characteristic about BFGS in its approximations is that it uses the information of latest two gradient evaluations $\nabla f(\boldsymbol{x}_k)$ and $\nabla f(\boldsymbol{x}_{k-1})$ to impose a *secant condition* on $\boldsymbol{B}_k$ [8]. In other words, the new Hessian approximation should satisfy

$$\boldsymbol{B}_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k, \tag{3.2}$$

where $(\boldsymbol{s}_k, \boldsymbol{y}_k)$ are called the *curvature pairs* and are defined as

$$\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k, \quad \boldsymbol{y}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k). \tag{3.3}$$

We require from $\boldsymbol{B}_{k+1}$ to be symmetric positive definite, which means that the curvature pairs must satisfy the *curvature condition*

$$\boldsymbol{s}_k^\top \boldsymbol{y}_k > 0, \tag{3.4}$$

as seen from (3.2) since $\boldsymbol{s}_k^\top \boldsymbol{B}_{k+1}\boldsymbol{s}_k = \boldsymbol{s}_k^\top \boldsymbol{y}_k > 0$ needs to hold.

The following theorem shows that every curvature pair incorporating a step size which satisfies the Wolfe conditions will automatically satisfy the curvature condition (3.4).

Theorem 3.1. *If a step size $\alpha_k > 0$ satisfies the Wolfe conditions in a quasi-Newton scheme (3.1), then every curvature pair $(\boldsymbol{s}_k, \boldsymbol{y}_k)$ will satisfy the curvature condition (3.4).*

Proof. Since α_k satisfies the Wolfe conditions, it follows directly from (2.7) that

$$\begin{aligned}
&\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_{k+1}) \geq c_2 \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) \\
\iff\ &\alpha_k^{-1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^\top \nabla f(\boldsymbol{x}_{k+1}) \geq c_2 \alpha_k^{-1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^\top \nabla f(\boldsymbol{x}_k) \\
\iff\ &\boldsymbol{s}_k^\top \nabla f(\boldsymbol{x}_{k+1}) \geq c_2 \boldsymbol{s}_k^\top \nabla f(\boldsymbol{x}_k) \\
\iff\ &\boldsymbol{s}_k^\top (\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)) \geq (c_2 - 1)\boldsymbol{s}_k^\top \nabla f(\boldsymbol{x}_k) \\
\iff\ &\boldsymbol{s}_k^\top \boldsymbol{y}_k \geq (c_2 - 1)\alpha_k \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k).
\end{aligned}$$

Now, since $\boldsymbol{p}_k$ is a descent direction $\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) < 0$ and $0 < c_2 < 1$, we have

$$\boldsymbol{s}_k^\top \boldsymbol{y}_k \geq (c_2 - 1)\alpha_k \boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k) > 0.$$

□

To compute $\boldsymbol{B}_{k+1}$, BFGS applies a rank-two update to $\boldsymbol{B}_k$ using symmetric matrices in order to preserve symmetry. A general rank-two update is given by

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + a\boldsymbol{u}\boldsymbol{u}^\top + b\boldsymbol{v}\boldsymbol{v}^\top. \tag{3.5}$$

Since the secant condition is required, we substitute (3.5) into (3.2) to obtain

$$\begin{aligned} & \boldsymbol{y}_k = \boldsymbol{B}_{k+1}\boldsymbol{s}_k \\ \iff\ & \boldsymbol{y}_k = (\boldsymbol{B}_k + a\boldsymbol{u}\boldsymbol{u}^\top + b\boldsymbol{v}\boldsymbol{v}^\top)\boldsymbol{s}_k \\ & \quad = \boldsymbol{B}_k\boldsymbol{s}_k + a\boldsymbol{u}(\boldsymbol{u}^\top\boldsymbol{s}_k) + b\boldsymbol{v}(\boldsymbol{v}^\top\boldsymbol{s}_k). \end{aligned} \tag{3.6}$$

In order for this equality to hold, we must set $\boldsymbol{u} = -\boldsymbol{B}_k\boldsymbol{s}_k$ and $\boldsymbol{v} = \boldsymbol{y}_k$, thus scalars a and b must satisfy

$$a = 1/(\boldsymbol{u}^\top\boldsymbol{s}_k), \quad b = 1/(\boldsymbol{v}^\top\boldsymbol{s}_k).$$

The rank-two update formula (3.5) is therefore given by

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k - \frac{\boldsymbol{B}_k\boldsymbol{s}_k\boldsymbol{s}_k^\top\boldsymbol{B}_k}{\boldsymbol{s}_k^\top\boldsymbol{B}_k\boldsymbol{s}_k} + \frac{\boldsymbol{y}_k\boldsymbol{y}_k^\top}{\boldsymbol{y}_k^\top\boldsymbol{s}_k}. \tag{3.7}$$

If we let $\boldsymbol{H}_{k+1} \coloneqq \boldsymbol{B}_{k+1}^{-1}$, we can apply the Sherman–Morrison–Woodbury formula [8] on (3.7) to obtain

$$\boldsymbol{H}_{k+1} = \boldsymbol{V}_k^\top\boldsymbol{H}_k\boldsymbol{V}_k + \rho_k\boldsymbol{s}_k\boldsymbol{s}_k^\top, \tag{3.8}$$

where $\rho_k = 1/\boldsymbol{y}_k^\top\boldsymbol{s}_k$ and $\boldsymbol{V}_k = \boldsymbol{I} - \rho_k\boldsymbol{y}_k\boldsymbol{s}_k^\top$ (see [8]). The BFGS scheme is therefore given by (3.1), where $\boldsymbol{B}_k^{-1} \coloneqq \boldsymbol{H}_k$ is computed according to (3.8).

The following theorem ensures that the BFGS method (3.1), where the inverse Hessians are generated by (3.8), establishes a superlinear rate of convergence when the objective function is smooth and convex. The proof and further insight can be found in [8].

Theorem 3.2. *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. Let $\boldsymbol{x}_0$ be a starting point for the BFGS scheme (3.1), where the Hessians are generated by (3.7) and $\boldsymbol{B}_0$ is a symmetric positive definite initial matrix. Assume that the level set $\Omega = \{\boldsymbol{x} : f(\boldsymbol{x}) \le f(\boldsymbol{x}_0)\}$ is convex and there exist constants $0 < m \le L$ such that*

$$m\|\boldsymbol{z}\|^2 \le \boldsymbol{z}^\top\nabla^2 f(\boldsymbol{x})\boldsymbol{z} \le L\|\boldsymbol{z}\|^2$$

for all $\boldsymbol{x} \in \Omega$ and $\boldsymbol{z} \in \mathbb{R}^n$. Then the sequence of iterates $\{\boldsymbol{x}_k\}$ generated by the BFGS scheme (3.1) converges to the minimizer $\boldsymbol{x}^*$. Furthermore, assume that $\nabla^2 f(\boldsymbol{x})$ is Lipschitz continuous at $\boldsymbol{x}^*$ and that

$$\sum_{k=1}^{\infty} \|\boldsymbol{x}_k - \boldsymbol{x}^*\| < \infty$$

holds, then $\boldsymbol{x}_k \to \boldsymbol{x}^*$ superlinearly.

3.1.1 Nesterov's Accelerated Quasi-Newton Method

One way of making the standard BFGS scheme (3.1) more robust to large-scale problems, where the inverse Hessian approximations are constructed via (3.8), is by incorporating Nesterov's accelerated gradient in the search direction [7].

Consider the derivation which lead to the Newton's scheme (2.13). Instead of using the standard increment $\Delta \boldsymbol{x} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, we introduce a so called *momentum term* $\mu \boldsymbol{p}_k$, where $\mu \in (0, 1)$ is a *momentum coefficient* and $\boldsymbol{p}_k$ is a search direction, and define $\Delta \boldsymbol{x} = \boldsymbol{x}_{k+1} - (\boldsymbol{x}_k + \mu \boldsymbol{p}_k)$. By linearly approximating the gradient around $\boldsymbol{x}_k + \mu \boldsymbol{p}_k$ and evaluating it at $\boldsymbol{x}_{k+1}$, we get

$$\nabla f(\boldsymbol{x}_{k+1}) \approx \nabla f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k) + \nabla^2 f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k) \Delta \boldsymbol{x}.$$

By setting this approximation to zero, we obtain Newton's method with the momentum term $\mu \boldsymbol{p}_k$, that is,

$$\begin{aligned} \boldsymbol{0} &= \nabla f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k) + \nabla^2 f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k) \Delta \boldsymbol{x} \\ \iff \Delta \boldsymbol{x} &= -(\nabla^2 f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k))^{-1} \nabla f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k) \\ \iff \boldsymbol{x}_{k+1} &= (\boldsymbol{x}_k + \mu \boldsymbol{p}_k) - (\nabla^2 f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k))^{-1} \nabla f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k). \end{aligned}$$

The Hessians $\nabla^2 f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k)$ will be successively approximated by a rank-two BFGS based scheme

$$\tilde{\boldsymbol{B}}_{k+1} = \tilde{\boldsymbol{B}}_k - \frac{\tilde{\boldsymbol{B}}_k \tilde{\boldsymbol{s}}_k \tilde{\boldsymbol{s}}_k^\top \tilde{\boldsymbol{B}}_k}{\tilde{\boldsymbol{s}}_k^\top \tilde{\boldsymbol{B}}_k \tilde{\boldsymbol{s}}_k} + \frac{\tilde{\boldsymbol{y}}_k \tilde{\boldsymbol{y}}_k^\top}{\tilde{\boldsymbol{y}}_k^\top \tilde{\boldsymbol{s}}_k}, \tag{3.9}$$

in which the curvature pairs (3.3) get redefined to $(\tilde{\boldsymbol{s}}_k, \tilde{\boldsymbol{y}}_k)$ such that

$$\tilde{\boldsymbol{s}}_k = \boldsymbol{x}_{k+1} - (\boldsymbol{x}_k + \mu \boldsymbol{p}_k), \quad \tilde{\boldsymbol{y}}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k). \tag{3.10}$$

By applying the Sherman–Morrison–Woodbury formula [8] on (3.9) and letting $\tilde{\boldsymbol{H}}_{k+1} \coloneqq \tilde{\boldsymbol{B}}_{k+1}^{-1}$, we obtain

$$\tilde{\boldsymbol{H}}_{k+1} = \tilde{\boldsymbol{V}}_k^\top \tilde{\boldsymbol{H}}_k \tilde{\boldsymbol{V}}_k + \sigma_k \tilde{\boldsymbol{s}}_k \tilde{\boldsymbol{s}}_k^\top, \tag{3.11}$$

where $\sigma_k = 1/\tilde{\boldsymbol{y}}_k^\top \tilde{\boldsymbol{s}}_k$ and $\boldsymbol{V}_k = \boldsymbol{I} - \sigma_k \tilde{\boldsymbol{y}}_k \tilde{\boldsymbol{s}}_k^\top$ [7, 8]. The Nesterov's accelerated quasi-Newton scheme (NAQ) is thus, in regards to (3.1), given by

$$\boldsymbol{x}_{k+1} = (\boldsymbol{x}_k + \mu \boldsymbol{p}_k) - \alpha_k \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k),$$

where α_k is a step size. This leads to a new search direction given by

$$\boldsymbol{p}_{k+1} = \mu \boldsymbol{p}_k - \alpha_k \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_k + \mu \boldsymbol{p}_k), \tag{3.12}$$

and NAQ can be stated as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}_{k+1}, \tag{3.13}$$

where $\boldsymbol{p}_{k+1}$ is computed via (3.12). Aside from initial $\boldsymbol{x}_0$ and $\tilde{\boldsymbol{H}}_0$, NAQ will also require a momentum parameter μ in advance, as well as an initial $\boldsymbol{p}_0$ which is set to $\mathbf{0}$ in [7].

The convergence for proposed NAQ follows from the convergence of BFGS by verifying that $\tilde{\boldsymbol{B}}_{k+1}$ given by (3.9) preserves symmetric positive definiteness if $\tilde{\boldsymbol{B}}_k$ is symmetric positive definite, and if the secant equation $\tilde{\boldsymbol{y}}_k = \tilde{\boldsymbol{B}}_{k+1}\tilde{\boldsymbol{s}}_k$ holds for it. We show the derivation from [7].

Theorem 3.3. *The curvature pairs $(\tilde{\boldsymbol{s}}_k, \tilde{\boldsymbol{y}}_k)$ given by (3.10) and generated by the NAQ scheme* (3.13) *satisfy the curvature condition $\tilde{\boldsymbol{s}}_k^\top \tilde{\boldsymbol{y}}_k > 0$.*

Proof. For NAQ, the curvature condition (3.4) is given by

$$\begin{aligned}
0 &< \tilde{\boldsymbol{s}}_k^\top \tilde{\boldsymbol{y}}_k \\
&= (\boldsymbol{x}_{k+1} - (\boldsymbol{x}_k + \mu\boldsymbol{p}_k))^\top (\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)) \\
&= (-\alpha_k \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k))^\top (\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)) \\
&= (-\alpha_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)^\top \tilde{\boldsymbol{H}}_k)(\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)) \\
&= -\alpha_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)^\top \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_{k+1}) + \alpha_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)^\top \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k).
\end{aligned}$$

Consider now the exact line search minimization subproblem, analogously to (2.3) but applied on (3.13),

$$\arg\min_{\alpha_k} f(\boldsymbol{x}_k + \boldsymbol{p}_{k+1}).$$

The solution to this problem gives an optimal step size α_k and is obtained by solving

$$\begin{aligned}
\frac{df(\boldsymbol{x}_k + \boldsymbol{p}_{k+1})}{d\alpha_k} = \frac{df(\boldsymbol{x}_k + \mu\boldsymbol{p}_k - \alpha_k \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k))}{d\alpha_k} &= 0 \\
\iff -\nabla f(\boldsymbol{x}_{k+1})^\top \alpha_k \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k) &= 0 \\
\iff -\alpha_k \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)^\top \nabla f(\boldsymbol{x}_{k+1}) &= 0.
\end{aligned}$$

It follows from the required curvature condition $0 < \tilde{\boldsymbol{s}}_k^\top \tilde{\boldsymbol{y}}_k$ that

$$\begin{aligned}
0 &< -\alpha_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)^\top \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_{k+1}) + \alpha_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)^\top \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k) \\
&= \alpha_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k)^\top \tilde{\boldsymbol{H}}_k \nabla f(\boldsymbol{x}_k + \mu\boldsymbol{p}_k),
\end{aligned}$$

and since $\tilde{\boldsymbol{H}}_k$ is the inverse of a positive definite matrix $\tilde{\boldsymbol{B}}_k$, it is also positive definite, hence $0 < \tilde{\boldsymbol{s}}_k^\top \tilde{\boldsymbol{y}}_k$ will be satisfied. □

Theorem 3.4. *Let $\tilde{\boldsymbol{B}}_{k+1}$ be a Hessian approximation generated by (3.9). If $\tilde{\boldsymbol{B}}_k$ is symmetric positive definite, then $\tilde{\boldsymbol{B}}_{k+1}$ will be symmetric positive definite and satisfy the secant condition $\tilde{\boldsymbol{y}}_k = \tilde{\boldsymbol{B}}_{k+1}\tilde{\boldsymbol{s}}_k$, where the curvature pairs $(\tilde{\boldsymbol{s}}_k, \tilde{\boldsymbol{y}}_k)$ are given by (3.10).*

Proof. For the secant condition, we have from (3.9) that

$$\begin{aligned}\tilde{\boldsymbol{y}}_k &= \tilde{\boldsymbol{B}}_{k+1}\tilde{\boldsymbol{s}}_k \\ &= \left(\tilde{\boldsymbol{B}}_k - \frac{\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k}{\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k} + \frac{\tilde{\boldsymbol{y}}_k\tilde{\boldsymbol{y}}_k^\top}{\tilde{\boldsymbol{y}}_k^\top\tilde{\boldsymbol{s}}_k}\right)\tilde{\boldsymbol{s}}_k \\ &= \tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k - \frac{\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k}{\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k} + \frac{\tilde{\boldsymbol{y}}_k\tilde{\boldsymbol{y}}_k^\top\tilde{\boldsymbol{s}}_k}{\tilde{\boldsymbol{y}}_k^\top\tilde{\boldsymbol{s}}_k} \\ &= \tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k - \tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k + \tilde{\boldsymbol{y}}_k = \tilde{\boldsymbol{y}}_k.\end{aligned}$$

The symmetry follows as well since

$$\begin{aligned}\tilde{\boldsymbol{B}}_{k+1}^\top &= \left(\tilde{\boldsymbol{B}}_k - \frac{\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k}{\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k} + \frac{\tilde{\boldsymbol{y}}_k\tilde{\boldsymbol{y}}_k^\top}{\tilde{\boldsymbol{y}}_k^\top\tilde{\boldsymbol{s}}_k}\right)^\top \\ &= \tilde{\boldsymbol{B}}_k^\top - \frac{\tilde{\boldsymbol{B}}_k^\top\tilde{\boldsymbol{s}}_k\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k^\top}{\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k} + \frac{\tilde{\boldsymbol{y}}_k\tilde{\boldsymbol{y}}_k^\top}{\tilde{\boldsymbol{y}}_k^\top\tilde{\boldsymbol{s}}_k} \\ &= \tilde{\boldsymbol{B}}_k - \frac{\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k}{\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{B}}_k\tilde{\boldsymbol{s}}_k} + \frac{\tilde{\boldsymbol{y}}_k\tilde{\boldsymbol{y}}_k^\top}{\tilde{\boldsymbol{y}}_k^\top\tilde{\boldsymbol{s}}_k} = \tilde{\boldsymbol{B}}_{k+1}.\end{aligned}$$

To show positive definiteness, consider the Cholesky decomposition $\tilde{\boldsymbol{B}}_{k+1} = \boldsymbol{L}\boldsymbol{L}^\top$, where $\boldsymbol{L}$ is some lower triangular matrix [8]. For an arbitrary nonzero vector $\boldsymbol{z}$, it follows that

$$\begin{aligned}\boldsymbol{z}^\top\tilde{\boldsymbol{B}}_{k+1}\boldsymbol{z} &= \boldsymbol{z}^\top\boldsymbol{L}\boldsymbol{L}^\top\boldsymbol{z} \\ &= \frac{(\boldsymbol{z}^\top\boldsymbol{L}\boldsymbol{L}^\top\boldsymbol{z})(\tilde{\boldsymbol{s}}_k^\top\boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k)}{\tilde{\boldsymbol{s}}_k^\top\boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k} - \frac{(\boldsymbol{z}^\top\boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k)^2}{\tilde{\boldsymbol{s}}_k^\top\boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k} + \frac{(\boldsymbol{z}^\top\boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k)^2}{\tilde{\boldsymbol{s}}_k^\top\boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k}.\end{aligned}$$

The secant condition gives us $\tilde{\boldsymbol{y}}_k = \boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k$, hence

$$\boldsymbol{z}^\top\tilde{\boldsymbol{B}}_{k+1}\boldsymbol{z} = \frac{(\boldsymbol{z}^\top\boldsymbol{L}\boldsymbol{L}^\top\boldsymbol{z})(\tilde{\boldsymbol{s}}_k^\top\boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k) - (\boldsymbol{z}^\top\boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k)^2}{\tilde{\boldsymbol{s}}_k^\top\boldsymbol{L}\boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k} + \frac{(\boldsymbol{z}^\top\tilde{\boldsymbol{y}}_k)^2}{\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{y}}_k}. \tag{3.14}$$

For some arbitrary vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, it follows from the Cauchy-Schwarz inequality that

$$0 \leq (\boldsymbol{u}^\top\boldsymbol{u})(\boldsymbol{v}^\top\boldsymbol{v}) - (\boldsymbol{u}^\top\boldsymbol{v})^2.$$

Let $\boldsymbol{u} = \boldsymbol{L}^\top\boldsymbol{z} \neq \boldsymbol{0}$ and $\boldsymbol{v} = \boldsymbol{L}^\top\tilde{\boldsymbol{s}}_k \neq \boldsymbol{0}$, and from (3.14) follows

$$\boldsymbol{z}^\top\tilde{\boldsymbol{B}}_{k+1}\boldsymbol{z} \geq \frac{(\boldsymbol{z}^\top\tilde{\boldsymbol{y}}_k)^2}{\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{y}}_k}.$$

Since the curvature condition ensures that $\tilde{\boldsymbol{s}}_k^\top\tilde{\boldsymbol{y}}_k > 0$, it follows that

$$\boldsymbol{z}^\top\tilde{\boldsymbol{B}}_{k+1}\boldsymbol{z} \geq 0.$$

To show that it is a strict inequality, and thus positive definiteness of $\tilde{\boldsymbol{B}}_{k+1}$, suppose for contradiction that the numerators in (3.14) can satisfy

$$(\boldsymbol{z}^\top \boldsymbol{L}\boldsymbol{L}^\top \boldsymbol{z})(\tilde{\boldsymbol{s}}_k^\top \boldsymbol{L}\boldsymbol{L}^\top \tilde{\boldsymbol{s}}_k) - (\boldsymbol{z}^\top \boldsymbol{L}\boldsymbol{L}^\top \tilde{\boldsymbol{s}}_k)^2 = 0$$

and

$$(\boldsymbol{z}^\top \tilde{\boldsymbol{y}}_k)^2 = 0.$$

This can hold if and only if $\boldsymbol{L}^\top \boldsymbol{z} = c\boldsymbol{L}^\top \tilde{\boldsymbol{s}}_k$, for some nonzero scalar c, and thus $\boldsymbol{z} = c\tilde{\boldsymbol{s}}_k$. As a consequence, the second numerator becomes

$$\begin{aligned} & (\boldsymbol{z}^\top \tilde{\boldsymbol{y}}_k)^2 = 0 \\ \iff\ & c^2(\tilde{\boldsymbol{s}}_k^\top \tilde{\boldsymbol{y}}_k)^2 = 0 \\ \implies\ & \tilde{\boldsymbol{s}}_k^\top \tilde{\boldsymbol{y}}_k = 0, \end{aligned}$$

but this contradicts Theorem 3.3 and thus (3.14) implies positive definiteness of $\tilde{\boldsymbol{B}}_{k+1}$. □

3.2 SR1 Method

So far we have paid attention to the BFGS based methods which update the Hessian approximation (or its inverse) of an objective function with a rank-two matrix. In this section we will explore a method which is based on a weaker rank-one update, the *symmetric rank-one* method (SR1) [5, 8]. The first drawback is that the SR1 update will not guarantee that the new updated $\boldsymbol{B}_{k+1}$ is positive definite, but only that it is symmetric. Although this may be considered a major disadvantage, the SR1 method can sometimes in practice, under mild conditions, give better convergence towards the true Hessian than the BFGS based algorithms [5, 8].

A general symmetric rank-one update is given by

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + a\boldsymbol{u}\boldsymbol{u}^\top, \tag{3.15}$$

where a and $\boldsymbol{u}$ are chosen such that $a = \pm 1$ and the secant equation (3.2) holds, that is,

$$\begin{aligned} & \boldsymbol{y}_k = (\boldsymbol{B}_k + a\boldsymbol{u}\boldsymbol{u}^\top)\boldsymbol{s}_k \\ \iff\ & (\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k) = (a\boldsymbol{u}^\top \boldsymbol{s}_k)\boldsymbol{u}, \end{aligned} \tag{3.16}$$

where $(\boldsymbol{s_k}, \boldsymbol{y}_k)$ are curvature pairs [8]. Since $(a\boldsymbol{u}^\top \boldsymbol{s}_k)$ is a scalar, it follows from (3.16) that $\boldsymbol{u}$ is a multiple of $(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k)$, thus for some scalar c

$$\boldsymbol{u} = c(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k).$$

We can now use this form of $\boldsymbol{u}$ to transform (3.16) into

$$(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k) = ac^2(\boldsymbol{s}_k^\top(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k))(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k),$$

hence a and c are given by

$$a = \operatorname{sign}(\boldsymbol{s}_k^\top(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k)), \quad c = \pm(\boldsymbol{s}_k^\top(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k))^{-1/2}.$$

The SR1 update scheme for the Hessian approximation is thus given by

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k)(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k)^\top}{(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k)^\top \boldsymbol{s}_k}. \tag{3.17}$$

The way we update $\boldsymbol{x}_k$ is this time not via line search, but via trust region. The SR1 method computes a search direction $\boldsymbol{p}_k$ by minimizing the constrained subproblem (2.10), where $\boldsymbol{B}_k$ in the quadratic model m_k (2.8) is given by (3.17). The current iterate is then updated according to

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}_k \tag{3.18}$$

if for some predetermined constant $\eta > 0$ we have $\rho_k \geq \eta$. If $\rho_k < \eta$, then the iterate is left unchanged. After each iteration, ρ_k gets moderated according to the list following its definition in (2.9).

The following theorem shows that $\boldsymbol{x}_k$ converges to a minimizer $\boldsymbol{x}^*$ superlinearly when updated by the SR1 scheme (3.18), under a fair number of assumptions.

Theorem 3.5. *Suppose that the sequence of iterates $\{\boldsymbol{x}_k\}$ generated by the SR1 scheme (3.18) does not terminate and remains in a compact convex set in which $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and has a unique stationary point $\boldsymbol{x}^*$. Suppose that $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite and $\nabla^2 f(\boldsymbol{x})$ is Lipschitz continuous in a neighborhood of $\boldsymbol{x}^*$. Lastly, suppose that the sequence of Hessian approximations $\{\boldsymbol{B}_k\}$ generated by (3.17) is bounded in norm and that*

$$|(\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k)^\top \boldsymbol{s}_k| \geq \epsilon \|\boldsymbol{s}_k\| \|\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k\| \tag{3.19}$$

holds for all k, where $\epsilon \in (0, 1)$. Then $\boldsymbol{x}_k \to \boldsymbol{x}^*$ at an $(n+1)$-step superlinear rate, that is,

$$\lim_{k\to\infty} \frac{\|\boldsymbol{x}_{k+n+1} - \boldsymbol{x}^*\|}{\boldsymbol{x}_k - \boldsymbol{x}^*} = 0.$$

Proof. (See [8].) □

3.3 Limited-memory BFGS and SR1 Methods

While working on large-scale problems such as considerably deep neural network training, the standard BFGS scheme described in (3.8) may become computationally inefficient. The reason for this lies in the cost of storing and/or manipulating dense inverse Hessian approximations $\boldsymbol{H}_k$. One way around this practical issue is to store a fixed number of curvature pairs $\{(\boldsymbol{s}_i, \boldsymbol{y}_i)\}$ corresponding to the building process of $\boldsymbol{H}_k$ according to (3.8), which gives a slightly less accurate version of it. The curvature pairs $\{(\boldsymbol{s}_i, \boldsymbol{y}_i)\}$ are then used to build $\boldsymbol{H}_k$, hence $\boldsymbol{H}_k \nabla f(\boldsymbol{x}_k)$ can be computed without the need to store $\boldsymbol{H}_k$ [1, 5, 8, 9].

Suppose we are at iteration k with m curvature pairs $\{(\boldsymbol{s}_i, \boldsymbol{y}_i)\}$, where $i = k - m, \ldots, k - 1$. The computation of $\boldsymbol{H}_k$ builds upon some initial value $\tilde{\boldsymbol{H}}_0$,

which when put into the BFGS scheme (3.8) forms the *limited-memory* BFGS (L-BFGS) scheme

$$\begin{aligned}\boldsymbol{H}_k = &\left(\boldsymbol{V}_{k-1}^\top \cdots \boldsymbol{V}_{k-m}^\top\right) \tilde{\boldsymbol{H}}_0 \left(\boldsymbol{V}_{k-m} \cdots \boldsymbol{V}_{k-1}\right) \\ &+ \rho_{k-m} \left(\boldsymbol{V}_{k-1}^\top \cdots \boldsymbol{V}_{k-m}^\top\right) \boldsymbol{s}_{k-m}\boldsymbol{s}_{k-m}^\top \left(\boldsymbol{V}_{k-m} \cdots \boldsymbol{V}_{k-1}\right) \\ &+ \rho_{k-m+1} \left(\boldsymbol{V}_{k-1}^\top \cdots \boldsymbol{V}_{k-m}^\top\right) \boldsymbol{s}_{k-m+1}\boldsymbol{s}_{k-m+1}^\top \left(\boldsymbol{V}_{k-m} \cdots \boldsymbol{V}_{k-1}\right) \\ &+ \dots \\ &+ \rho_{k-1}\boldsymbol{s}_{k-1}\boldsymbol{s}_{k-1}^\top,\end{aligned} \tag{3.20}$$

from which $\boldsymbol{H}_k \nabla f(\boldsymbol{x}_k)$ can be computed recursively [8, 9]. When $\boldsymbol{H}_k \nabla f(\boldsymbol{x}_k)$ is computed, the next iterate is obtained according to line search applied on (3.1), that is,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{H}_k \nabla f(\boldsymbol{x}_k),$$

and the set of curvature pairs $\{(\boldsymbol{s}_i, \boldsymbol{y}_i)\}$ gets updated to the new most recent m pairs. The procedure is repeated until convergence and defines the L-BFGS method. Notice that the initial value for $\boldsymbol{x}_0$ should also be provided and that every $\boldsymbol{H}_k$ can have a different initial $\tilde{\boldsymbol{H}}_0$ in (3.20).

Following the same idea as for the L-BFGS scheme (3.20), the SR1 method can also be modified to a limited-memory variant. In that case, m curvature pairs $\{(\boldsymbol{s}_i, \boldsymbol{y}_i)\}$ will get stored and used to calculate (3.17) recursively in each iteration, analogously to (3.20). The corresponding L-SR1 formula for $\boldsymbol{H}_k$ will this time get more complicated compact form (see e.g. [9]).

3.3.1 Sampled Limited-memory BFGS and SR1 Methods

While trying to make the BFGS and SR1 methods more practical, we have seen how L-BFGS and L-SR1 methods can be used as a way around the computational cost and manipulation of dense inverse Hessian approximations. Another technique which builds upon the limited-memory schemes is *sampled* L-BFGS and L-SR1 (SL-BFGS and SL-SR1) proposed in [1]. The main difference in this new approach is the way in which curvature pairs $\{(\boldsymbol{s}_i, \boldsymbol{y}_i)\}$ are constructed and used. Here, the past curvature information is not used in the construction of new pairs, but is obtained via *sampling*. In some cases, this has proven to capture the curvature information of the objective function f better than the regular BFGS, SR1, L-BFGS and L-SR1 methods, especially while training deep networks [1].

Given a current iterate $\boldsymbol{x}_k$, a number of curvature pairs m and a sampling radius r, the SL-BFGS and SL-SR1 methods construct curvature pairs $\{(\boldsymbol{s}_i, \boldsymbol{y}_i)\}$ by sampling a random direction $\boldsymbol{\sigma}_i$, setting $\tilde{\boldsymbol{x}}_k = \boldsymbol{x}_k + r\boldsymbol{\sigma}_i$ and then evaluating

$$\boldsymbol{s}_i = \boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k, \quad \boldsymbol{y}_i = \begin{cases} \nabla f(\boldsymbol{x}_k) - \nabla f(\tilde{\boldsymbol{x}}_k), & \text{Option I} \\ \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_i, & \text{Option II} \end{cases} \tag{3.21}$$

for every $i = 1, \dots, m$ [1]. Both Option I & II are viable and should be chosen with regards to the nature of a given problem and computational resources.

The SL-BFGS method is carried out according to the limited-memory scheme (3.20), where the curvature pairs are constructed via (3.21), leading to the line search

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{H}_k \nabla f(\boldsymbol{x}_k), \tag{3.22}$$

given initial $\boldsymbol{x}_0$, sampling radius r and memory m.

Analogously to SL-BFGS and L-SR1, the SL-SR1 method is derived by solving the trust region subproblem (2.10) to obtain a search direction $\boldsymbol{p}_k$, where ρ_k is used to control the trust radius Δ_k according to the list below its definition in (2.11), and the Hessian approximation in the quadratic model (2.8) is computed recursively via sampling (3.21). Hence,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}_k \tag{3.23}$$

defines the SL-SR1 scheme given some initial $\boldsymbol{x}_0$, initial trust radius Δ_0, sampling radius r and memory m. The current iterate $\boldsymbol{x}_k$ is left unchanged if for some predetermined constant η we have $\rho_k < \eta$.

We will now take a look at the convergence of these newly proposed methods, as presented and proved by their authors in [1]. Starting with the SL-BFGS, we will take a look at the convergence of strongly convex objective functions f.

Lemma 3.6. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and assume there exist constants $0 < m \leq L$ such that $\nabla^2 f(\boldsymbol{x}) - m\boldsymbol{I}$ and $L\boldsymbol{I} - \nabla^2 f(\boldsymbol{x})$ are positive semi-definite for all $\boldsymbol{x} \in \mathbb{R}^n$. Then there exist some constants $0 < \mu_1 \leq \mu_2$ such that for the inverse Hessian approximations $\boldsymbol{H}_k$ generated within the SL-BFGS scheme (3.22), $\boldsymbol{H}_k - \mu_1 \boldsymbol{I}$ and $\mu_2 \boldsymbol{I} - \boldsymbol{H}_k$ are positive semi-definite for all $k \geq 0$.*

Theorem 3.7. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and assume there exist constants $0 < m \leq L$ such that $\nabla^2 f(\boldsymbol{x}) - m\boldsymbol{I}$ and $L\boldsymbol{I} - \nabla^2 f(\boldsymbol{x})$ are positive semi-definite for all $\boldsymbol{x} \in \mathbb{R}^n$. Let $\boldsymbol{x}^*$ be a minimizer of f, set $f^* \coloneqq f(\boldsymbol{x}^*)$ and let $\{\boldsymbol{x}_k\}$ be generated by the SL-BFGS scheme (3.22), where we let*

$$0 < \alpha_k = \alpha \leq \frac{\mu_1}{\mu_2^2 L}. \tag{3.24}$$

Then for all $k \geq 0$, it follows that

$$f(\boldsymbol{x}_k) - f^* \leq (1 - \alpha m \mu_1)^k (f(\boldsymbol{x}_0) - f^*). \tag{3.25}$$

Proof. Let $\boldsymbol{p}_k = -\alpha \boldsymbol{H}_k \nabla f(\boldsymbol{x}_k)$ and we have

$$f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{x}_k + \boldsymbol{p}_k). \tag{3.26}$$

Taylor expanding the right-hand side of (3.26) gives

$$f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top \boldsymbol{p}_k + \frac{1}{2} \boldsymbol{p}_k^\top \nabla^2 f(\boldsymbol{x} + t\boldsymbol{p}_k) \boldsymbol{p}_k, \tag{3.27}$$

where $t \in (0,1)$. Since we have assumed that $L\mathbf{I} - \nabla^2 f(\boldsymbol{x})$ is positive semi-definite for all $\boldsymbol{x} \in \mathbb{R}^n$, it follows that

$$\begin{aligned} & \boldsymbol{p}_k^\top (L\mathbf{I} - \nabla^2 f(\boldsymbol{x}_k + t\boldsymbol{p}_k))\boldsymbol{p}_k \geq 0 \\ \iff & \boldsymbol{p}_k^\top L\mathbf{I}\boldsymbol{p}_k - \boldsymbol{p}_k^\top \nabla^2 f(\boldsymbol{x}_k + t\boldsymbol{p}_k)\boldsymbol{p}_k^\top \geq 0 \\ \iff & \boldsymbol{p}_k^\top \nabla^2 f(\boldsymbol{x}_k + t\boldsymbol{p}_k)\boldsymbol{p}_k^\top \leq L\|\boldsymbol{p}_k\|^2, \end{aligned}$$

hence for (3.27) we get

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k + \frac{L}{2}\|\boldsymbol{p}_k\|^2 \\ &= f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top (-\alpha \boldsymbol{H}_k \nabla f(\boldsymbol{x}_k)) + \frac{L}{2}\|\alpha \boldsymbol{H}_k \nabla f(\boldsymbol{x}_k)\|^2 \\ &= f(\boldsymbol{x}_k) - \alpha \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{H}_k \nabla f(\boldsymbol{x}_k) + \frac{\alpha^2 L}{2}\|\boldsymbol{H}_k \nabla f(\boldsymbol{x}_k)\|^2. \end{aligned}$$

Using Lemma 3.6 gives similarly

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) - \alpha \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{H}_k \nabla f(\boldsymbol{x}_k) + \frac{\alpha^2 L}{2}\|\boldsymbol{H}_k \nabla f(\boldsymbol{x}_k)\|^2 \\ &\leq f(\boldsymbol{x}_k) - \alpha\mu_1 \|\nabla f(\boldsymbol{x}_k)\|^2 + \frac{\alpha^2 \mu_2^2 L}{2}\|\nabla f(\boldsymbol{x}_k)\|^2 \\ &= f(\boldsymbol{x}_k) + \alpha\left(\frac{\alpha\mu_2^2 L}{2} - \mu_1\right)\|\nabla f(\boldsymbol{x}_k)\|^2, \end{aligned}$$

and since for α we have (3.24), it follows that

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) + \alpha\left(\frac{\alpha\mu_2^2 L}{2} - \mu_1\right)\|\nabla f(\boldsymbol{x}_k)\|^2 \\ &\leq f(\boldsymbol{x}_k) + \alpha\left(\frac{\mu_1\mu_2^2 L}{2\mu_2^2 L} - \mu_1\right)\|\nabla f(\boldsymbol{x}_k)\|^2 \\ &= f(\boldsymbol{x}_k) - \alpha\frac{\mu_1}{2}\|\nabla f(\boldsymbol{x}_k)\|^2. \end{aligned} \tag{3.28}$$

Since f is twice continuously differentiable and $\nabla^2 f(\boldsymbol{x}) - m\boldsymbol{I}$ is positive semi-definite for all $\boldsymbol{x}$, it follows that f is strongly convex [6], i.e.

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{x} - \boldsymbol{y}) + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 \tag{3.29}$$

for some $\mu > 0$ and all $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f. It follows as a consequence of (3.29) that

$$2\mu(f(\boldsymbol{y}) - f(\boldsymbol{x})) \leq 2\mu \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|^2 \tag{3.30}$$

for some $\mu > 0$ and all $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f [6]. Now, by setting $\boldsymbol{x} = \boldsymbol{x}^*$ in (3.30), we get

$$2\mu(f(\boldsymbol{y}) - f^*) \leq \|\nabla f(\boldsymbol{y})\|^2,$$

hence it follows from (3.28) that

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) - \alpha\frac{\mu_1}{2}\|\nabla f(\boldsymbol{x}_k)\|^2 \\ &\leq f(\boldsymbol{x}_k) - \alpha\mu\mu_1(f(\boldsymbol{x}_k) - f^*) \\ \iff f(\boldsymbol{x}_{k+1}) - f^* &\leq (1 - \alpha\mu\mu_1)(f(\boldsymbol{x}_k) - f^*). \end{aligned} \tag{3.31}$$

From (3.31) follows linear convergence, and recursive application of it gives (3.25). □

On the other hand, if the objective function f is non-convex, a slightly different approach is needed because the standard L-BFGS method can diverge. In this case, a *cautious approach* is used, that is, only those curvature pairs $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ that satisfy

$$\boldsymbol{s}_k^\top \boldsymbol{y}_k \geq \epsilon\|\boldsymbol{s}_k\|^2 \tag{3.32}$$

for some predetermined $\epsilon > 0$ are used in the computation of inverse Hessian approximations [1].

Lemma 3.8. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and $\nabla f(\boldsymbol{x})$ is Lipschitz continuous for all $\boldsymbol{x} \in \mathbb{R}^n$. Let $\{\boldsymbol{H}_k\}$ be the inverse Hessian approximations generated within the SL-BFGS scheme (3.22) using only $(\boldsymbol{s}_k, \boldsymbol{y}_k)$ that satisfy (3.32), and let $\boldsymbol{H}_k = \boldsymbol{I}$ if no curvature pairs satisfy it. Then there exist some constants $0 < \mu_1 \leq \mu_2$ such that $\boldsymbol{H}_k - \mu_1\boldsymbol{I}$ and $\mu_2\boldsymbol{I} - \boldsymbol{H}_k$ are positive semi-definite for all $k \geq 0$.*

Theorem 3.9. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and bounded below by some scalar B, and $\nabla f(\boldsymbol{x})$ is Lipschitz continuous for all $\boldsymbol{x} \in \mathbb{R}^n$. Let $\{\boldsymbol{x}_k\}$ be a sequence of iterates generated by the SL-BFGS scheme (3.22), where*

$$\boldsymbol{s}_k^\top \boldsymbol{y}_k \geq \epsilon\|\boldsymbol{s}_k\|^2 \tag{3.33}$$

is imposed on the curvature pairs $(\boldsymbol{s}_k, \boldsymbol{y}_k)$ for some predecided constant $\epsilon > 0$, and $\boldsymbol{H}_k = \boldsymbol{I}$ is set if no curvature pairs satisfy (3.32). Let also $\boldsymbol{x}_0$ be the initial value and impose

$$0 < \alpha_k = \alpha \leq \frac{\mu_1}{\mu_2^2 L} \tag{3.34}$$

on the step size α_k. Then

$$\lim_{k\to\infty} \|\nabla f(\boldsymbol{x}_k)\| \to 0, \tag{3.35}$$

and for every $N > 1$,

$$\lim_{N\to\infty} \frac{1}{N}\sum_{k=0}^{N-1} \|\nabla f(\boldsymbol{x}_k)\|^2 \leq \lim_{N\to\infty} \frac{2(f(\boldsymbol{x}_0) - B)}{\alpha\mu_1 N} \to 0. \tag{3.36}$$

Proof. We have from (3.28) that

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq -\frac{\alpha\mu_1}{2}\|\nabla f(\boldsymbol{x})\|^2.$$

Taking a sum on both sides for $k = 0, \ldots, N-1$ gives

$$\begin{aligned} -\sum_{k=0}^{N-1} \frac{\alpha\mu_1}{2} \|\nabla f(\boldsymbol{x}_k)\|^2 &\geq \sum_{k=0}^{N-1} (f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)) \\ &\geq f(\boldsymbol{x}_N) - f(\boldsymbol{x}_0) \\ &\geq \hat{f} - f(\boldsymbol{x}_0), \end{aligned}$$

where $\hat{f}$ is the assumed lower bound of f. Hence

$$\begin{aligned} &\sum_{k=0}^{N-1} \|\nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{2}{\alpha\mu_1}(\hat{f} - f(\boldsymbol{x}_0)) \\ \implies 0 < &\sum_{k=0}^{\infty} \|\nabla f(\boldsymbol{x}_k)\|^2 < \infty \end{aligned}$$

is obtained, which means that the sum is convergent. It follows that

$$\lim_{k\to\infty} \|\nabla f(\boldsymbol{x}_k)\| \to 0.$$

If follows as well from

$$\sum_{k=0}^{N-1} \|\nabla f(\boldsymbol{x}_k)\|^2 \leq \frac{2}{\alpha\mu_1}(\hat{f} - f(\boldsymbol{x}_0))$$

that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \|\nabla f(\boldsymbol{x}_k)\|^2 \leq \lim_{N\to\infty} \frac{1}{N} \frac{2}{\alpha\mu_1}(\hat{f} - f(\boldsymbol{x}_0)) = 0.$$

□

To establish convergence for SL-SR1, another *cautious approach* is used to make sure that the denominator in (3.17) is bounded away from zero [1]. This ensures that $\boldsymbol{B}_{k+1}$ is well-defined and we impose

$$|(\boldsymbol{y}_k - \boldsymbol{B}_k \boldsymbol{s}_k)^\top \boldsymbol{s}_k| \geq \epsilon \|\boldsymbol{s}_k\| \|\boldsymbol{y}_k - \boldsymbol{B}_k \boldsymbol{s}_k\|$$

for some predetermined constant $\epsilon > 0$.

Lemma 3.10. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and $\nabla f(\boldsymbol{x})$ is Lipschitz continuous for all $\boldsymbol{x} \in \mathbb{R}^n$. Assume also that there are constants $\xi \in (0, 1)$ and $\beta_k = 1 + \|\boldsymbol{B}_k\|$ such that for all k the inequality in (2.12) holds, where m_k is defined by (2.8) and Δ_k is a trust radius (2.9). Let $\{\boldsymbol{B}_k\}$ be a sequence of Hessian approximations generated within the SL-SR1 scheme (3.23) using only $(\boldsymbol{s}_k, \boldsymbol{y}_k)$ that satisfy (3.19), and let $\boldsymbol{B}_k = \boldsymbol{I}$ if no curvature pairs satisfy it. Then there exists a constant $\nu > 0$ such that $\nu \geq \|\boldsymbol{B}_k\|$ for all k.*

Theorem 3.11. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and bounded below by some scalar $\hat{f}$, $\nabla f(\boldsymbol{x})$ is Lipschitz continuous for all $\boldsymbol{x} \in \mathbb{R}^n$ and (2.12) holds for some $\xi \in (0, 1)$, $\beta_k = 1 + \|\boldsymbol{B}_k\|$ and all k. Let $\{\boldsymbol{x}_k\}$ be a sequence of iterates generated by the SL-SR1 scheme (3.23) using only $(\boldsymbol{s}_k, \boldsymbol{y}_k)$ for which the Hessians $\boldsymbol{B}_k$ satisfy (3.19) for some $\epsilon > 0$, and let $\boldsymbol{B}_k = \boldsymbol{I}$ if no curvature pairs satisfy it. Then*

$$\lim_{k \to \infty} \|\nabla f(\boldsymbol{x}_k)\| = 0.$$

Proof. Recall that in (3.23) some iterates $\boldsymbol{x}_k$ will be left unchanged if the corresponding trust radius $\rho_k < \eta$ for some predetermined η. Here, we will focus on those iterates for which $\rho_k \geq \eta$. Suppose for contradiction that there is a subsequence of accepted iterates $\{\boldsymbol{x}_{t_i}\}$, with the indices $t_i \subseteq \mathcal{S} = \{k \geq 0 : \rho_k > \eta\}$, such that for all i and some $\delta > 0$

$$0 < 2\delta \leq \|\nabla f(\boldsymbol{x}_{t_i})\|. \tag{3.37}$$

Then, as a consequence of a theorem mentioned in A.7 [1], there exist certain indices $l_i > t_i$ such that

$$0 < \|\nabla f(\boldsymbol{x}_{l_i})\| < \delta.$$

Hence, there exists another subsequence of iterates $\{\boldsymbol{x}_{l_i}\}$, with the indices $l_i \subseteq \mathcal{S}$, such that for all $t_i \leq k < l_i$ it follows that

$$\|\nabla f(\boldsymbol{x}_{l_i})\| < \delta \leq \|\nabla f(\boldsymbol{x}_k)\|. \tag{3.38}$$

Recall the definition of ρ_k given in (2.11) and it follows from the assumption (2.12) that for all indices in $\mathcal{S}$,

$$\begin{aligned}
\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})}{m_k(\boldsymbol{0}) - m_k(\boldsymbol{p}_k)} &\geq \eta \\
\iff f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) &\geq \eta(m_k(\boldsymbol{0}) - m_k(\boldsymbol{p}_k)) \\
&\geq \eta\zeta\|\nabla f(\boldsymbol{x}_k)\| \min\left\{\frac{\|\nabla f(\boldsymbol{x}_k)\|}{\beta_k}, \Delta_k\right\}.
\end{aligned}$$

If we now restrict ourselves to only those indices in $\mathcal{K} = \{k \in \mathcal{S} : t_i \leq k < l_i\} \subseteq \mathcal{S}$, it follows from (3.38) and Lemma 3.10 that

$$\begin{aligned}
f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) &\geq \eta\zeta\|\nabla f(\boldsymbol{x}_k)\| \min\left\{\frac{\|\nabla f(\boldsymbol{x}_k)\|}{\beta_k}, \Delta_k\right\} \\
&\geq \eta\zeta\delta \min\left\{\frac{\delta}{1+\nu}, \Delta_k\right\}.
\end{aligned}$$

Since we have assumed a lower bound $\hat{f}$ for f and because ρ_k guarantees that $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$, it follows that $\{f(\boldsymbol{x}_k)\}$ is monotonically decreasing and

bounded below, hence it converges. It follows that $f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \to 0$ as $k \to \infty$, hence $\Delta_k \to 0$ as $k \to \infty$. For a sufficiently large k, we get

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq \eta\zeta\delta \min\left\{\frac{\delta}{1+\nu}, \Delta_k\right\} = \eta\zeta\delta\Delta_k$$
$$\implies \Delta_k \leq \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})}{\eta\zeta\delta}.$$

For sufficiently large i, we get consequently

$$\|\boldsymbol{x}_{t_i} - \boldsymbol{x}_{l_i}\| \leq \sum_{\substack{j=t_i \\ j\in\mathcal{K}}}^{l_i-1} \|\boldsymbol{x}_j - \boldsymbol{x}_{j+1}\| \leq \sum_{\substack{j=t_i \\ j\in\mathcal{K}}}^{l_i-1} \Delta_j \leq \frac{f(\boldsymbol{x}_j) - f(\boldsymbol{x}_{j+1})}{\eta\zeta\delta}.$$

For similar reasons as before, we get that $f(\boldsymbol{x}_j) - f(\boldsymbol{x}_{j+1}) \to 0$ as $i \to \infty$, hence $\|\boldsymbol{x}_{t_i} - \boldsymbol{x}_{l_i}\| \to 0$ as $i \to \infty$. The assumed Lipschitz continuity of ∇f implies now

$$\|\nabla f(\boldsymbol{x}_{t_i}) - \nabla f(\boldsymbol{x}_{l_i})\| \to 0,$$

but this is a contradiction because (3.37) and (3.38) imply

$$\|\nabla f(\boldsymbol{x}_{t_i}) - \nabla f(\boldsymbol{x}_{l_i})\| \geq \delta,$$

thus no subsequence will satisfy (3.37). □

4 Numerical Experiments

To give some intuition about quasi-Newton methods' performance in practice, a small neural network with one hidden layer is trained on a personal computer with NVIDIA GeForce RTX 2060 6 GB, Intel Core i5-10400F and 8 GB RAM, using CUDA parallel computing and Python 3.8.

The network has single neurons in the input/output layers and 7 neurons in the hidden layer. Hidden neurons are activated with hyperbolic tangent. The goal is to approximate a highly nonlinear function suggested in *Example 1* in [7], i.e.,

$$f(x) = 1 + (x + 2x^2)\sin(-x^2), \quad |x| \leq 4.$$

The methods that will be covered are BFGS, NAQ, L-BFGS and SL-BFGS using various parameters. However, it should be mentioned that most of these methods are well suited and designed for large-scale problems involving deeper network structures. In the case of SL-BFGS, Option I (3.21) is used with $r = 0.1$. The step lengths are computed using backtracking line search involving the Armijo condition discussed in [8] with $c = 0.001$ and $\rho = 0.001$. The training set is generated by sampling 400 observations $\{x_i, f(x_i)\},\ i = 1, \ldots, 400$ from the interval $x_i \in [-4, 4]$ using a fixed increment 0.02, starting from -4 to 4, which are then shuffled. The testing set is generated by sampling 10,000 uniformly random values from the same interval. The error function is defined as (2.26) to which

normalized values from the training/testing sets are passed in the evaluation. The initial values for the network weights $\boldsymbol{w}_0$ are randomly generated with uniform values on the interval $(-0.5, 0.5)$ for each complete training, and the initial inverse Hessians $\boldsymbol{H}_0$ are set to identity for BFGS and NAQ, and (7.20) in [8] for L-BFGS and SL-BFGS. Each training involved at most 100 epochs over the training set after which the final network was run on the testing set. The results were obtained by averaging 3 independent runs for each method using different initial $\boldsymbol{w}_0$. The training was terminated if the gradient norm was sufficiently small, i.e., if $\|\nabla E(\boldsymbol{w})\| < 0.001$.

Table 1: Simulation results for BFGS-based methods

Method	Step length	μ	m	Training error Ave./Best/Worst	Time(s) Ave./Best/Worst	Epochs Ave.	Testing error Ave./Best/Worst
BFGS	Armijo	-	-	0.03/0.02/0.15	5.70/5.44/7.25	9	0.01/0.01/0.02
NAQ	Armijo	0.8	-	0.02/0.02/0.06	5.89/5.58/6.55	6	0.02/0.01/0.02
		0.85	-	0.11/0.01/0.62	5.67/5.38/7.13	16	0.02/0.01/0.02
		0.9	-	2.34/0.02/10.07	6.11/5.37/8.14	15	0.01/0.01/0.02
L-BFGS	Armijo	-	3	0.03/0.02/0.11	7.18/5.33/10.27	58	0.01/0.01/0.02
		-	5	0.03/0.01/0.13	7.09/5.44/9.81	79	0.01/0.01/0.02
SL-BFGS	Armijo	-	3	0.03/0.02/0.25	9.85/9.04/12.70	100	0.02/0.01/0.02
		-	5	0.17/0.06/0.22	15.46/13.17/19.06	100	0.05/0.05/0.07

The comparative results shown in Table 1 exhibit reasonable behavior when we take in consideration the computational and storage costs of respective methods. For reference, we can take a look at a table in [1] (where we have added NAQ as well), that is:

Table 2: Costs of BFGS-based methods per iteration

Method	Computational cost	Storage cost
BFGS/NAQ	$nd + d^2 + lnd$	d^2
L-BFGS	$nd + 4md + lnd$	$2md$
SL-BFGS	$nd + mnd + 4md + lnd$	-

In Table 2, n denotes the number of training samples, d number of weights (gradient dimension), m memory size for limited-memory methods and l number of line search iterations. Roughly speaking, the difference in costs between BFGS/NAQ and L-BFGS lies in the fact that BFGS/NAQ computes whole Hessians or their inverses $O(d^2)$, while L-BFGS only needs m most recent gradient evaluations $O(md)$. Hence, if d becomes very large (in deep networks usually thousands or millions of parameters), the cost per iteration for BFGS/NAQ will become much higher than L-BFGS since BFGS/NAQ grows as $O(d^2)$ in difference and L-BFGS as $O(md)$. On the other hand, if d is reasonably small as in our example presented in Table 1, L-BFGS would not necessarily gain advantage (same goes for SL-BFGS), hence we get very similar timestamps. However,

since L-BFGS does not compute whole Hessians, it needs more epochs to obtain precision, thus we see that the average number of epochs is larger for L-BFGS. When it comes to SL-BFGS, we must note that it does not store any curvature information but instead samples it in each iteration, hence there is no storage cost and the computational cost is slightly greater. The true advantage over L-BFGS is obtained when the method is applied on very deep networks where $m \ll n, d$ and the computational costs approach $O(nd)$ [1]. In these situations SL-BFGS wins because the storage is not as strained. We can see in Table 1 that SL-BFGS gets slightly slower timestamps and requires even more epochs to obtain precision because the curvature pairs are sampled. Lastly, we would like to compare BFGS and NAQ but that is not as easy using information from Tables 1 & 2. It can be shown that NAQ under certain circumstances, including proper μ parameter choice, obtains desired precision, i.e. $\|\nabla E(\boldsymbol{w})\| < \delta$ for some δ, in fewer epochs than BFGS (see Tables I & III in [7]).

5 Discussion

This paper explores some novel varieties of quasi-Newton methods which proved to be effective on large-scale problems involving neural network training [1, 7]. All of these methods come with different benefits and drawbacks which are not easy to generalize. In the case of NAQ, a proper study of parameter μ should be considered and tuned for desired problems. A possible further research could involve implementation of limited-memory NAQ. On the other hand, sampled limited-memory variants of BFGS and SR1 are modified in [1] to disregard past curvature pairs, which are usually stored in memory, and instead simply sample them. This technique proved to be effective when the cost of computing and storing those curvature pairs becomes more expensive. This technique could also be investigated for NAQ. It should also be mentioned that different choices of memory size m can affect these methods drastically. There is a brief study of this for L-BFGS mentioned in Table 7.1 in [8] which could be interesting to test on SL-BFGS for comparison.

References

[1] A. S. Berahas, M. Jahani, and M. Takáč, *Quasi-newton methods for deep learning: Forget the past, just sample*, arXiv preprint arXiv:1901.09997, (2019).

[2] C. M. Bishop, *Pattern recognition and machine learning*, springer, 2006.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016. http://www.deeplearningbook.org.

[4] C. T. Kelley, *Iterative methods for linear and nonlinear equations*, SIAM, 1995.

[5] ——, *Iterative methods for optimization*, SIAM, 1999.

[6] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2003.

[7] H. Ninomiya, *A novel quasi-newton-based optimization for neural network training incorporating nesterov's accelerated gradient*, Nonlinear Theory and Its Applications, IEICE, 8 (2017), pp. 289–301.

[8] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[9] R. B. Schnabel, J. Nocedal, and R. H. Byrd, *Representations of quasi-newton matrices and their use in limited memory methods; cu-cs-612-92*, (1992).