Titre: Title:	Stochastic First and Second Order Optimization Methods for Machine Learning
Auteur: Author:	Sanae Lotfi
Date:	2020
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lotfi, S. (2020). Stochastic First and Second Order Optimization Methods for Machine Learning [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/5457/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/5457/
Directeurs de recherche: Advisors:	Andrea Lodi, & Dominique Orban
Programme: Program:	Maîtrise recherche en mathématiques appliquées

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Stochastic First and Second Order Optimization Methods for Machine Learning

SANAE LOTFI
Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques appliquées

Août 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Stochastic First and Second Order Optimization Methods for Machine Learning

présenté par **Sanae LOTFI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Charles AUDET, président
Andrea LODI, membre et directeur de recherche
Dominique ORBAN, membre et codirecteur de recherche
Ioannis MITLIAGKAS, membre

DEDICATION

To Professor Said Farouj,
thank you for believing in me.

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my advisor, Professor Andrea Lodi, for guiding me, trusting me and giving me the freedom to work on my topics of interest without any limitations. I will always be grateful to him for giving me the opportunity to explore research in the most authentic way. I would like to thank my co-advisor, Professor Dominique Orban, for his guidance, patience and presence all along the process. I must also thank my research partner, Tiphaine Bonniot, who dedicated time to explain her work to me and gave me the opportunity to build upon it.

I would like to acknowledge and thank Professor Ioannis Mitliagkas for his feedback and help, and for inspiring my interest in optimization for deep learning. My thanks also go to the Montreal Machine Learning and Optimization group for providing early feedback on my work, especially Professor Courtney Paquette and Doctor Nicolas Loizou. I would like to thank Professor Charles Audet as well for accepting to be a part of my committee and for teaching me the fundamentals of continuous optimisation with a lot of passion. I also thank Professor Jonathan Jalbert for his unwavering support and great help, as well as Professor Loubna Benabbou for her valuable guidance and constant encouragement.

I was lucky to be surrounded by a great and supportive group in the lab as well. Didier, Aleksandr, Sriram and Elias, thank you very much for our long discussions and for being great friends and mentors to me. Khalid, I am very thankful for your help and support, and I cherish all the discussions that we had. Federico, thank you for the great philosophical discussions about meditation and life. Mehdi and Mariia, thank you for your support and for helping me out with all the administrative paperwork. To everyone else in the lab: thank you for the nice time and coffee breaks that we shared.

I would like to thank my friends in Montreal who made my time in this city full of great moments of joy and laughter, and who supported me when doubt knocked at my door. Thank you for your encouragement, support, and love. I am also grateful to my friends oversees who had to manage the time difference with me, especially Salma, Mariem and Hind.

It goes without saying that I am extremely grateful to my family: my parents, my sister and my brothers for their unconditional love.

Finally, I would like to thank Professor Said Farouj, Professor Mustapha Zemzami, Professor Asmahan El-Khattabi, and every single professor and mentor who played an instrumental role in defining my career path and/or helped me build my own vision.

RÉSUMÉ

L'objectif principal de ce travail est de proposer des méthodes d'optimisation du premier et deuxième ordre pour le contexte spécifique de l'apprentissage automatique et l'optimisation à grande échelle. Notre point de départ est le travail de recherche de Tiphaine Bonniot de Ruisselet, avec Prof. Dominique Orban, à l'intersection de l'optimisation déterministe et stochastique sans contraintes.

Pour la première partie de notre travail, nous exploitons les méthodes de région de confiance afin de pouvoir définir un pas d'optimisation adaptatif. Nous utilisons aussi une méthode d'échantillonage adaptatif afin de réduire la variance des estimations du gradient au cours de l'optimisation. Plus précisément, nous utilisons un test stochastique afin de marquer la transition entre les deux phases d'optimisation stochastique suivantes :

- La première phase est une phase de convergence vers une région d'intérêt, où se trouve potentiellement un optimum local ou global. Nous souhaitons que cette phase soit la plus courte possible, mais aussi qu'elle mène à une région de convergence prometteuse. Donc, durant cette phase, nous utilisons un pas d'optimisation adaptatif en nous basant sur une méthode stochastique de région de confiance, tandis que la taille d'échantillonage reste fixe.
- La deuxième phase est une phase de convergence vers un optimum local ou global dans la région d'intérêt. Durant cette phase, nous choisissons de réduire la variance des estimations du gradient pour assurer une convergence rapide, mais nous utilisons un pas d'optimisation strictement décroissant pour assurer la convergence globale.

Ainsi, nous arrivons non seulement à définir une démarche de passage du cas déterministe au cas stochastique pour ce type de méthodes, mais nous réussissons également à proposer un nouvel algorithme adaptatif dont les performances sont prometteuses.

Quant aux méthodes de deuxième ordre, nous proposons une variante de l'algorithme BFGS stochastique tronqué à mémoire limitée. Nous développons une heuristique pour contrôler les valeurs propres de l'estimation de la matrice hessienne. Cela permet d'avoir une meilleure performance pour les problèmes où cette approximation devient mal-conditionnée au cours de l'optimisation. Nous proposons également de combiner cet algorithme avec une méthode de réduction de variance afin d'améliorer la qualité de l'estimation de courbature obtenue.

Nous prouvons que notre algorithme converge vers un point stationnaire. De plus, nous montrons empiriquement que le contrôle des bornes sur les valeurs propres de l'approximation de la matrice hessienne rend l'algorithme plus stable et robuste face aux problèmes mal-conditionnés.

ABSTRACT

In this work, we explore promising research directions to improve upon existing first- and second-order optimization algorithms in both deterministic and stochastic settings. Our starting point is the work developed by Tiphaine Bonniot de Ruisselet and Prof. Dominique Orban at the intersection of unconstrained deterministic and stochastic optimization.

As for first-order optimization methods, we first define a framework to transform a semi-deterministic trust-region method, that takes exact function values and inexact gradient estimates, to a fully stochastic optimization algorithm that is adapted to the context of machine learning. Then, we go a step further to propose a novel first-order optimization algorithm that exploits the two-phase nature of stochastic optimization.

The first phase is a global convergence phase. We would like to speed up the algorithm during this phase and converge to a region of interest that contains a good local or global optimum, while allowing some noise in the estimates which helps to explore the optimization landscape. Therefore, we propose to use a trust-region method during this phase to define the step size adaptively, allowing it to grow and shrink as needed. In the second phase, the objective is to converge to a specific optimum in the region of interest identified in the first phase. During this phase, we need to reduce the variance of the updates in order to improve their quality and converge fast. Hence, we propose to use adaptive sampling with a strictly decreasing step size in order to guarantee the overall convergence.

Our work enables us to define a generic framework for adapting trust-region methods to the stochastic setting. We also demonstrate that our algorithm is competitive for machine learning problems.

For second-order methods, we propose a new version of a stochastic damped L-BFGS method that is more robust on ill-conditioned problems. First, we prove that the eigenvalues of the Hessian approximation are bounded, using a less restrictive assumption compared to related works. Then we harness the potential of using these bounds to control the quality of the Hessian approximation.

We prove that our algorithm has appealing theoretical properties, as it converges almost surely to a stationary point. We also demonstrate that it is more robust than a previously proposed stochastic damped L-BFGS algorithm in the highly non-convex case that characterizes problems in deep learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ADAGRAD	Adaptive Gradient Algorithm
ADAM	Adaptive Moment estimation algorithm
AMSGRAD	Exponential moving average variant of ADAM proposed by (Reddi et al., 2018)
ARAS	Adaptive Regularization and Adaptive Sampling method using Pflug diagnostic
ARIG	Adaptive Regularization with Inexact Gradients
ARIG+	ARIG combined with the adaptive sampling strategy in (Byrd et al., 2012)
ARIGAF	Adaptive Regularization with Inexact GrAdient and Function values
BFGS	Broyden–Fletcher–Goldfarb–Shanno algorithm
CPU	Central Processing Unit
GPU	Graphics Processing Unit
L-BFGS	Limited-Memory Broyden–Fletcher–Goldfarb–Shanno algorithm
oBFGS	Stochastic Online BFGS algorithm
oLBFGS	Online Limited storage BFGS algorithm
RES	REgularized Stochastic BFGS algorithm
RCV1	Reuters Corpus Volume I
RMSProp	Root Mean Square Propagation
SA	Stochastic Approximation methods
SAA	Sample Average Approximation methods
SAG	Stochastic Average Gradient algorithm
SAGA	Improved version of SAG and SVRG proposed by (Defazio et al., 2014)
SARAH	StochAstic Recursive grAdient algoritHm
SARAH+	Practical variant of SARAH with an automatic and adaptive choice of the size of the inner loop size in SARAH
SDCA	Stochastic Dual Coordinate Ascent algorithm
SdLBFGS	Stochastic Damped L-BFGS
SdLBFGS-VR	Variance-Reduced Stochastic Damped L-BFGS
SdLBFGS-CHN	Stochastic Damped L-BFGS with Controlled Hessian Norm algorithm
Sd-REG-LBFGS	Stochastic Damped and REGularized L-BFGS algorithm
SFO	Stochastic First-order Oracle
SGD	Stochastic Gradient Descent

SGD-QN	Quasi-Newton Stochastic Gradient-Descent algorithm
SPIDER	Stochastic Path-Integrated Differential EstimatoR
STORM	STochastic Optimization with Random Models
SVMs	Support Vector Machines
SVRG	Stochastic Variance Reduced Gradient
SWATS	SWitches from ADAM To SGD algorithm
TRish	Trust-Region-ish algorithm
VR-SdLBFGS-CHN	Variance-Reduced Stochastic Damped L-BFGS with Controlled Hessian Norm algorithm

CHAPTER 1 INTRODUCTION

Mathematical optimization contributes in a crucial way to the increasing success of machine learning-based intelligent systems. It is used to compute the optimal parameters such that those intelligent systems are able to make good decisions on unseen data. Over the past few years, considerable interests in designing powerful large-scale optimization methods has emerged, mainly to make use of access to enormous datasets and powerful computational capacities and accelerate progress in machine learning.

First-order optimization methods have gained a lot of popularity in optimization for machine learning, as they have low per-iteration cost and can converge at linear or sublinear rates, depending on the problem. On the other hand, second-order methods can be more computationally expensive per-iteration, but they enjoy local superlinear or quadratic convergence rates. Moreover, they can be more suitable for ill-conditioned and highly non-linear problems.

Therefore, we propose in this thesis different algorithms that leverage the strengths of first- and second-order methods. In the following sections, we will set the theoretical framework of our work as well as the kind of problems that we are considering.

1.1 Framework and definitions

We consider general unconstrained stochastic optimization problems of the following form:

$$\min_{x \in \mathbb{R}^n} f(x) := \mathbb{E}_\xi[F(x, \xi)], \tag{1.1}$$

where $\xi \in \mathbb{R}^d$ denotes a random variable and $F : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable function, possibly non-convex. We introduce additional assumptions about F as needed. We refer to this problem as an *online* optimization problem, since we typically sample data points during the optimization process to have *new* samples in each iteration, in contrast with having a fixed dataset that is known up-front. We refer to f as the *expected risk* or the *expected loss*.

In the context of machine learning, a special case of the optimization problem (1.1) is *empirical risk minimization*, which consists of minimizing a different form of f,

$$\min_{x \in \mathbb{R}^n} f(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x), \tag{1.2}$$

where f_i represents the loss function that corresponds to the i-th element of our dataset and N is the number of samples that we have, i.e., the size of the dataset. We refer to this problem as a *finite-sum* problem and to f as the *empirical risk* or the *empirical loss.*

We consider the situation where it is not possible to evaluate the exact values of the gradient $\nabla f(x)$; rather, we have access to an approximation of this gradient denoted $g(x_k, \xi_k)$. If we consider problem (1.2), then a popular choice for $g(x_k, \xi_k)$ is what we call the *mini-batch* gradient

$$g(x_k, \xi_k) = \frac{1}{m_k} \sum_{i=1}^{m_k} \nabla f_{\xi_{k,i}}(x_k), \tag{1.3}$$

where ξ_k corresponds to the subset of samples considered in iteration k, from a given set of realizations of the random variable ξ, $\xi_{k,i}$ is the i-th sample of ξ_k in iteration k and m_k is the batch size used in iteration k, i.e., the number of samples we use in iteration k to evaluate the gradient approximation. It follows that $f_{\xi_{k,i}}$ represents the loss function that corresponds to the sample $\xi_{k,i}$.

For both methods that we propose in this thesis, the update rule of parameters $x \in \mathbb{R}^n$ takes the following form:

$$x_{k+1} = x_k - \alpha_k d_k, \tag{1.4}$$

where α_k is the step size in iteration k and d_k is the search direction. d_k generally takes the following form:

$$d_k = \begin{cases} g(x_k, \xi_k), & \text{for first order methods} \\ B_k^{-1} g(x_k, \xi_k), & \text{for quasi-newton methods} \end{cases} \tag{1.5}$$

where B_k is an approximation to the Hessian matrix $\nabla^2 f(x_k)$, and we note $H_k = B_k^{-1}$, such that H_k^{-1} approximates the Hessian matrix $\nabla^2 f(x_k)$.

A common practice in machine learning is to divide the dataset we have into a *training* dataset, that is used to train the chosen model, and a *test* dataset, that is used to evaluate the performance of the trained model on unseen data. The training and test datasets are usually divided into *batches*, which are subsets of these datasets. We use the term *epoch* to refer to an optimization cycle over the entire training dataset, which would practically mean that all batches were used during this cycle. Hence, the number of epochs refers to the number of passes over the entire training dataset.

We use the expression *semi-deterministic approach* to refer to an optimization algorithm that takes exact function values and inexact gradient values.

Notation We use $\|.\| := \|.\|_2$ to denote the Euclidean norm throughout this thesis. Other types of norms will be introduced by the notation $\|.\|_p$, where the value of p will be specified as needed. Capital Latin letters such as A, B, and H represent matrices, lowercase Latin letters such as s, x, and y represent vectors in $\mathbb{R}^n$, and lowercase Greek letters such as α, β and γ represent scalars. The operators $\mathbb{E}_\xi[.]$ and $\mathbb{V}_\xi[.]$ represent the expectation and variance over random variable ξ, respectively. We use the notation $A \succ B$ to imply that the matrix $A - B$ is positive definite, and $A \succeq B$ to imply that the matrix $A - B$ is positive semidefinite. We use $\lceil . \rceil$ to denote the ceiling function that maps a real number y to the least integer greater than or equal to y.

1.2 Problem characteristics

In this section, we discuss the characteristics of optimization problems that arise in large-scale optimization and machine learning. In Subsection 1.2.1, we justify the use of stochastic methods instead of full-gradient methods in large-scale optimization in general and machine learning in particular. This motivates our work in Chapter 3, which consists of adapting a semi-deterministic optimization algorithm to the fully stochastic context. Our discussion draws heavily from Section 3.3 in (Bottou et al., 2018). In Subsection 1.2.2, we motivate the design of optimization algorithms that allow an adaptive and automatic choice of the algorithm's hyperparameters. This discussion is inspired by (Curtis and Scheinberg, 2020), a paper that not only explored the motivation behind the design of new adaptive stochastic optimization algorithms, but also summarized recent works on adaptive methods.

1.2.1 Full-gradient vs. stochastic methods: why do we need stochastic algorithms?

The main two optimization approaches used to solve optimization problems (1.1) and (1.2) in machine learning are:

1. full gradient methods, also known as batch methods or Sample Average Approximation (SAA) methods (Verweij et al., 2003). In these methods, the full gradient of the objective function is used to compute the search direction d_k. The simplest algorithm in this category is the *steepest descent algorithm* (Cauchy, 1847), where the update rule is the following:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k). \tag{1.6}$$

2. stochastic optimization methods, also known as Stochastic Approximation (SA) methods (Fu, 2002). They rely on the use of gradient approximations instead of full gradient values. The reference algorithm in this category is *Stochastic Gradient Descent (SGD)* (Robbins and Monro, 1951; Bottou, 2010), where the update takes the following form:

$$x_{k+1} = x_k - \alpha_k g(x_k, \xi_k). \tag{1.7}$$

There are several advantages to the use of a full-gradient method. First of all, there is a great number of deterministic optimization algorithms that were developed over the years and proved to achieve linear, superlinear and quadratic convergence rates for different settings. The rich literature includes line search, trust-region, conjugate gradient and quasi-Newton methods (Nocedal and Wright, 2006; Gould et al., 2005). Second, parallelization can be used when solving finite-sum optimization problems (1.2) in order to compute $f_i(x_k)$ and its gradient for each data point.

However, these advantages do not outweigh the benefits of using a stochastic approach. Let us consider the strongly convex case, then the number of iterations needed to obtain ϵ-optimality for a full-gradient method, in the worst case, is proportional to $n \log(1/\epsilon)$, whereas it is proportional to $1/\epsilon$ for a stochastic method (Bottou et al., 2018). Therefore, if n is sufficiently large, which is the case in large-scale optimization, then we can conclude that the stochastic method would converge faster than the full-gradient method. From a practical point of view, one can argue that many datasets contain redundant data points and that it would be more efficient not to compute the full gradient in this case, but rather to choose a subset of the data points to compute gradient estimates. Moreover, splitting the dataset into training and test datasets is common practice. Let us suppose that we only use half of the entire dataset as a training set. (Bottou et al., 2018) argue that if half of the data is judged to be sufficient to train the model, then it would follow that the use of the entire training dataset to compute the gradient at each iteration is not necessary. This intuition was confirmed by empirically studying the performance of SGD compared to that of the full-gradient limited-memory BFGS method, named after Broyden, Fletcher, Goldfarb and Shanno (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). Figure 1.1 from (Bottou et al., 2018, Figure 3.1, page 18) shows the comparative performance of the two algorithms, used to solve a binary classification problem with a logistic loss objective on a real dataset: RCV1 (Lewis et al., 2004), with a constant step size. We notice that SGD offers an accelerated performance because more steps are taken for the same number of accessed data points. The slower decrease of the empirical risk for SGD within a few epochs can be explained by the high variance of the updates. In fact, computing gradient estimates with a high variance in the beginning of the optimization can be effective because it yields a better exploration of the optimization landscape, resulting in a fast convergence towards a local or global optimum region. However, this high variance influences negatively the performance of the algorithm in that region because SGD may oscillate infinitely around the optimum without converging, especially when a constant step size is used. Moreover, (Bottou and Bousquet, 2008) showed that even when stochastic algorithms are not better optimizers, they have a better generalization performance, which means that the prediction accuracy is higher for stochastic methods. For all these reasons, stochastic optimization algorithms are considered to be more suitable to the context of machine learning and large-scale optimization.

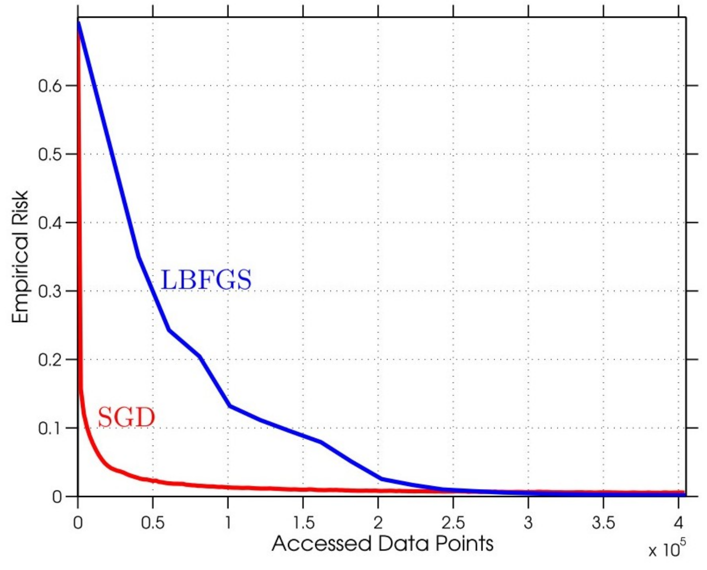

Figure 1.1 The evolution of the empirical risk (loss function) for a binary classification problem using the RCV1 dataset, for SGD and L-BFGS. Figure taken from (Bottou et al., 2018, Figure 3.1, page 18).

1.2.2 Why are we interested in adaptive optimization algorithms?

As we discussed in the previous subsection, stochastic optimization methods are the default choice for solving large-scale optimization problems. However, the low per-iteration cost is just the tip of the iceberg since the search for the most problem-adapted hyperparameters involves significant computational costs. (Asi and Duchi, 2019) highlighted the alarming computational and engineering energy used to find the best set of hyperparameters in training neural networks by citing 3 recent works. The number of computation days needed to tune the optimization algorithm and find the best neural structure was equivalent to: $31,500$ graphics processing unit (GPU) days for (Real et al., 2019), $22,000$ GPU days for (Zoph and Le, 2016) and $750,000$ central processing unit (CPU) days for (Collins et al., 2016). This means that the amount of energy used by the third paper alone (Collins et al., 2016) would be sufficient to drive $4,000$ Toyota Camrys from San Francisco and Los Angeles (≈ 380 miles). These numbers indicate an urgent problem to solve, given the large number of papers that are published each year in the training of neural networks category, as well as their massive industrial use. The immediate solution to this issue is to design optimization algorithms with adaptive parameters, that automatically adjust to the nature of the problem without the need for a hyperparameter search. Famous algorithms in this category include adaptive step size and batch size methods. In Chapter 3, we present a novel first order algorithm with both adaptive step size and batch size.

1.3 Objective

The objectives of our work on first-order optimization methods are:

- Defining a framework of transition from deterministic or semi-deterministic adaptive regularization algorithms to the fully stochastic setting.
- Designing a first order algorithm with adaptive sampling and adaptive step sizes. As discussed in the previous subsections, adaptive methods are highly interesting for both machine learning and optimization research communities.

The objectives of our work on second-order optimization methods are:

- Finding an easy-to-compute expression of the bounds on the eigenvalues of the inverse Hessian approximation, with less restrictive assumptions than related works.
- Leveraging the expression of the bounds to control the quality of the Hessian approximation all along the optimization, leading to a new stochastic damped version of the Limited-Memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm.

1.4 Outline

The two main components of this work are Chapters 3 and 4. Although the two chapters are related under the umbrella of stochastic large-scale optimization, the two approaches are independent in the sense that we do not seek any sort of combination or intersection between them.

In Chapter 2, we present a critical literature review of some of the existing first- and second-order methods to solve optimization problems arising in machine learning. We also dedicate 2 subsections to trust-region methods and variance reduction techniques, as they represent prominent components of the algorithms that we propose.

In Chapter 3, we start by presenting our reference algorithm. Then, we introduce our work on adaptive regularization and adaptive sampling, which resulted in 3 novel algorithms. Finally, we compare the characteristics of these algorithms.

In the same fashion, we start Chapter 4 by presenting the formulation of stochastic damped L-BFGS. Then, we dive into the technical details of our algorithm as well as the intuition behind its design. In the subsection that follows, we discuss its theoretical guarantees. Finally, we discuss the numerical outcomes of our experiments.

In Chapter 5, we offer a general discussion to conclude. We make an assessment of our contributions, discuss the limits of our propositions and offer recommendations for future work.

CHAPTER 2 LITERATURE REVIEW

Gradient-based optimization methods have been widely used to solve online optimization problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) := \mathbb{E}_\xi[F(x, \xi)],$$

and finite-sum optimization problems, where f is defined as

$$\min_{x \in \mathbb{R}^n} f(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x).$$

In this chapter, we offer a non-extensive overview of first and second-order methods that were proposed for this purpose. In Section 2.1, we discuss popular stochastic first-order methods. In Section 2.2, we present trust-region methods used in the stochastic setting and emphasize their theoretical and empirical outcomes. In Section 2.3, we discuss two research directions to reduce the noise in gradient approximations, which are gradient aggregation methods and adaptive sampling. In Section 2.4, we present stochastic second-order methods for machine learning that are related to our work.

Due to the similarity in spirit between trust-region methods and regularization methods, we do not cover the latter in this review. However, cubic regularization methods (Cartis et al., 2011a,b) enjoy favorable complexity bounds that were later used to modify trust-region methods.

2.1 Stochastic first-order optimization methods

The stochastic gradient method or stochastic gradient descent (SGD) (Robbins and Monro, 1951; Bottou, 2010) has played a paramount role in large-scale optimization, where deterministic optimization algorithms are not very effective. The update rule for SGD is very simple $x_{k+1} = x_k - \alpha_k g(x_k, \xi_k)$, which makes it convenient and not computationally expensive. However, SGD does not converge to the solution when a constant step size is used throughout the optimization process (Bottou et al., 2018).

In order to remedy this problem, several works proposed to use a decreasing sequence of step sizes (Bottou, 1998; Kiwiel, 2001). The convergence rate of SGD becomes sublinear in this case (Bottou et al., 2018). Momentum gradient descent, or the heavy-ball algorithm, inspired by the physical interpretation of the optimization procedure, was then introduced by (Polyak, 1964) in order to accelerate the convergence of SGD. It consists of using the following update:

$$x_{k+1} = x_k + v_k, \quad \text{where} \quad v_k = \beta v_{k-1} - \gamma g(x_k, \xi_k).$$

Precisely, β is the momentum term, typically $\beta \in [0, 1]$, $\gamma > 0$ is a constant step size and v_k represents the velocity term, a modified accumulation of past gradient estimates. *Nesterov momentum* is a modified version of this method that was presented by (Nesterov, 1983). In contrast with the heavy-ball method, Nesterov momentum uses an intermediate update of the parameter vector $\hat{x}_{k+1}$, to which the velocity term is added, that is then used to compute the gradient estimate. The update scheme is the following:

$$\hat{x}_{k+1} = x_k + \beta v_{k-1}, \quad x_{k+1} = x_k + v_k, \quad \text{where} \quad v_k = \beta v_{k-1} - \gamma g(\hat{x}_{k+1}, \xi_k).$$

A different line of research focuses on accelerating SGD by addressing the choice of the step size sequence. Such methods use past estimates of the moments of the gradient to compute individual step sizes for each optimization parameter. They have proven to be very efficient for the training of deep networks tasks and are currently the default choice in deep learning. The first algorithm of this kind is the Adaptive Gradient Algorithm (ADAGRAD) (Duchi et al., 2011), which provides adaptive step sizes to all parameters. However, as it relies on the accumulation of the gradient estimates from the beginning of the optimization process, the learning becomes slow. (Tieleman and Hinton, 2012) presented the Root Mean Square Propagation method (RMSPROP), a modified version of ADAGRAD which uses an exponential moving average instead of the accumulation of the gradient estimates. Finally, remarkable improved versions of the combination of RMSPROP and momentum, such as Adaptive Moment Estimation algorithm (ADAM) (Kingma and Ba, 2015), were designed. ADAM keeps estimates of the first and second moments of the gradient in order to perform a coordinate-wise tuning of the learning rate. The regret bound on the convergence rate of ADAM was shown to be competitive with the best results for online optimization problems in the convex setting. In recent years, ADAM became one of the most popular optimization algorithms in deep learning, if not the most popular. However, (Reddi et al., 2018) argued that in many applications, these methods fail to converge to an optimal solution in the convex case, or to a critical point in the non-convex case. More specifically, the authors explained than in these cases, the effect of large informative gradients vanishes quickly due to the exponential moving average. Through a rigorous analysis, they pointed out an incorrect assumption in the proof of ADAM, and proposed a new variant of ADAM, called AMSGRAD, for which the assumption is satisfied.

Although these methods seem to perform better than SGD in the training of neural networks, (Keskar and Socher, 2017) showed that they do not generalize as well as SGD. This means that their performance on unseen data is inferior to that of SGD. Therefore, (Keskar and Socher, 2017) proposed a hybrid algorithm called SWATS for SWitches from ADAM To SGD. As its name indicates, SWATS starts the training procedure using ADAM and switches to SGD when a certain condition is satisfied. To do so, the authors first determined the step size for SGD after the transition from ADAM and maintained its exponential moving average. Then, they obtained the transition condition by comparing the current value of the step size to its exponential moving average. The switch happens when these two are very close. The authors demonstrated empirically that their method yields better testing errors than ADAM and SGD used separately. The results of this work lead to the legitimate question: is it possible to develop other optimization methods that provably enjoy both fast convergence rates and good generalization properties?

2.2 Stochastic trust-region methods

Trust-region methods, alongside line search and quasi-Newton methods, represent some of the most solid, efficient and reliable methods for deterministic optimization. Algorithm 1 is a basic trust-region algorithm in the deterministic setting for solving the optimization problem $\min_{x\in\mathbb{R}^n} f(x)$, where f is a differentiable real-valued function (Conn et al., 2000).

In Algorithm 1, $\mathcal{B}_k := \{x \in \mathbb{R}^n, \|x - x_k\|_k \leq \sigma_k\}$ is the *trust region* and $\|.\|_k$ is a norm that depends on the iteration. σ_k is called the *trust-region radius*. We assume that the norms are uniformly equivalent, i.e., there exist constants c_1 and c_2 such that,

$$c_1\|x\| \leq \|x\|_k \leq c_2\|x\|\,, \quad \forall x \in \mathbb{R}^n,$$

where $\|.\|_k$ is the norm at iteration k and $\|.\|$ is the Euclidean norm. In practice however, we generally choose the same norm for all iterations. At each iteration k, we define the model l_k such that it gives an approximation of the objective function within the trust region. The purpose of step 4 is to compute the decrease rate ρ_k and decide whether to update the parameter vector x_k or not, based on the amount of decrease that the update would achieve.

In practice, l_k is chosen to be a first-order or second-order Taylor series approximation of f, and s_k is computed such that it minimizes l_k directly.

Motivated by their performance in the deterministic setting, researchers have been interested

Algorithm 1 Basic trust-region algorithm

```
Require: $x_0 \in \mathbb{R}^n$ and initial trust-region radius $\sigma_0$
1: Choose constants $\eta_1, \eta_2, \gamma_1$ and $\gamma_2$ such that
       $$0 < \eta_1 \leq \eta_2 < 1 \quad \text{and} \quad 0 < \gamma_1 \leq \gamma_2 < 1.$$
   Set $k = 0$.
2: Choose $\|.\|_k$ and model $l_k$ in $\mathcal{B}_k$.
3: Compute a step $s_k$ that sufficiently reduces the model $l_k$, such that $x_k + s_k \in \mathcal{B}_k$ .
4: Evaluate $f(x_k + s_k)$ and define
       $$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{l_k(x_k) - l_k(x_k + s_k)}$$
   If $\rho_k \geq \eta_1$, then define $x_{k+1} = x_k + s_k$. Otherwise, define $x_{k+1} = x_k$.
5: Set
       $$\sigma_{k+1} \in \begin{cases} [\sigma_k, \infty) & \text{if } \rho_k \geq \eta_2, \\ [\gamma_2 \sigma_k, \sigma_k] & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_1 \sigma_k, \gamma_2 \sigma_k] & \text{if } \rho_k < \eta_1. \end{cases}$$
   Increment $k$ by one and go to step 2.
```

in developing stochastic versions of trust-region methods for several years. (Chang et al., 2013) proposed to use a stochastic trust-region to improve a response-surface method in the context of simulation optimization. Their algorithm uses repeated sampling to obtain approximate values of the function and gradient. The number of samples used to compute the estimates is increased during the optimization to ensure the overall convergence of the algorithm.

To avoid repeated sampling, (Larson and Billups, 2016) presented a novel trust-region algorithm that solves stochastic online derivative-free optimization problems. The authors focused on the case where independent and identically distributed noise, with mean zero and finite variance, is added to the function values. Assuming that the function has bounded level sets and its gradient is Lipschitz continuous, the authors proved that the gradient values converge in probability to zero.

Similarly, (Chen et al., 2018) introduced a general framework for their stochastic trust-region method, called STORM for STochastic Optimization with Random Models. The authors used a probabilistic approach, since their work is based on the use of probabilistic models in a trust-region derivative free framework for nonlinear functions (Bandeira et al., 2014). Therefore, they assumed a certain level of accuracy in the function and model approximations with a fixed probability, in order to prove an almost sure global convergence. Moreover, they showed that obtaining accurate estimates is feasible in various settings and that STORM has a superior performance compared to similar methods. However, STORM was not proven to enjoy competitive convergence rates with these methods.

In contrast to the method proposed by (Larson and Billups, 2016), STORM's global convergence does not rely on the assumption that the function estimates are unbiased. Moreover, (Larson and Billups, 2016) assumed that the probabilities of having accurate function estimates and an accurate model converge to 1, whereas a fixed probability is used in the convergence analysis of STORM. However, STORM restricts the acceptable step sizes whenever the gradient norm becomes small, whereas the algorithm proposed by (Larson and Billups, 2016) uses adaptive step sizes based on the quality of the model approximation of the function's behaviour.

Although STORM demonstrated a competitive performance for the training of a linear classifier by minimizing a smooth regularized logistic loss, it is unclear, and even doubtful, that it would give the same results for deep neural networks. The nature of this latter setting makes that obtaining sufficiently accurate estimates would most probably require the use of the entire dataset to compute the estimates within a few epochs.

(Blanchet et al., 2019) proposed a convergence rate analysis for STORM in the first and second-order settings. In terms of the expected number of iterations to obtain a desired accuracy level $\epsilon > 0$ for the norm of the exact gradient, they showed a similar upper bound to that established by recent work on stochastic algorithms for non-convex optimization (Reddi et al., 2016; Nguyen et al., 2017a). This bound represents an extension of best-known worst-case complexity results of first-order optimization algorithms in the non-convex setting (Nesterov, 2013). The authors did not make the assumption that the function, gradient, and possibly Hessian estimates are unbiased, which is a common assumption in similar works. However, they did require that these estimates maintain a sufficient accuracy with a sufficiently high probability. Additionally, they left the question of establishing a termination criterion such that $|f(x_t)| \leq \epsilon$, where $t \in \mathbb{N}$ is the last iteration, to a future work that would make use of the theoretical analysis they provided.

Note that the previous works required a certain level of accuracy of the function and gradient estimates to be maintained throughout the optimization process. Therefore, the decrease rate $\rho_k = (f(x_k) - f(x_k + s_k)) \,/\, (l_k(x_k) - l_k(x_k + s_k))$ remains a meaningful measure of the model's approximation quality. (Curtis et al., 2019) proposed a new Trust-Region-ish algorithm (TRish) for stochastic optimization that drops this requirement. The authors are interested in solving online and finite-sum minimization problems. At iteration k, given the gradient estimate $g(x_k, \xi_k)$ and the step size α_k, they considered the following trust-region sub-problem that consists of minimizing the first-order approximation model $f(x_k) + g(x_k, \xi_k)^\top s_k$:

$$\min_{s_k \in \mathbb{R}^n} f(x_k) + g(x_k, \xi_k)^\top s_k \quad \text{s.t.} \quad \|s_k\| \le \alpha_k,$$

where s_k is the step in iteration k. The closed-form solution to this problem is $s_k = -\alpha_k g(x_k, \xi_k)/\|g(x_k, \xi_k)\|$. The authors used an example where $x_k \in \mathbb{R}$ to illustrate how employing this step for all values of the gradient norm may hinder the convergence of the algorithm in expectation. The solution that they proposed is to employ the step $s_k = -\alpha_k g(x_k, \xi_k)/\|g(x_k, \xi_k)\|$ only when the gradient norm falls in a specific interval. Therefore, they suggested the following scheme:

$$s_k = -\begin{cases} \gamma_{1,k}\alpha_k g(x_k, \xi_k), & \text{if} \quad \|g(x_k, \xi_k)\| \in [0, \frac{1}{\gamma_{1,k}}), \\ \alpha_k g(x_k, \xi_k)/\|g(x_k, \xi_k)\|, & \text{if} \quad \|g(x_k, \xi_k)\| \in [\frac{1}{\gamma_{1,k}}, \frac{1}{\gamma_{2,k}}], \\ \gamma_{2,k}\alpha_k g(x_k, \xi_k), & \text{if} \quad \|g(x_k, \xi_k)\| \in (\frac{1}{\gamma_{2,k}}, \infty), \end{cases}$$

where $\gamma_{1,k} > \gamma_{2,k} > 0$ are constants that have to be chosen carefully alongside the step size α_k. The update rule is similar to that of SGD $x_{k+1} = x_k + s_k$, without any additional steps or constraints. TRish enjoys similar convergence properties to those of SGD and proved to be more effective than SGD in training neural networks and solving logistic regression problems for several datasets. Although the authors mentioned that more numerical experiments would be needed to explore the full potential of TRish, their simple method can inspire new stochastic trust-region methods that drop the accuracy requirement, making them more adapted to the context of deep learning.

2.3 Variance reduction methods

SGD, as well as several optimization methods that are based on it, suffers from the detrimental effect of using noisy estimates of the gradient. This implies a non-vanishing variance of the gradient estimates, i.e., $\mathbb{E}_{\xi_k}[\|g(x_k, \xi_k) - \nabla f(x_k)\|^2]$ can be very large, leading to slow convergence.

Two of the most promising directions to address this problem are gradient aggregation techniques and adaptive sampling, as both lead to a reduction of the noise of the gradient estimates.

2.3.1 Gradient Aggregation

(Johnson and Zhang, 2013) proposed the Stochastic Variance Reduced Gradient (SVRG) algorithm to alleviate the issue of noisy gradients in SGD. For SVRG, the gradient estimate used is computed as follows:

$$g(x_k, \xi_k) = f_{i_t}(x_k) - \left(f_{i_t}(x_l) - \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_l) \right), \tag{2.1}$$

where $i_t \in \{1, \ldots, N\}$ is chosen randomly and x_l is the parameter vector available at the beginning of the current optimization epoch/cycle. Therefore, $\frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_l)$ represents the full gradient evaluated at the beginning of the epoch. Moreover, $g(x_k, \xi_k)$ is an unbiased estimate of the full gradient evaluated at x_k that is expected to have a smaller variance than $f_{i_t}(x_k)$. In fact, $\left(f_{i_t}(x_l) - \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_l) \right)$ can be seen as the bias in the gradient estimate $f_{i_t}(x_l)$, and it is used to correct the gradient approximation in iteration k (Bottou et al., 2018).

SVRG has an edge over other algorithms enjoying similar fast convergence rates, such as the Stochastic Average Gradient (SAG) method (Roux et al., 2012), the adaptation of the Stochastic Dual Coordinate Ascent (SDCA) to deep learning (Shalev-Shwartz and Zhang, 2013) and SAGA (Defazio et al., 2014), since it does not require the storage of the gradient estimates for each data point. While SVRG's linear convergence rate was only proven for smooth and strongly convex function, SAGA provides a linear convergence rate for non-strongly convex problems as well without any modification of the version of SAGA used to solve convex problems. Other variance reduction algorithms were proposed later on, such as the StochAstic Recursive grAdient algoritHm (SARAH) and its practical version SARAH+ (Nguyen et al., 2017b) for the strongly convex case, as well as the Stochastic Path-Integrated Differential EstimatoR (SPIDER) by (Fang et al., 2018; Wang et al., 2019) for the non-convex case.

Variance reduction methods by gradient aggregation were demonstrated to work well in the non-convex setting (Reddi et al., 2016; Allen-Zhu and Hazan, 2016). Nevertheless, more recent works (Defazio and Bottou, 2018; Chavdarova et al., 2019) argued that their naive application is ineffective in the particular context of non-convex optimization problems, which characterizes the training of deep learning models. Moreover, (Bottou et al., 2018) pointed out that the computing times for SVRG, SAGA and SAG grow with the number of samples N. Therefore, for very large datasets, these algorithms can be less efficient than SGD or its other popular variants mentioned in Section 2.1.

2.3.2 Adaptive sampling

The use of an adaptive batch size throughout the optimization process is another promising variance reduction technique that has been receiving increasing attention in the past years.

(Friedlander and Schmidt, 2012) proved that they can improve their algorithm's convergence rate from sublinear to linear if they used adaptive sampling to control the error in the gradient estimates. Their *hybrid* method consists of increasing the batch size at a geometric rate using a preset formula, such that the algorithm resembles SGD in it first iterations and a deterministic, full-gradient approach towards the end of the optimization process. However, the adaptive sampling heuristic presented by their work increases the batch size very fast, which makes it impractical for large-scale optimization. Other works proposed to use the available information about the gradient estimate in each iteration in order to choose the new batch size. Let us first explain their theoretical context. We know that $g(x_k, \xi_k)$ is a descent direction if, for a given $w_g \in [0, 1)$, we have

$$\|g(x_k, \xi_k) - \nabla f(x_k)\| \leq w_g \|g(x_k, \xi_k)\|. \tag{2.2}$$

This inequality is similar to what is known as the *norm test*, which is,

$$\|g(x_k, \xi_k) - \nabla f(x_k)\| \leq w_g \|\nabla f(x_k)\|. \tag{2.3}$$

Inequality (2.2) characterizes the error in the gradient estimate and w_g can be seen as an error threshold. In many algorithms, including trust-region methods described in the previous section, the error threshold w_g at each iteration k needs to be bounded from above. One can then try to find the batch size m_k such that (2.2) holds for a given w_g. Nevertheless, one would need first to find a way to evaluate the left-hand side of (2.2), $\|g(x_k, \xi_k) - \nabla f(x_k)\|$ without computing the full gradient.

(Byrd et al., 2012) used the fact that if $g(x_k, \xi_k)$ is an unbiased gradient estimate of $\nabla f(x_k)$, which implies that

$$\mathbb{E}_{\xi_k}[\|g(x_k, \xi_k) - \nabla f(x_k)\|^2] = \|\mathbb{V}_{\xi_k}[g(x_k, \xi_k)]\|_1, \tag{2.4}$$

where $\mathbb{V}_{\xi_k}[g(x_k, \xi_k)]$ is a vector of the same length as x_k. Let us elaborate on the calculation resulting in a change of norm in equation (2.4). Using the assumption that $g(x_k, \xi_k)$ is an unbiased estimate of $\nabla f(x_k)$, we have

$$\begin{aligned}
\mathbb{E}_{\xi_k}[\|g(x_k,\xi_k)-\nabla f(x_k)\|^2] &= \mathbb{E}_{\xi_k}[\|g(x_k,\xi_k)\|^2 - 2\nabla f(x_k)^\top g(x_k,\xi_k) + \|\nabla f(x_k)\|^2] \\
&= \mathbb{E}_{\xi_k}[\|g(x_k,\xi_k)\|^2] - \|\nabla f(x_k)\|^2 \\
&= \sum_{l=1}^{n} \left(\mathbb{E}_{\xi_k}\left[(g_l(x_k,\xi_k))^2 \right] - (\nabla f_l(x_k))^2 \right) \\
&= \sum_{l=1}^{n} \mathbb{V}_{\xi_k}[g_l(x_k,\xi_k)] \\
&= \|\mathbb{V}_{\xi_k}[g(x_k,\xi_k)]\|_1,
\end{aligned}$$

where $g_l(x_k,\xi_k)$ and $\nabla f_l(x_k)$ represent the l-th components of the vectors $g(x_k,\xi_k)$ and $\nabla f(x_k)$, respectively. Hence, the definition of $\mathbb{V}_{\xi_k}[g(x_k,\xi_k)]$ as a vector of the same length as x_k.

Then, as in (Freund and Walpole, 1971, page 183), we have

$$\mathbb{V}_{\xi_k}[g(x_k,\xi_k)] = \frac{\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k))}{m_k}\,\frac{(N-m_k)}{N-1}. \tag{2.5}$$

Since the computation of the sample variance $\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k))$ is expensive, (Byrd et al., 2012) suggested to use the following estimate:

$$\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k)) \approx \frac{1}{m_k-1}\sum_{i\in m_k}(\nabla f_i(x_k) - g(x_k,\xi_k))^2, \tag{2.6}$$

where the square is applied element wise.

It follows from (2.4) and (2.5) that

$$\mathbb{E}_{\xi_k}[\|g(x_k,\xi_k)-\nabla f(x_k)\|^2] = \frac{\|\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k))\|_1}{m_k}\,\frac{(N-m_k)}{N-1}. \tag{2.7}$$

In the context of large-scale optimization, we can take the limit $N\to\infty$. Hence,

$$\mathbb{E}_{\xi_k}[\|g(x_k,\xi_k)-\nabla f(x_k)\|^2] \approx \frac{\|\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k))\|_1}{m_k}. \tag{2.8}$$

Therefore, inequality (2.2) implies

$$\frac{\|\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k))\|_1}{m_k} \le w_g^2\|g(x_k,\xi_k)\|^2. \tag{2.9}$$

This inequality shows the direct link between the batch size m_k and the gradient error threshold w_g.

(Byrd et al., 2012) then used the following adaptive sampling heuristic:

- In iteration k, we sample a new batch of size $m_k = m_{k-1}$, then we compute the sample variance and mini-batch gradient in (2.9).
- If condition (2.9) holds, we use mini-batch m_k in iteration k. Otherwise, we define the new batch size $\tilde{m}_k$ as follows:

$$\tilde{m}_k = \left\lceil \frac{\|\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k))\|_1}{w_g^2\|g(x_k,\xi_k)\|^2} \right\rceil. \tag{2.10}$$

Note that to be able to define the new batch as in (2.10), (Byrd et al., 2012) assumed that the change in the batch is gradual and that

$$\|\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k))\|_1 \approx \|\mathbb{V}_{i\in\tilde{\xi}_k}(\nabla f_i(x_k))\|_1 \quad \text{and} \quad \|g(x_k,\tilde{\xi}_k)\| \approx \|g(x_k,\xi_k)\|, \tag{2.11}$$

where $\tilde{\xi}_k$ corresponds to $\tilde{m}_k$. Therefore, (Byrd et al., 2012) defined a heuristic for adapting the batch size successfully to available information on the gradient estimates. (Pasupathy et al., 2015) showed that for a linearly (resp. sublinearly) convergent batch optimization algorithm to be efficient, i.e., have the lowest computational complexity, the sampling rate should be geometric (resp. slightly faster than geometric). However, the sampling strategy by (Byrd et al., 2012) does not guarantee that the sampling rate is geometric. Additionally, one can question the validity of the assumptions in (2.11).

(Hashemi et al., 2014) used a similar *norm test* to define their sequential procedure for adaptive sampling. They ensured that the sample size grows geometrically by introducing a geometric growth factor γ that the user needs to choose. However, the authors did not explore the convergence properties of their adaptive sampling scheme.

In contrast, (Bollapragada et al., 2018a) proposed to replace the *norm test* with an *inner product* test. It consists of satisfying the following condition:

$$\frac{\mathbb{E}_{i\in\xi_k}\left[\left(\nabla f_i(x_k)^\top \nabla f(x_k) - \|\nabla f(x_k)\|^2\right)^2\right]}{m_k} \le \theta^2 \|\nabla f(x_k)\|^4, \tag{2.12}$$

with $\theta > 0$. This condition ensures that the variance of the inner product $\nabla f_i(x_k)^\top \nabla f(x_k)$ is controlled. The authors suggested to approximate the variance in the left-hand side with the sample variance, and the full gradient with the mini-batch gradient. Inequality (2.12) becomes

$$\frac{\mathbb{V}_{i\in\xi_k}\left[\nabla f_i(x_k)^\top g(x_k,\xi_k)\right]}{m_k} \leq \theta^2\|g(x_k,\xi_k)\|^4, \tag{2.13}$$

where

$$\mathbb{V}_{i\in\xi_k}\left[\nabla f_i(x_k)^\top g(x_k,\xi_k)\right] = \frac{1}{m_k-1}\sum_{i\in\xi_k}\left(\nabla f_i(x_k)^\top g(x_k,\xi_k) - \|g(x_k,\xi_k)\|^2\right)^2. \tag{2.14}$$

Condition (2.13) is the *approximate inner product test*. The authors argue that this condition is less restrictive than the *norm test*, as it only requires for the gradient estimate to lie within an infinite band around the full gradient, whereas the norm test requires that it lies within a ball centered at the full gradient. The *approximate inner product test* is then used to change the batch size adaptively, following the same kind of heuristic used by (Byrd et al., 2012). They also used similar approximations as (2.11). The authors proved global linear convergence for strongly convex functions, and global convergence for non-convex functions. Their empirical results show that the inner product test is more effective and that the batch size growth is slower than the batch size growth when using the norm test.

(Bahamou and Goldfarb, 2019) demonstrated empirically that both methods,i.e, norm test and inner product test, result in a fast growth of the used sample size that becomes eventually very large within a few hundreds of iterations. They introduced a new test, called the *acute-angle test* that consists of satisfying the following condition:

$$\mathbb{E}\left(\left\|\frac{g(x_k,\xi_k)}{\|g(x_k,\xi_k)\|} - \frac{g(x_k,\xi_k)^\top\nabla f(x_k)}{\|\nabla f(x_k)\|\|\nabla f(x_k)\|^2}\nabla f(x_k)\right\|^2\right) \leq p\nu^2, \tag{2.15}$$

where the constant $0 < p < 1$ represents the probability of test failure and the constant $0 < \nu < 1$ represents the level of angle acuteness. Condition (2.15) guarantees that the normalised gradient estimate stays close to its projection on the full normalized gradient. Figure 2.1 from (Bahamou and Goldfarb, 2019, Figure 2, page 4) shows the difference between the three tests.

The method developed by (Bahamou and Goldfarb, 2019) adaptively controls the step size as well. The authors did not provide the convergence rate of their method, but they proved the global convergence of their algorithm. Moreover, they made the assumption that f is self-concordant.

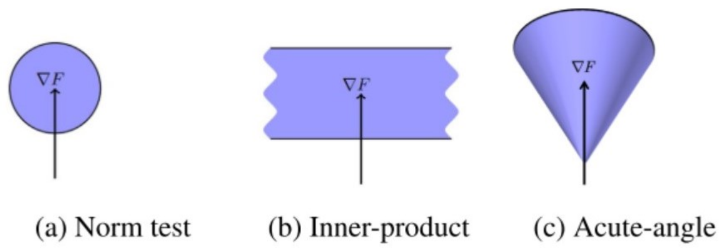

Figure 2.1 Acceptable values of the gradient estimate should fall in shaded areas: (a) for the norm test, (b) for the inner product test, (c) for the acute-angle test. ∇F represents the full gradient. Figure taken from (Bahamou and Goldfarb, 2019, Figure 2, page 4).

2.4 Stochastic second-order optimization methods

Using optimization algorithms that incorporate second-order information, such as Newton's method (Dennis Jr and Schnabel, 1996), can result in faster convergence rates that can even be quadratic in some cases. Quasi-Newton methods are widely used because they take advantage of the information on the gradients gathered along previous iterations to construct approximations of the Hessian without direct access to it.

The most popular Quasi-Newton method is the BFGS method (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970).

This method was shown to achieve a local superlinear convergence rate (Dennis and Moré, 1974). Its update is in the following form:

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k), \tag{2.16}$$

where $H_k =: B_k^{-1}$ and B_k is a symmetric positive definite approximation to the Hessian.

To add the curvature information obtained at each iteration, we define

$$s_k = x_{k+1} - x_k, \quad \text{and} \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k). \tag{2.17}$$

Most importantly, the updated Hessian approximation must verify the secant equation

$$B_{k+1} s_k = y_k, \tag{2.18}$$

which means that the second-order Taylor expansion is verified for s_k and y_k.

Because B_{k+1} must be positive definite to ensure that d_k is a descent direction, s_k and y_k should satisfy the curvature condition

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

The recursive rule to update H_k is given by

$$H_{k+1} = (I - \rho_k s_k y_k^\top) H_k (I - \rho_k y_k s_k^\top) + \rho_k s_k s_k \top, \quad \text{where} \quad \rho_k = \frac{1}{s_k^\top y_k}. \tag{2.19}$$

The initial approximation H_0 has to be chosen positive definite. Under these conditions, H_k is positive definite (see, for instance, Fletcher, 1970). However, it is not always possible to compute the complete BFGS update. The limited-memory BFGS is a possible alternative (Nocedal, 1980; Liu and Nocedal, 1989), because it is more memory-friendly, thus, better adapted to large-scale optimization.

Considering a positive integer p, the approximation of the Hessian only depends on the last p iterations and an initial H_k^0, which is chosen positive definite

$$\begin{aligned} H_k = {} & (V_{k-1}^\top \dots V_{k-p}^\top) H_k^0 (V_{k-p} \dots V_{k-1}) \\ & + \rho_{k-p} (V_{k-1} \top \dots V_{k-p+1} \top) s_{k-p} s_{k-p} \top (V_{k-p+1} \dots V_{k-1}) \\ & + \rho_{k-p+1} (V_{k-1} \top \dots V_{k-p+2} \top) s_{k-p+1} s_{k-p+1} \top (V_{k-p+2} \dots V_{k-1}) \\ & \vdots \\ & + \rho_{k-1} s_{k-1} s_{k-1} \top, \end{aligned} \tag{2.20}$$

where $V_k = (I - \rho_k s_k y_k \top)$.

This method was successfully adapted to the stochastic setting by several works. (Schraudolph et al., 2007) proposed the stochastic Online BFGS (oBFGS) and Online Limited storage BFGS (oLBFGS) algorithms for online optimization of convex functions. Inspired by oLBFG, (Bordes et al., 2009) design SGD-QN, a Quasi-Newton Stochastic Gradient-Descent algorithm that replaces the inverse Hessian matrix by a diagonal rescaling matrix.

Their algorithm follows the recommendation from (LeCun et al., 1998) to rescale input features to improve the condition number of the Hessian matrix in multi-layer neural networks. However, the convergence analysis of SGD-QN only applies to the linear case of Support Vector Machines (SVMs) and assumes a positive definite bounded Hessian. ADAGRAD (Duchi et al., 2011) also exploits the idea of using a diagonal matrix as an approximation to the Hessian in the convex setting. However, the results for ADAGRAD are more general SGD-QN's since the convergence analysis for ADAGRAD was not based on the assumptions that the objective is smooth, nor did the authors assume that the Hessian is a positive definite bounded matrix.

In the stochastic non-convex setting, we lose the guarantee that the curvature condition holds, thus we cannot claim that the updated inverse Hessian matrices are definite positive.

One way to ensure that the updated inverse Hessian remains sufficiently positive definite in the non-convex setting is to apply *damping*, which means replacing the gradient displacement vector y_k by a modified expression of it, $\tilde{y}_k$, such that the curvature condition holds for s_k and $\tilde{y}_k$. In this vein, (Wang et al., 2017) proposed a general framework for stochastic Quasi-Newton methods for non-convex optimization, with available information on stochastic gradients that is obtained through a Stochastic First-order Oracle (SFO). Within this framework, they introduced a Stochastic Damped version of L-BFGS (SdLBFGS).

Additionally, (Wang et al., 2017) introduced a variance-reduced version of this algorithm (SdLBFGS-VR), that allows for the use of a constant step size, which speeds up the convergence. The authors adapted an SVRG-like variance-reduction technique, which was shown to be effective when combined with stochastic second-order optimization methods in recent years. In fact, (Moritz et al., 2016) proved that their variance-reduced stochastic L-BFGS algorithm achieves a linear convergence rate for strongly convex and smooth functions. This result is obtained by successfully applying an SVRG-like variance reduction to the stochastic quasi-Newton method proposed by (Byrd et al., 2016). In the same fashion, (Gower et al., 2016) combined a stochastic limited-memory block BFGS approach with an SVRG-type of variance reduction to achieve a linear convergence rate for convex functions. The block BFGS method consists of solving the following *sketched inverse equation*:

$$H_k \nabla^2 f(x_k, \xi_t) D_k = D_k,$$

instead of solving the *inverse equation*

$$H_k \nabla^2 f(x_k, \xi_t) = \mathbb{I},$$

to obtain the inverse Hessian approximation H_k. Here $\nabla^2 f(x_k, \xi_t)$ represents a sub-sampled Hessian matrix and $D_k \in \mathbb{R}^{n \times q}$ is a randomly generated matrix such that $n \gg q$, which means that D_k has relatively few columns.

Although SdLBFGS-VR by (Wang et al., 2017) proved to have interesting theoretical and experimental properties, we cannot verify whether or not the Hessian matrix stays well-conditioned throughout the optimization. Moreover, the overall convergence may be hindered if the Hessian approximation becomes singular. (Mokhtari and Ribeiro, 2014) proposed a REgularized version of Stochastic BFGS (RES) to address the near-singularity issue. More specifically, they proposed to define the Hessian approximation B_{k+1} such that it represents the solution to the following regularization problem:

$$B_{k+1} \in \underset{Z \in \mathbb{R}^{n\times n}}{\operatorname{argmin}} \ \operatorname{tr}[B_k^{-1}(Z - \delta \mathbb{I})] - \log\det[B_k^{-1}(Z - \delta\mathbb{I})] - n, \tag{2.21}$$

$$\text{s.t.} \qquad Zs_k = y_k, \quad Z \succeq 0, \tag{2.22}$$

where $\delta > 0$ is the regularization parameter. Note that the expression

$$\operatorname{tr}[B_k^{-1}(Z)] - \log\det[B_k^{-1}(Z)] - n,$$

represents the Gaussian differential entropy between the two distributions $\mathcal{N}(0, B_k)$ and $\mathcal{N}(0, Z)$. It is equal to zero if and only if $B_k = Z$. The motivation behind this choice is that the secant condition $B_{k+1}s_k = y_k$ is not sufficient to characterize B_{k+1} completely. Therefore, one can also require for B_{k+1} and B_k to have close differential entropy. The conditions $Zs_k = y_k$ and $Z \succeq 0$ guarantee that B_{k+1} satisfies the secant condition and that it is positive semidefinite, respectively. Finally, the regularization as formulated in (2.22) ensures that the eigenvalues of B_{k+1} are superior to the positive regularization constant $\delta > 0$. Therefore, the singularity of the sequence B_k is avoided and the authors provided a closed-form solution B_{k+1} to problem (2.22). RES achieves a linear convergence rate in expectation for strongly convex functions. Numerical experiments demonstrated that RES yields a faster convergence compared to SGD and non-regularized stochastic BFGS.

To solve the singularity problem in the non-convex case, (Chen et al., 2019) introduced their Stochastic Damped and REGularized L-BFGS (Sd-REG-LBFGS) method. The method is a careful combination of the stochastic damped L-BFGS proposed by (Wang et al., 2017) and the regularized BFGS by (Mokhtari and Ribeiro, 2014). The authors defined a new rule to choose the regularization parameter δ and proved that this rule is sufficient to guarantee that the Hessian approximation remains positive definite. Inspired by the work of (Byrd et al., 2016), the authors did not update the Hessian approximation at each iteration. They rather computed the average Hessian approximations at regular intervals. This strategy reduces the computation costs that would disadvantage Sd-REG-LBFGS in comparison to SGD. The authors proved the overall convergence of the Sd-REG-LBFGS method and provided the number of iterations needed to converge. Numerical results showed that the algorithm outperforms the stochastic damped L-BFGS algorithm by (Wang et al., 2017) in the convex and non-convex settings for both real and synthetic datasets. Sd-REG-LBFGS was also shown to be less sensitive to the batch size and memory size choices than SdLBFGS. Interestingly, the authors did not apply an SVRG-type of variance reduction, but they implied that the use of regularization helps reduce the variance, which may explain their algorithm's robustness to the choice of the batch size.

An alternative approach to solve the singularity problem consists of monitoring the quality of the inverse Hessian approximation during the optimization by estimating its maximum and minimum eigenvalues.

To the best of our knowledge, there has been no work that proposed to use the bounds on the eigenvalues of the Hessian approximation matrix in order to improve the performance of stochastic damped limited memory BFGS.

CHAPTER 3 STOCHASTIC ADAPTIVE REGULARIZATION WITH DYNAMIC SAMPLING

In this chapter, we present a novel adaptive first-order algorithm to solve convex and non-convex large-scale unconstrained optimization problems. Our method distinguishes two phases of the stochastic optimization process, and adapts the step size and batch size accordingly.

3.1 Introduction

In the first part of this chapter, our problems of interest are unconstrained optimization problems of the form

$$\min_{x \in \mathbb{R}^n} \; f(x), \tag{3.1}$$

where we consider a semi-deterministic context, which means that we have access to exact function values and inexact estimates of the gradient values. Then, we shift our attention to optimization problems that arise in machine learning in the second part. Particularly, online optimization problems

$$\min_{x \in \mathbb{R}^n} f(x) := \mathbb{E}_\xi[F(x, \xi)], \tag{3.2}$$

where $\xi \in \mathbb{R}^d$ denotes a random variable and $F : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable function, possibly non-convex; and finite sum optimization problems, where f takes the form

$$\min_{x \in \mathbb{R}^n} f(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x), \tag{3.3}$$

where f_i represents the loss function that corresponds to the i-th element of the dataset and N is the size of the dataset.

In Section 3.2, we assume that the accuracy of gradient estimates can be selected by the user, but we do not require errors to converge to zero. We then propose our Adaptive Regularization with Inexact Gradients (ARIG) algorithm and establish global convergence and optimal iteration complexity. This algorithm was mainly developed by my colleague Tiphaine Bonniot de Ruisselet (Bonniot, 2018). We build upon ARIG to present two stochastic variants in Section 3.3, as well as a new stochastic adaptive regularization algorithm with dynamic sampling that leverages Pflug diagnostic in Section 3.4. Finally, we present a brief conclusion in Section 3.5.

Our contributions:

- Defining a framework of transition from deterministic or semi-deterministic trust-region methods to stochastic trust-region methods.
- Proposing a stochastic version of the initial adaptive regularization with inexact gradients algorithm, offering the same theoretical guarantees.
- Leveraging dynamic sampling to guarantee the accuracy condition on the gradient estimates in adaptive regularization algorithms.
- Proposing a novel algorithm that combines adaptive regularization and adaptive sampling by exploiting the two-phase nature of stochastic optimization procedures.

3.2 Adaptive Regularization with Inexact Gradients (ARIG)

We consider the optimization problem (3.1) and assume that we can obtain approximations of $\nabla f(x)$. More specifically, we assume that it is possible to obtain $\nabla f(x, \omega_g) \approx \nabla f(x)$ using a user-specified relative error threshold $\omega_g > 0$, i.e.,

$$\|\nabla f(x, \omega_g) - \nabla f(x, 0)\| \leq \omega_g \|\nabla f(x, \omega_g)\|, \quad \text{with } \nabla f(x, 0) = \nabla f(x). \tag{3.4}$$

Geometrically speaking, this assumption requires that the full gradient lie within a ball centered at the gradient estimate.

We define our assumptions as follows:

Assumption 1. *The function f is bounded below on $\mathbb{R}^n$, i.e., there exists κ_{low} such that $f(x) \geq \kappa_{low}$ for all $x \in \mathbb{R}^n$.*

Assumption 2. *The function f is continuously differentiable over $\mathbb{R}^n$.*

Assumption 3. *The gradient of f is Lipschitz continuous, i.e., there exists $L > 0$ such that for all x, $y \in \mathbb{R}^n$, $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$.*

3.2.1 Formulation for ARIG

Let $T(x,s)$ be the Taylor series of the function $f(x+s)$ at x truncated at the first-order, i.e.,

$$T(x,s) = f(x) + \nabla f(x)^\top s.$$

We recall the following results implied by Taylor's theorem (Birgin et al., 2017).

For all $x, s \in \mathbb{R}^n$,

$$|f(x+s) - T(x,s)| \leq \tfrac{1}{2} L \|s\|^2, \tag{3.5}$$

$$\|\nabla f(x+s) - \nabla_s T(x,s)\| \leq L\|s\|, \tag{3.6}$$

where L represents the Lipschitz constant defined in Assumption 3.

At each iteration k, we consider the following approximate Taylor series using the inexact gradient $\nabla f(x, \omega_g)$ defined in (3.4):

$$\bar{T}_k(s) = f(x_k) + \nabla f(x_k, \omega_g^k)^\top s. \tag{3.7}$$

From inequality (3.5), the Cauchy-Schwarz inequality and the error specification on the inexact gradient in (3.4), we find that at each iteration k and for all $s \in \mathbb{R}^n$, we have

$$\begin{aligned} |f(x_k+s) - \bar{T}_k(s)| &\leq |f(x_k+s) - T(x_k,s)| + |T(x_k,s) - \bar{T}_k(s)| \\ &\leq |f(x_k+s) - T(x_k,s)| + |\nabla f(x_k)^\top s - \nabla f(x_k, \omega_g^k)^\top s| \\ &\leq \tfrac{1}{2} L\|s\|^2 + \|\nabla f(x_k) - \nabla f(x_k, \omega_g^k)\| \, \|s\| \\ &\leq \tfrac{1}{2} L\|s\|^2 + \omega_g^k \|\nabla f(x_k, \omega_g^k)\| \, \|s\|. \end{aligned} \tag{3.8}$$

Similarly, using the inequality (3.6) and the tolerance on the inexact gradient (3.4), we have

$$\begin{aligned} \|\nabla f(x_k+s) - \nabla_s \bar{T}_k(s)\| &\leq \|\nabla f(x_k+s) - \nabla_s T(x_k,s)\| + \|\nabla_s T(x_k,s) - \nabla_s \bar{T}_k(s)\| \\ &\leq \|\nabla f(x_k+s) - \nabla_s T(x_k,s)\| + \|\nabla f(x_k) - \nabla f(x_k, \omega_g^k)\| \\ &\leq L\|s\| + \omega_g^k \|\nabla f(x_k, \omega_g^k)\|. \end{aligned} \tag{3.9}$$

In order to describe our algorithm, we also define the approximate regularized Taylor series

$$l_k(s) = \bar{T}_k(s) + \tfrac{1}{2}\sigma_k \|s\|^2, \tag{3.10}$$

whose gradient is

$$\nabla_s l_k(s) = \nabla_s \bar{T}_k(s) + \sigma_k s = \nabla f(x_k, \omega_g^k) + \sigma_k s,$$

where σ_k is the regularization factor updated at each iteration according to the algorithm's mechanisms described in Section 3.2.2.

3.2.2 Complete Algorithm

Algorithm 2 summarizes our Adaptive Regularization with Inexact Gradients method. The technique is called adaptive regularization because $\frac{1}{2}\sigma_k\|s\|^2$ represents a regularization term, whereas $\bar{T}_k(s)$ is the main function approximation in the definition of the model (3.10). The adaptive regularization property is described in step 5 of Algorithm 2, where the regularization parameter σ_k is updated based on the quality of the model l_k. Note that this algorithm is similar to a trust-region algorithm, such as defined in Algorithm 1, Section 2.2. The regularization parameter σ_k can be interpreted as the inverse of a trust-region radius.

Let us explain the intuition behind this algorithm. The decrease rate

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{\bar{T}_k(0) - \bar{T}_k(s_k)} = \frac{f(x_k) - f(x_k + s_k)}{\frac{1}{\sigma_k}\|\nabla f(x_k, \omega_g^k)\|^2},$$

represents the actual decrease in the function values related to the expected decrease by the model if the trial step s_k is used. To accept the trial step s_k, the decrease rate ρ_k needs to be positive and superior to the constant η_1. The regularization parameter is then updated using the following scheme:

- If $\rho_k \geq \eta_2$, then the model's quality is sufficiently high to decrease the regularization parameter, i.e., increase the trust-region radius.
- If $\eta_1 \leq \rho_k < \eta_2$, then the model's quality is sufficient to take the trial step s_k, but insufficient to decrease the regularization parameter. We choose to increase the regularization parameter.
- If $\rho_k < \eta_1$, then the model's quality is low and not sufficient to take the trial step s_k.

Therefore, we increase the model's regularization parameter more aggressively than we do in the previous case, which is equivalent to decreasing the trust-region radius.

In the following section, we offer a convergence analysis for ARIG under the assumptions presented.

Algorithm 2 Adaptive Regularization with Inexact Gradients - ARIG

```
Require: $x_0 \in \mathbb{R}^n$
1: Choose the accuracy level $\epsilon > 0$, the initial regularization parameter $\sigma_0 > 0$, and the
   constants $\eta_1, \eta_2, \gamma_1, \gamma_2, \gamma_3$ and $\sigma_{\min}$ such that
       $\sigma_{\min} \in (0, \sigma_0],\ 0 < \eta_1 \le \eta_2 < 1 \quad \text{and} \quad 0 < \gamma_1 < 1 < \gamma_2 < \gamma_3.$     (3.11)
   Set $k = 0$.
2: Choose $\omega_g^k$ such that $0 < \omega_g^k \le 1/\sigma_k$ and compute $\nabla f(x_k, \omega_g^k)$ such that (3.4) holds. If
   $\|\nabla f(x_k, \omega_g^k)\| \le \epsilon/(1+\omega_g^k)$, terminate with the approximate solution $x_\epsilon = x_k$.
3: Compute the step $s_k = -\frac{1}{\sigma_k}\nabla f(x_k, \omega_g^k)$.
4: Evaluate $f(x_k + s_k)$ and define
       $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{\bar{T}_k(0) - \bar{T}_k(s_k)} = \frac{f(x_k) - f(x_k + s_k)}{\frac{1}{\sigma_k}\|\nabla f(x_k, \omega_g^k)\|^2}.$     (3.12)
   If $\rho_k \ge \eta_1$, then define $x_{k+1} = x_k + s_k$. Otherwise, define $x_{k+1} = x_k$.
5: Set
       $\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1\sigma_k), \sigma_k] & \text{if } \rho_k \ge \eta_2, \\ [\sigma_k, \gamma_2\sigma_k] & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_2\sigma_k, \gamma_3\sigma_k] & \text{if } \rho_k < \eta_1. \end{cases}$     (3.13)
   Increment $k$ by one and go to step 2 if $\rho_k \ge \eta_1$ or to step 3 otherwise.
```

3.2.3 Convergence and Complexity Analysis

The tolerance condition (3.4) is a key condition in the design of ARIG. In fact, this condition ensures that at each iteration k,

$$\|\nabla f(x_k)\| \le \|\nabla f(x_k) - \nabla f(x_k, \omega_g^k)\| + \|\nabla f(x_k, \omega_g^k)\| \le (1+\omega_g^k)\|\nabla f(x_k, \omega_g^k)\|.$$

Therefore, when the termination occurs, $\|\nabla f(x_k, \omega_g^k)\| \le \epsilon/(1+\omega_g^k)$, which implies that $\|\nabla f(x_\epsilon)\| \le \epsilon$ and the first-order critical point x_ϵ satisfies the desired stopping condition.

The following analysis represents the adaptation of the general properties presented by (Birgin et al., 2017) to a second-order model with inexact gradients.

We deduce from (Birgin et al., 2017, Lemma 2.2) an upper bound on the regularization parameter σ_k.

Lemma 3.2.1. *For all $k \geq 0$,*

$$\sigma_k \leq \sigma_{\max} = \max\left[\sigma_0, \frac{\gamma_3(\frac{1}{2}L+1)}{1-\eta_2}\right]. \tag{3.14}$$

Proof. We deduce from (3.12) and (3.8) that

$$|\rho_k - 1| = \frac{|f(x_k+s_k) - \bar{T}_k(s_k)|}{|\bar{T}_k(0) - \bar{T}_k(s_k)|} \leq \frac{\frac{1}{2}L + \omega_g^k \sigma_k}{\sigma_k}.$$

Since we require in step 2 that $\omega_g^k \leq 1/\sigma_k$, we deduce

$$|\rho_k - 1| \leq \frac{\frac{1}{2}L+1}{\sigma_k}.$$

Now assume that

$$\sigma_k \geq \frac{\frac{1}{2}L+1}{1-\eta_2}.$$

We obtain from the two previous inequalities that $|\rho_k - 1| \leq 1-\eta_2$ and thus $\rho_k \geq \eta_2$. Thus, iteration k is very successful in that $\rho_k \geq \eta_2$ and $\sigma_{k+1} \leq \sigma_k$. As a consequence, the mechanism of the algorithm ensures that (3.14) holds. $\square$

We now bound the number of unsuccessful iterations in terms of the number of successful iterations.

Lemma 3.2.2 (Birgin et al., 2017, Lemma 2.4)**.** *For all $k \geq 0$,*

$$k \leq |S_k|\left(1 + \frac{|\log\gamma_1|}{\log\gamma_2}\right) + \frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right). \tag{3.15}$$

where $S_k = \{0 \leq j \leq k \mid \rho_j \geq \eta_1\}$ denotes the set of "successful" iterations between 0 and k.

Proof. We denote by U_k the complement of S_k in $\{1, ..., k\}$, i.e., the index set of unsuccessful iterations between 0 and k. The regularization parameter update (3.13) gives that, for each $k \geq 0$,

$$\gamma_1\sigma_j \leq \max(\gamma_1\sigma_j, \sigma_{\min}) \leq \sigma_{j+1}, \quad j \in S_k, \quad \text{and} \quad \gamma_2\sigma_j \leq \sigma_{j+1}, \quad j \in U_k.$$

Thus we deduce inductively that

$$\sigma_0\gamma_1^{|S_k|}\gamma_2^{|U_k|} \leq \sigma_k.$$

Therefore, using 3.2.1, we obtain

$$|S_k|\log\gamma_1 + |U_k|\log\gamma_2 \leq \log\left(\frac{\sigma_{\max}}{\sigma_0}\right),$$

which then implies that

$$|U_k| \leq -|S_k|\frac{\log\gamma_1}{\log\gamma_2} + \frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right),$$

because $\gamma_2 > 1$. The desired result then follows from the equality $k = |S_k| + |U_k|$ and the inequality $\gamma_1 < 1$ given by (3.11).

□

Using all the above results, we are now in position to state our main evaluation complexity result.

Theorem 3.2.3. *Let 1, 2 and 3 be satisfied. Assume $\omega_g^k \leq 1/\sigma_k$ for all $k \geq 0$. Then, given $\epsilon > 0$, Algorithm 2 needs at most*

$$\left\lfloor \kappa_s \frac{f(x_0) - f_{\text{low}}}{\epsilon^2} \right\rfloor$$

successful iterations (each involving one evaluation of f and its approximate derivative) and at most

$$\left\lfloor \kappa_s \frac{f(x_0) - f_{\text{low}}}{\epsilon^2} \right\rfloor \left(1 + \frac{|\log\gamma_1|}{\log\gamma_2}\right) + \frac{1}{\log\gamma_2}\log\left(\frac{\sigma_{\max}}{\sigma_0}\right)$$

iterations in total to produce an iterate x_ϵ such that $\|\nabla f(x_\epsilon)\| \leq \epsilon$, where $\sigma_{\max}$ is given by 3.2.1 and where

$$\kappa_s = \frac{(1+\sigma_{\max})^2}{\eta_1\sigma_{\min}}.$$

Proof. At each successful iteration, we have

$$\begin{aligned} f(x_k) - f(x_k + s_k) &\geq \eta_1(\bar{T}_k(0) - \bar{T}_k(s_k)) \\ &\geq \frac{\eta_1}{\sigma_k}\|\nabla f(x_k, \omega_g^k)\|^2 \\ &\geq \frac{\eta_1\sigma_{\min}}{(1+\sigma_{\max})^2}\epsilon^2 \end{aligned}$$

where we used (3.12) and the fact that before termination

$$\|\nabla f(x_k, \omega_g^k)\| \geq \frac{1}{1+\omega_g^k}\epsilon \geq \frac{1}{1+\frac{1}{\sigma_k}}\epsilon \geq \frac{\sigma_k}{1+\sigma_k}\epsilon \geq \frac{\sigma_{\min}}{1+\sigma_{\max}}\epsilon.$$

Thus we deduce that as long as termination does not occur,

$$f(x_0) - f(x_{k+1}) = \sum_{j \in S_k} [f(x_j) - f(x_j + s_j)] \geq \frac{|S_k|}{\kappa_s}\epsilon^2, \tag{3.16}$$

from which the desired bound on the number of successful iterations follows. 3.2.2 is then invoked to compute the upper bound on the total of iterations.

□

3.3 Stochastic adaptive regularization with dynamic sampling

Although ARIG enjoys interesting theoretical guarantees, it is not adapted to the context of machine learning, where we do not have access to the exact values of the function either. Therefore, we propose to build upon ARIG in order to design a stochastic first-order optimization with adaptive regularization.

3.3.1 Adaptive Regularization with Inexact Gradient and Function values (ARIGAF)

Let us assume that we do not have access to direct evaluations of $f(x)$, but we can obtain an approximation of $f(x)$, $f(x_k, \omega_f^k)$, with a user-specified error threshold ω_f^k, such that

$$|f(x, \omega_f^k) - f(x, 0)| \leq \omega_f^k, \quad \text{with} \quad f(x, 0) = f(x). \tag{3.17}$$

We redefine the approximate Taylor series (3.7) using the inexact gradient and function values:

$$\bar{T}_k(s) = f(x, \omega_f^k) + \nabla f(x_k, \omega_g^k)^\top s. \tag{3.18}$$

If the approximations of $f(x_k)$ and $f(x_k + s_k)$ are not sufficiently accurate, then the decrease rate ρ_k defined in (3.12) looses its meaning, since it is supposed to quantify the ratio of the ***actual*** *decrease* of the function value to the *predicted decrease* by the model.

Following the same steps as the analysis in (Conn et al., 2000), we suppose two values of the error threshold ω_f^k and $\hat{\omega}_f^k$, such that,

$$|f(x_k, \omega_f^k) - f(x_k)| \leq \omega_f^k, \tag{3.19}$$

and

$$|f(x_k + s_k, \hat{\omega}_f^k) - f(x_k + s_k)| \leq \hat{\omega}_f^k. \tag{3.20}$$

We also suppose a constant $\eta_0 < \frac{1}{2}\eta_1$ and we require that

$$\max(\omega_f^k, \hat{\omega}_f^k) \leq \eta_0 \left[\bar{T}_k(0) - \bar{T}_k(s_k)\right]. \tag{3.21}$$

Using (3.19) and (3.21), we deduce that

$$f(x_k, \omega_f^k) - f(x_k) \leq |f(x_k, \omega_f^k) - f(x_k)| \leq \omega_f^k \leq \eta_0 \left[\bar{T}_k(0) - \bar{T}_k(s_k)\right]. \tag{3.22}$$

Using (3.20) and (3.21), we have that

$$\begin{aligned} f(x_k + s_k, \hat{\omega}_f^k) - f(x_k + s_k) &\leq |f(x_k + s_k, \hat{\omega}_f^k) - f(x_k + s_k)|, \quad (3.23) \\ &\leq \hat{\omega}_f^k \leq \eta_0[\bar{T}_k(0) - \bar{T}_k(s_k)]. \quad (3.24) \end{aligned}$$

From (3.22) and (3.23), we have that

$$\beta_k := \frac{\left[f(x_k, \omega_f^k) - f(x_k)\right] - \left[f(x_k + s_k, \hat{\omega}_f^k) - f(x_k + s_k)\right]}{\bar{T}_k(0) - \bar{T}_k(s_k)} \leq \frac{\omega_f^k + \hat{\omega}_f^k}{\bar{T}_k(0) - \bar{T}_k(s_k)} \leq 2\eta_0. \tag{3.25}$$

Notice that in this case, we have

$$\bar{\rho}_k := \frac{f(x_k, \omega_f^k) - f(x_k + s_k, \hat{\omega}_f^k)}{\bar{T}_k(0) - \bar{T}_k(s_k)} = \frac{f(x_k) - f(x_k + s_k)}{\bar{T}_k(0) - \bar{T}_k(s_k)} + \beta_k = \rho_k + \beta_k. \tag{3.26}$$

Combining (3.25) and (3.26), we obtain the following implication:

$$\bar{\rho}_k \geq \eta_1 \implies \rho_k \geq \eta_1 - 2\eta_0 \geq \bar{\eta}_1 > 0, \tag{3.27}$$

where $\bar{\eta}_1$ is by definition equal to $\eta_1 - 2\eta_0$.

This result is very strong because it means that whenever condition (3.21) is satisfied, a sufficient decrease in the *stochastic* function, i.e., $\bar{\rho}_k \geq \eta_1$, implies that the *achieved* decrease in the exact function value is at least a fraction $\bar{\eta}_1$ of the decrease predicted by the model. It also means that, as long as we can guarantee that condition (3.21) is satisfied at each iteration, our convergence and complexity analysis from the previous section remains valid.

However, incorporating condition (3.21) in our model is not straightforward, because we cannot obtain the value of $\bar{T}_k(s_k)$ before that of $f(x_k, \omega_f^k)$. Therefore, a backtracking strategy will be used.

We propose a new version of ARIG that incorporates inexact function values satisfying the error condition (3.17). We call this algorithm ARIGAF for Adaptive Regularization with Inexact GrAdient and Function values. In particular, ARIGAF ensures that condition (3.21) is satisfied, which means that the convergence and complexity results obtained in Section 3.2.3 remain valid for ARIGAF. Therefore, this algorithm represents the successful adaptation of ARIG to the stochastic setting, while maintaining the convergence guarantees that ARIG comes with. ARIGAF is described in Algorithm 3.

3.3.2 Stochastic ARIG with adaptive sampling

To use a stochastic version of ARIG in machine learning, we need to compute a gradient approximation $g(x_k, \xi_k)$ for mini-batch ξ_k,

$$g(x_k, \xi_k) = \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k, \xi_{k,i}), \tag{3.31}$$

that satisfies

$$\|g(x_k, \xi_k) - \nabla f(x_k)\| \leq \omega_g^k \|g(x_k, \xi_k)\|, \tag{3.32}$$

where ω_g^k is the iteration-dependent error-threshold.

Notice that condition (3.32) is similar to the *norm test*, as defined in Section 2.3.2 of the literature review. Therefore, we can use the adaptive sampling strategy proposed by (Byrd et al., 2012) to maintain this condition satisfied during the optimization. Let us recall their

Algorithm 3 Adaptive Regularization with Inexact Gradient and Function values - ARIGAF

```
Require: $x_0 \in \mathbb{R}^n$
1: Choose the accuracy level $\epsilon > 0$, the initial regularization parameter $\sigma_0 > 0$, the initial
   function error threshold $\omega_f^0$, and the constants $\eta_0, \eta_1, \eta_2, \gamma_1, \gamma_2, \gamma_3$ and $\sigma_{\min}$ such that

   $$\sigma_{\min} \in (0,\, \sigma_0],\ 0 < \eta_1 \le \eta_2 < 1,\ 0 < \eta_0 < \frac{1}{2}\eta_1 \quad \text{and} \quad 0 < \gamma_1 < 1 < \gamma_2 < \gamma_3. \tag{3.28}$$

   Set $k = 0$.
2: Compute $f(x_k, \omega_f^k)$ such that (3.17) holds.
3: Choose $\omega_g^k$ such that $0 < \omega_g^k \le 1/\sigma_k$ and compute $\nabla f(x_k, \omega_g^k)$ such that (3.4) holds. If
   $\|\nabla f(x_k, \omega_g^k)\| \le \epsilon/(1 + \omega_g^k)$, terminate with the approximate solution $x_\epsilon = x_k$.
4: Compute the step $s_k = -\frac{1}{\sigma_k}\nabla f(x_k, \omega_g^k)$.
5: If $\omega_f^k$ does not satisfy (3.21), decrease the value of $\omega_f^k$ and go back to step 2. Otherwise,
   compute $\hat{\omega}_f^k$ such that (3.21) holds.
6: Evaluate $f(x_k + s_k, \hat{\omega}_f^k)$ and define

   $$\bar{\rho}_k = \frac{f(x_k, \omega_f^k) - f(x_k + s_k, \hat{\omega}_f^k)}{\bar{T}_k(0) - \bar{T}_k(s_k)} = \frac{f(x_k, \omega_f^k) - f(x_k + s_k, \hat{\omega}_f^k)}{\frac{1}{\sigma_k}\|\nabla f(x_k, \omega_g^k)\|^2}. \tag{3.29}$$

7: If $\bar{\rho}_k \ge \eta_1$, then define $x_{k+1} = x_k + s_k$ and $\omega_f^{k+1} = \hat{\omega}_f^k$. Otherwise, define $x_{k+1} = x_k$ and
   $\omega_f^{k+1} = \omega_f^k$.
8: Set
   $$\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1\sigma_k), \sigma_k] & \text{if } \bar{\rho}_k \ge \eta_2, \\ [\sigma_k, \gamma_2\sigma_k] & \text{if } \bar{\rho}_k \in [\eta_1, \eta_2), \\ [\gamma_2\sigma_k, \gamma_3\sigma_k] & \text{if } \bar{\rho}_k < \eta_1. \end{cases} \tag{3.30}$$

   Increment $k$ by one and go to step 2 if $\bar{\rho}_k \ge \eta_1$ or to step 3 otherwise.
```

adaptive sampling strategy.

Since ω_g^k should verify $0 < \omega_g^k \leq 1/\sigma_k$, let us fix it $\omega_g^k = 1/\sigma_k$. This choice is natural since a lower error-threshold would require the use of more samples to compute the gradient estimate $g(x_k, \xi_k)$.

Since $g(x_k, \xi_k)$ is obtained as a sample average (3.31), it is an unbiased estimate of $\nabla f(x_k)$. Therefore,

$$\mathbb{E}_{\xi_k}[\|g(x_k, \xi_k) - \nabla f(x_k)\|^2] = \|\mathbb{V}_{\xi_k}[g(x_k, \xi_k)]\|_1, \tag{3.33}$$

where $\mathbb{V}_{\xi_k}[g(x_k, \xi_k)]$ is a vector of the same length as x_k. Then, as in (Freund and Walpole, 1971, page 183), we have

$$\mathbb{V}_{\xi_k}[g(x_k, \xi_k)] = \frac{\mathbb{V}_{i \in \xi_k}(\nabla f_i(x_k))}{m_k} \frac{(N - m_k)}{N - 1}. \tag{3.34}$$

The direct computation of the sample variance $\mathbb{V}_{i \in \xi_k}(\nabla f_i(x_k))$ is expensive. Hence, (Byrd et al., 2012) suggested to use the following variance estimate:

$$\mathbb{V}_{i \in \xi_k}(\nabla f_i(x_k)) \approx \frac{1}{m_k - 1} \sum_{i \in \xi_k} (\nabla f_i(x_k) - g(x_k, \xi_k))^2, \tag{3.35}$$

where the square is applied element wise.

It follows from (3.33) and (3.34) that

$$\mathbb{E}_{\xi_k}\left[\|g(x_k, \xi_k) - \nabla f(x_k)\|^2\right] = \frac{\|\mathbb{V}_{i \in \xi_k}(\nabla f_i(x_k))\|_1}{m_k} \frac{(N - m_k)}{N - 1}. \tag{3.36}$$

In the context of large-scale optimization, we can take the limit $N \to \infty$. Thus, inequality (3.32) implies

$$\frac{\|\mathbb{V}_{i \in \xi_k}(\nabla f_i(x_k))\|_1}{m_k} \leq \frac{1}{\sigma_k^2} \|g(x_k, \xi_k)\|^2. \tag{3.37}$$

Finally, the strategy that we will use to obtain a gradient estimate $g(x_k, \xi_k)$ that satisfies condition (3.32), with $\omega_g^k = 1/\sigma_k$, is the following:

- At iteration k, we sample a new batch of size $m_k = m_{k-1}$, then we compute the sample variance and mini-batch gradient in (3.37).

- If condition (3.37) holds, we use mini-batch m_k in iteration k. Otherwise, we define the new batch size $\tilde{m}_k$ as follows:

$$\tilde{m}_k = \left\lceil \frac{\sigma_k^2 \|\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k))\|_1}{\|g(x_k,\xi_k)\|^2} \right\rceil. \tag{3.38}$$

It is important to note that this strategy relies on the assumption that the change in the batch size is gradual. Therefore, we assume that

$$\|\mathbb{V}_{i\in\tilde{\xi}_k}\left(\nabla f_i(x_k)\right)\|_1 \approx \|\mathbb{V}_{i\in\xi_k}(\nabla f_i(x_k))\|_1 \quad \text{and} \quad \|g(x_k,\tilde{\xi}_k)\| \approx \|g(x_k,\xi_k)\|, \tag{3.39}$$

where $\tilde{\xi}_k$ corresponds to $\tilde{m}_k$.

We call the resulting algorithm ARIG+. It represents the combination of the stochastic version of ARIG with adaptive sampling. Algorithm 4 describes ARIG+.

Numerical experiments

We use ARIG+ to solve the logistic regression problem for the MNIST dataset (LeCun et al., 2010). It is a dataset of handwritten digits. The task consists of predicting the digit given its black-and-white handwritten version. The training set contains $60,000$ examples, and the test set contains $10,000$ examples. We use a grid-search to determine the best set of hyperparameters for ARIG+ and the best learning rate for SGD. We run the two algorithms for 10 epochs.

Figure (3.1) shows the evolution of the training loss and validation accuracy for ARIG+ and SGD. We notice that the two algorithms have on par performance for the training loss. However, ARIG+ has a better validation accuracy than SGD.

Figure 3.2 shows the evolution of the percentage batch size. We notice that the batch size values stay reasonable throughout the optimization.

Figure 3.3 shows the evolution of the step size. Its value oscillates and does not converge to zero.

Algorithm 4 Stochastic ARIG with adaptive sampling - ARIG+

```
Require: $x_0 \in \mathbb{R}^n$
 1: Choose the initial regularization parameter $\sigma_0 > 0$, the initial batch size $m_0$, the total
    number of epochs $N_{\text{epochs}}$, and the constants $\eta_1, \eta_2, \gamma_1, \gamma_2, \gamma_3$ and $\sigma_{\min}$ such that
        $$\sigma_{\min} \in (0, \sigma_0], \; 0 < \eta_1 \leq \eta_2 < 1, \quad \text{and} \quad 0 < \gamma_1 < 1 < \gamma_2 < \gamma_3. \tag{3.40}$$
 2: Set $n = 1$ and $k = 0$.
 3: Sample $\xi_k$. Compute $g(x_k, \xi_k)$ and $\|V_{i\in\xi_k}(\nabla f_i(x_k))\|_1$.
 4: if condition (3.37) holds then
 5:     Go to step 10.
 6: else
 7:     Set
            $$m_k = \left\lceil \frac{\sigma_k^2 \|V_{i\in\xi_k}(\nabla f_i(x_k))\|_1}{\|g(x_k, \xi_k)\|^2} \right\rceil$$
 8:     Sample a new batch $\xi_k$ of new size $m_k$ and compute $g(x_k, \xi_k)$.
 9: end if
10: Compute the step $s_k = -\frac{1}{\sigma_k} g(x_k, \xi_k)$.
11: Evaluate $f(x_k + s_k, \xi_k)$ and define
        $$\bar{\rho}_k = \frac{f(x_k, \xi_k) - f(x_k + s_k, \xi_k)}{\bar{T}_k(0) - \bar{T}_k(s_k)} = \frac{f(x_k, \xi_k) - f(x_k + s_k, \xi_k)}{\frac{1}{\sigma_k}\|g(x_k, \xi_k)\|^2}. \tag{3.41}$$
12: If $\bar{\rho}_k \geq \eta_1$, then define $x_{k+1} = x_k + s_k$. Otherwise, define $x_{k+1} = x_k$.
13: Set
        $$\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1\sigma_k), \sigma_k] & \text{if } \bar{\rho}_k \geq \eta_2, \\ [\sigma_k, \gamma_2\sigma_k] & \text{if } \bar{\rho}_k \in [\eta_1, \eta_2), \\ [\gamma_2\sigma_k, \gamma_3\sigma_k] & \text{if } \bar{\rho}_k < \eta_1. \end{cases} \tag{3.42}$$
14: Set $m_{k+1} = m_k$.
15: if the epoch is completed and $n = N_{\text{epochs}}$ then
16:     Terminate with the approximate solution $x_{k+1}$.
17: else if the epoch is completed and $n < N_{\text{epochs}}$ then
18:     Increment $n$ by one, shuffle the dataset and increment $k$ by one. Go to step 3 if $\bar{\rho}_k \geq \eta_1$
        or to step 10 otherwise.
19: else
20:     Increment $k$ by one and go to step 3 if $\bar{\rho}_k \geq \eta_1$ or to step 10 otherwise.
21: end if
```

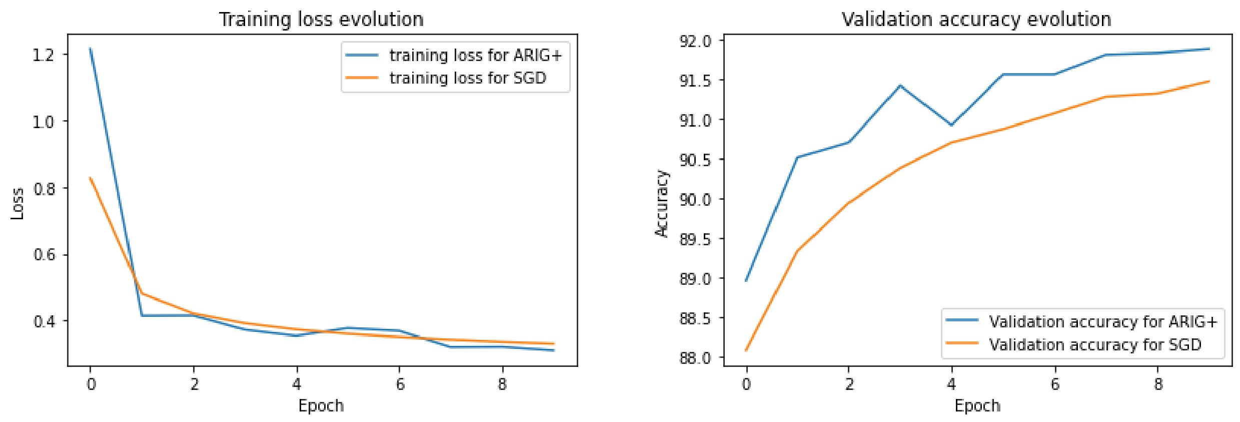

Figure 3.1 Evolution of the training loss (left) and the validation accuracy (right) for SGD and ARIG+, solving a logistic regression problem on MNIST.

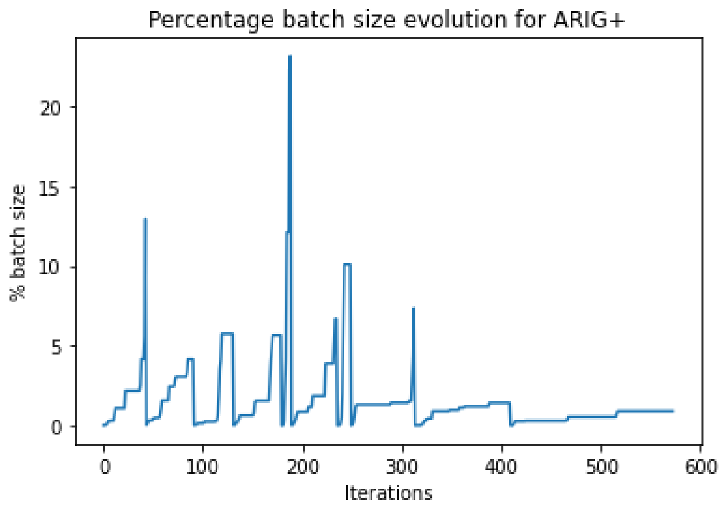

Figure 3.2 Evolution of the percentage batch size for ARIG+ on MNIST.

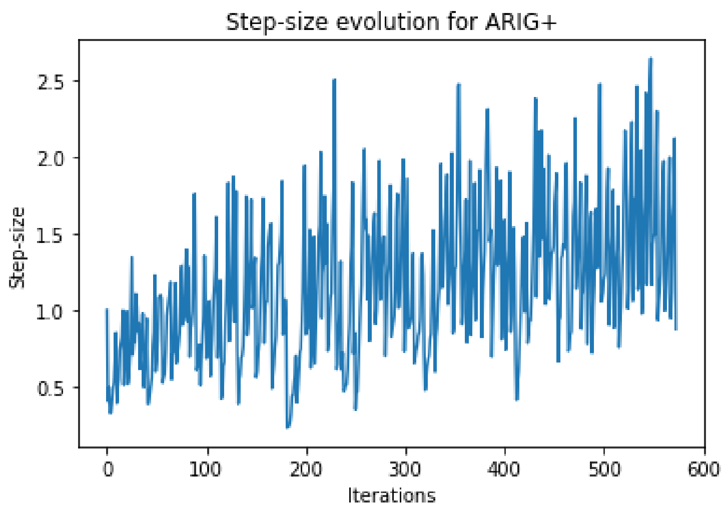

Figure 3.3 Evolution of the step size for ARIG+ on MNIST.

Theoretically speaking, for our algorithm to converge, either the variance of the gradient estimates or the step size has to converge to zero (Paquette, 2020). For ARIG+, the step size does not converge to zero. Moreover, the percentage batch size would need to be close to 100% for the gradient variance to converge to zero. This discussion motivates our work in the Section 3.4, where we propose a new algorithm that leverages both adaptive regularization and adaptive sampling, but not simultaneously. The new algorithm is guaranteed to globally converge because we choose a step size that converges to zero in the second phase of the optimization.

3.4 Stochastic adaptive regularization with dynamic sampling and convergence diagnostics

In this section, we present a novel algorithm that incorporates both adaptive regularization and adaptive sampling by leveraging the two-phase nature of stochastic optimization. This algorithm is an improved version of ARIG+, since we use a step size sequence that converges to zero in the second phase of the optimization. This ensures the overall convergence of our new algorithm.

3.4.1 Statistical testing for stochastic optimization

The two-phase nature of stochastic optimization procedures was explored even before (Murata, 1998). In fact, stochastic optimization algorithms exhibit two phases: a transient phase and a stationary one. The transient phase is characterized by the convergence of the algorithm to a region of interest that contains potentially a good local or global optimum. In the stationary phase, the algorithm oscillates around that optimum in the region of interest.

Several works aimed at deriving a stationary condition to indicate when stochastic procedures leave the transient phase and enter the stationary phase. Some of the early works in this line of research are Pflug's works (Pflug, 1983, 1990, 1992). In particular, Pflug's procedure was introduced in (Pflug, 1990), and it consists of keeping a running average of the inner product of successive gradients in order to detect the end of the transient phase. Pflug's idea is simple: when the running average becomes negative, that suggests that the stochastic gradients are pointing to different directions. This should signal that the algorithm entered the stationary phase where it oscillates around a local optimum. More recently, (Chee and Toulis, 2018) developed a statistical diagnostic test in order to detect when SGD enters the stationary phase, by combining the statistical test from (Pflug, 1990) with the SGD update. They presented theoretical arguments and practical experiments to prove that the test activation occurs in the phase-transition for SGD. Algorithm 5 describes Pflug diagnostic proposed by (Chee and Toulis, 2018) for convergence of SGD, where $g(x_k)$ represents an approximation of the full gradient $\nabla f(x_k)$. Note that in this case, the number of samples in ξ_k is exactly 1.

The authors prove that the convergence diagnostic in Algorithm 5 satisfies $\mathbb{E}(S_k - S_{k-1}) < 0$ as $k \to +\infty$. This implies that the algorithm terminates almost surely.

Algorithm 5 Pflug diagnostic for convergence of SGD

```
Require: $x_0 \in \mathbb{R}^n$
 1: Choose $\gamma > 0$ and burn-in $> 0$.
 2: Set $S_0 = 0$
 3: Compute $g(x_0)$ and make step $x_1 = x_0 - \gamma g(x_0)$.
 4: for $k = 1, 2, \ldots$ do
 5:    Compute $g(x_k)$ and make step $x_{k+1} = x_k - \gamma g(x_k)$.
 6:    Define $S_k = S_{k-1} + g(x_k)^\top g(x_{k-1})$.
 7:    if $k >$ burn-in and $S_k < 0$ then
 8:       return k
 9:    end if
10: end for
```

Figure 3.4 from (Chee and Toulis, 2018, Figure 1, page 12) shows the changes of Pflug diagnostic, $\mathbb{E}(S_k - S_{k-1})$, within different regions of the optimization space. The red star represents the true optimum. The blue polygon represents the convergence region of SGD, i.e., the region where iterates oscillate around the true optimum for 95% of the time, computed over 1000 simulations. The darkest-shaded area represents the area where Pflug diagnostic is decreased in expectation. As the figure shows, Pflug diagnostic represents a good approximation of the convergence region of SGD.

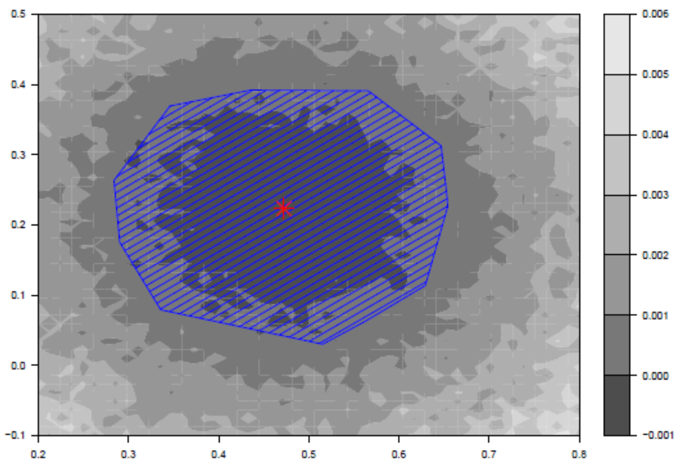

Figure 3.4 Expected increase or decrease (color legend on the right) of Pflug diagnostic $\mathbb{E}(S_k - S_{k-1})$. Red star: the true optimum value. Shaded area in the center: region where Pflug diagnostic decreases in expectation. Polygon around shaded area: convergence region of SGD. Figure taken from (Chee and Toulis, 2018, Figure 1, page 12).

3.4.2 Adaptive regularization and sampling using Pflug diagnostic

To harness Pflug diagnostic in our new algorithm, we discuss the bias-variance trade-off in machine learning. Generally speaking, the bias of a model represents the extent to which it approximates the true optimum. The variance of a model represents the amount of variation in the solution if the training set changes. The generalization error in machine learning represents the sum of the bias and variance. Therefore, there is a trade-off between the two quantities.

Now notice that

- in the transient phase, we would like to reduce the bias of the model by converging to a promising region of interest. Therefore, we propose to use adaptive regularization in order to adapt the step size automatically, taking into account the information on the decrease rate of the function ρ_k. We use a fixed batch size during this phase.
- in the stationary phase, we would like to reduce the variance of the gradient estimates. Therefore, we propose a variance reduction technique that consists of adaptive sampling. While the batch size is adjusted automatically during this phase, we choose a step size sequence that converges to zero. This ensures the global convergence of our algorithm.

Algorithm 6 summarizes this method.

Algorithm 6 Adaptive Regularization and Sampling using Pflug diagnostic - ARAS

```
Require: $x_0 \in \mathbb{R}^n$
 1: Choose the initial regularization parameter $\sigma_0 > 0$, the initial batch size $m_0$, the
    maximum batch size $m_{\max}$, the total number of epochs $N_{\text{epochs}}$ and the constants
    burn-in $> 0, \eta, \gamma_1, \gamma_2,$ and $\sigma_{\min}$ such that
        $$\sigma_{\min} \in (0,\ \sigma_0],\ 0 < \eta < 1 \quad \text{and} \quad 0 < \gamma_1 < 1 < \gamma_2. \tag{3.43}$$
 2: Set $n = 1$, $k = 0$, $t = 2$, $S_0 = 0$, and Transient = True.
 3: if Transient is True then
 4:     Sample $\xi_k$ and compute $g(x_k, \xi_k)$.
 5:     Compute $s_k = -\frac{1}{\sigma_k} g(x_k, \xi_k)$ and define $x_{k+1} = x_k + s_k$.
 6:     Evaluate $f(x_{k+1}, \xi_k)$ and $g(x_{k+1}, \xi_k)$, and define
        $$\bar{\rho}_k = \frac{f(x_k, \xi_k) - f(x_{k+1}, \xi_k)}{\bar{T}_k(0) - \bar{T}_k(s_k)} = \frac{f(x_k, \xi_k) - f(x_{k+1}, \xi_k)}{\frac{1}{\sigma_k}\|g(x_k, \xi_k)\|^2}. \tag{3.44}$$
 7:     Set
        $$\begin{cases} \sigma_{k+1} \in [\max(\sigma_{\min}, \gamma_1 \sigma_k), \sigma_k] & \text{if } \bar{\rho}_k \geq \eta, \\ \sigma_{k+1} \in [\sigma_k, \gamma_2 \sigma_k] & \text{if } \bar{\rho}_k < \eta. \end{cases} \tag{3.45}$$
 8:     Compute $S_{k+1} = S_k + g(x_{k+1}, \xi_k)^\top g(x_k, \xi_k)$ and set $m_{k+1} = m_k$.
 9:     If ($k >$ burn-in and $S_{k+1} < 0$), then set Transient = False.
10: else
11:     Sample $\xi_k$. Compute $g(x_k, \xi_k)$ and $\|V_{i \in \xi_k}(\nabla f_i(x_k))\|_1$.
12:     if condition (3.37) holds then
13:         Go to step 18.
14:     else
15:         Set
            $$m_k = min\left(\left\lceil \frac{\sigma_k^2 \|V_{i \in \xi_k}(\nabla f_i(x_k))\|_1}{\|g(x_k, \xi_k)\|^2} \right\rceil, m_{\max}\right)$$
16:         Sample a new $\xi_k$ of new size $m_k$ and compute $g(x_k, \xi_k)$.
17:     end if
18:     Compute the step $s_k = -\frac{1}{\sigma_k} g(x_k, \xi_k)$ and define $x_{k+1} = x_k + s_k$.
19:     Set $m_{k+1} = m_k$, $\sigma_{k+1} = \sigma_k \frac{t}{t-1}$, and increment t by one.
20: end if
21: if the epoch is completed and $n = N_{\text{epochs}}$ then
22:     Terminate with the approximate solution $x_{k+1}$.
23: else if the epoch is completed and $n < N_{\text{epochs}}$ then
24:     Increment $n$ by one, shuffle the dataset, increment $k$ by one and go to step 3.
25: else
26:     Increment $k$ by one and go to step 3.
27: end if
```

We call our algorithm ARAS for Adaptive Regularization and Adaptive Sampling algorithm. Note that we reduce the number of parameters that were initially introduced in ARIG by eliminating γ_3 and η_2, which makes the algorithm more practical. Inspired by (Curtis et al., 2019), we do not require for the decrease rate to be sufficient to make the step. Instead, we make the step in all cases. We also use a maximum batch size constant, $m_{\max}$, to ensure that the batch size values stay reasonable.

3.4.3 Experiments

In this section, we empirically study the performance of ARAS in both the convex and non-convex settings. First, we compare the performance of ARAS to that of SGD, SGD with momentum and ARIG+ in the convex setting. The problem of interest is a logistic regression problem. Then, we compare ARAS to its modified version, as well as SGD in the non-convex setting. The problem of interest is a non-convex support vector machine problem, with a sigmoid loss function.

Logistic regression with MNIST

To evaluate the performance of our algorithm in the convex setting, we compare ARAS, ARIG+, SGD and SGD with momentum for solving a logistic regression problem on the MNIST dataset. All parameters were set using a grid-search and the algorithms were trained for 10 epochs each.

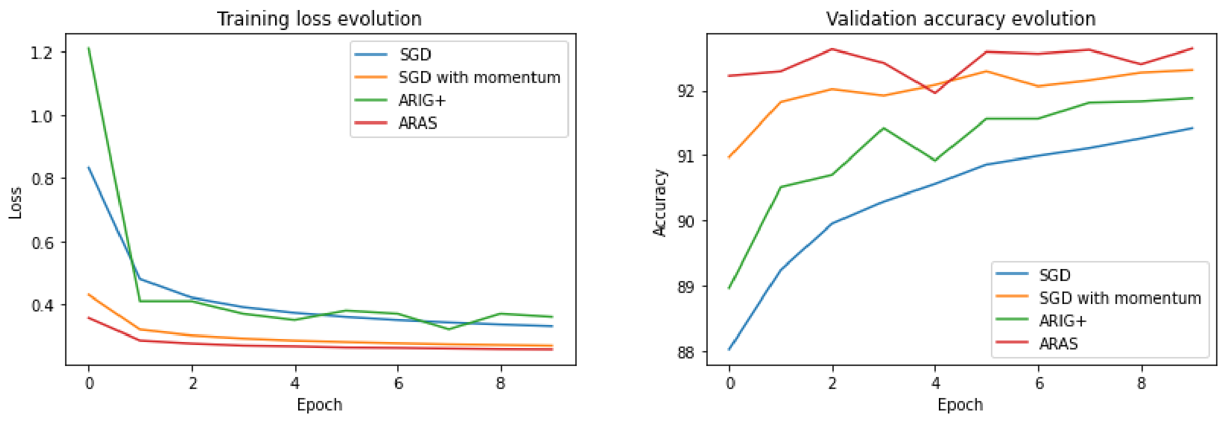

Figure 3.5 Evolution of the training loss (left) and the validation accuracy (right) for ARIG+, ARAS, SGD and SGD with momentum, solving a logistic regression problem on MNIST.

Figure 3.5 shows the performance of the four algorithms. Our algorithm outperforms all three other algorithms on both the training and validation sets. It is worth noting that the transition from the transient phase to the stationary phase was not detected during the training.

Non-convex support vector machine with RCV1

In the non-convex setting, we compare the performance of ARAS and SGD for solving the finite-sum version of a non-convex support vector machine problem, with a sigmoid loss function, a problem that was considered in (Wang et al., 2017):

$$\min_{x\in\mathbb{R}^n} f(x) := \mathbb{E}_{u,v}[1 - \tanh(v\, x^\top u)] + \lambda\|x\|^2, \tag{3.46}$$

where $u \in \mathbb{R}^n$ represents the feature vector, $v \in \{-1, 1\}$ represents the label value, and λ represents the regularization parameter.

The finite-sum version of problem (3.46) can be written as follows:

$$\min_{x\in\mathbb{R}^n} h(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x) + \lambda\|x\|^2, \tag{3.47}$$

where $f_i(x) = 1 - \tanh(v_i\, x^\top u_i)$ for $i \in \{1, \dots, N\}$.

As for the dataset, we use a subset [1] of the dataset Reuters Corpus Volume I (RCV1) (Lewis et al., 2004), which is a collection of manually categorized newswire stories. The subset contains 9625 stories with 29992 distinct words. There are four categories to which the stories belong: "GCAT", with 2901 stories, "MCAT", with 2638 stories, "C15", with 2022 stories, and "ECAT", with 2064 stories. We combine the two categories "MCAT" and "ECAT" into one category, to which we assign the label value 1; and the two categories "C15" and "GCAT" into one category, to which we assign the label value -1. Our purpose is to solve this binary classification problem.

We train the algorithms for 10 epochs, each. We compare ARAS to SGD, as well as another version of ARAS in which we force the transition from the transient state to the stationary in the fifth epoch. The results obtained are shown in Figure 3.6.

We notice that the forced transition does not improve the performance of ARAS, especially on the training set. In fact, we tried different values for the epoch in which we force the transition, and it is the fifth epoch that gave the best results. However, the resulting algorithm is still outperformed by the vanilla version of ARAS. This means that it is better in this specific case to trust our statistical diagnostic test in detecting the transition, rather than

[1]We downloaded the used subset from http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

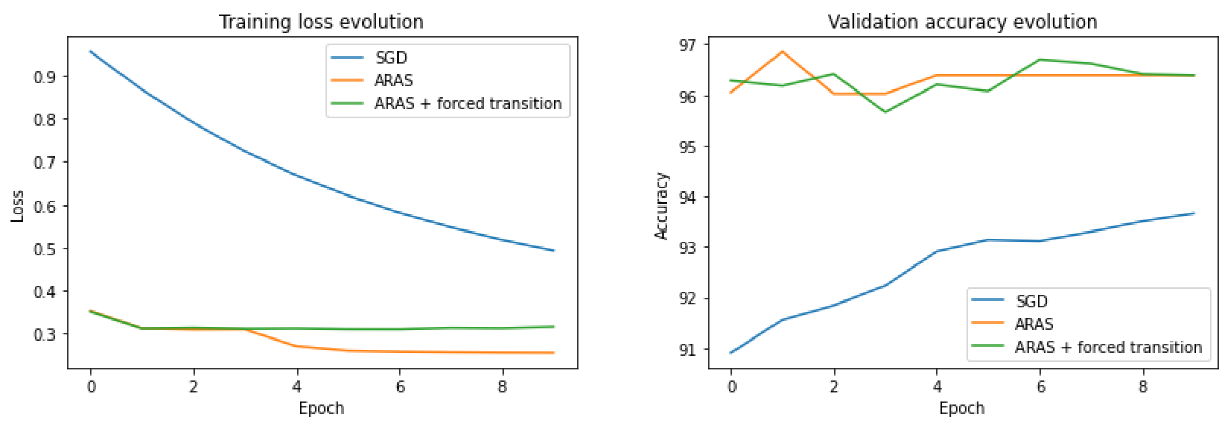

Figure 3.6 Evolution of the training loss (left) and the validation accuracy (right) for SGD, ARAS, and ARAS with a forced transition in the fifth epoch, solving a non-convex SVM problem on RCV1.

forcing it. Moreover, we notice that both versions of ARAS largely outperform SGD on both the training and the test tasks.

3.5 Conclusion

In this chapter, we proposed novel first-order optimization algorithms in the context of machine learning and large-scale optimization. Table 3.1 summarizes the characteristics of these algorithms.

Table 3.1 Comparison of the characteristics of our first-order algorithms.

	ARIG	ARIGAF	ARIG+	ARAS
Stochasticity	✗	✓	✓	✓
Theoretical guarantees	✓	✓	✗	?
Adaptive step size	✓	✓	✓	✓
Adaptive batch size	✗	✗	✓	✓
Practical for machine learning	✗	✗	✓	✓
Conceivable overall convergence	✓	✓	✗	✓

The first algorithm that we proposed, ARIGAF, is an adaptation of ARIG to the stochastic setting that maintains the same theoretical guarantees. However, this algorithm is not practical for solving machine learning problems. Therefore, we proposed ARIG+ that leverages adaptive sampling to satisfy the accuracy condition on the gradient approximation. This second algorithm is practical for machine learning at the cost of losing ARIG's theoretical guarantees. Finally, we proposed ARAS, which solves both the practicality concern that comes with ARIGAF and the overall convergence concern that comes with ARIG+. ARAS was demonstrated empirically to outperform SGD and enjoys a conceivable overall convergence property since the step size converges to zero. Nevertheless, its convergence rate and iteration complexity remain an open question.

CHAPTER 4 STOCHASTIC DAMPED L-BFGS WITH CONTROLLED NORM OF THE HESSIAN APPROXIMATION

In this chapter, we propose a new stochastic damped L-BFGS algorithm (see Section 2.4), where we make use of the bounds on the largest and smallest eigenvalues of the Hessian approximation to monitor its quality. Our algorithm draws heavily from the work of (Wang et al., 2017), who proposed recently a stochastic damped version of L-BFGS for online and finite-sum problems.

4.1 Introduction and motivation

We consider the online optimization problem introduced previously

$$\min_{x\in\mathbb{R}^n} f(x) := \mathbb{E}_\xi[F(x,\xi)], \tag{4.1}$$

where $\xi \in \mathbb{R}^d$ denotes a random variable and $F : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable function, possibly non-convex.

Our algorithm also applies to finite-sum problems, with the alternative definition of f as

$$\min_{x\in\mathbb{R}^n} f(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{4.2}$$

where f_i represents the loss function that corresponds to the i-th element of the dataset and N is the size of the dataset.

As discussed in the literature review, Chapter 2, SGD and its variants are widely used to solve optimization problems (4.1) and (4.2) in machine learning. However, they might not be well-suited for highly non-convex and ill-conditioned problems. An efficient way to solve these issues is to make use of the second-order information. Second-order algorithms are well studied in the deterministic case but there are many areas to explore in the stochastic context. Specifically, the use of damping in L-BFGS is an interesting technique to be leveraged in the stochastic case. (Wang et al., 2017) proposed a stochastic damped L-BFGS (SdLBFGS) algorithm and proved almost sure convergence to a stationary point. However, as mentioned by (Chen et al., 2019), the damping does not prevent the Hessian approximation from being ill-conditioned. Additionally, the overall convergence of SdLBFGS may be heavily affected if the Hessian approximation becomes singular during the optimization process. SdLBFGS does not guarantee that it will not be the case. In order to remedy this issue, (Chen et al., 2019) proposed to combine SdLBFGS with regularized BFGS presented by (Mokhtari and Ribeiro, 2014). We take a different direction to solve this problem.

Our contributions:

- Less restrictive assumptions: we prove that the inverse Hessian approximation is bounded by requiring for the stochastic gradient to be Lipschitz continuous, which is a less restrictive assumption than (Wang et al., 2017), where they require for the stochastic function to be twice differentiable with respect to the parameter vector, and for the norm of its Hessian to be bounded for all parameter and random sampling vectors.
- A new damped L-BFGS: we propose a new version of stochastic damped L-BFGS, that uses an estimation of these bounds to change the update of the inverse Hessian approximation. Different solutions are proposed when ill-conditioning is detected. The algorithm is shown to converge almost surely to a stationary point.
- Choice of the initial inverse Hessian approximation: we propose a new formula of the initial inverse Hessian approximation to make the algorithm more stable.

4.2 Formulation of our method

We assume that we do not have access to full gradients, but we can obtain a stochastic approximation of the gradient via a Stochastic First-order Oracle (SFO). Let $g(x_k, \xi_k)$ denote the gradient approximation obtained at iteration k, where ξ_k corresponds to the subset of samples considered in this iteration, taken from a given set of realizations of the random variable ξ. In addition, $g(x_k, \xi_k)$ satisfies

$$g(x_k, \xi_k) = \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k, \xi_{k,i}), \tag{4.3}$$

where $\xi_{k,i}$ is to the i-th sample of ξ_k in iteration k and m_k is the batch size used in iteration k. Note that we do not assume here that $g(x_k, \xi_{k,i}) = \nabla f_{\xi_{k,i}}(x_k)$.

The Hessian approximation constructed at iteration k and its inverse are denoted B_k and H_k, respectively, such that $H_k = B_k^{-1}$ and B_k is positive definite.

The update rule is the following:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{4.4}$$

where

$$d_k = -H_k g(x_k, \xi_k),$$

and $\alpha_k > 0$ is the step size.

Now, let us define the step and gradient displacement vectors at iteration k

$$s_k = x_{k+1} - x_k, \quad \text{and} \quad y_k = g(x_{k+1}, \xi_k) - g(x_k, \xi_k). \tag{4.5}$$

Following the BFGS conditions, the updated Hessian approximation should verify the secant equation:

$$B_{k+1} s_k = y_k. \tag{4.6}$$

Then, the following curvature condition allows B_{k+1} to be positive definite, which ensures that d_k is a descent direction

$$s_k^\top y_k > 0. \tag{4.7}$$

The inverse Hessian update for BFGS is given by

$$H_{k+1} = (I - \rho_k s_k y_k^\top) H_k (I - \rho_k y_k s_k^\top) + \rho_k s_k s_k^\top, \quad \text{where} \quad \rho_k = \frac{1}{s_k^\top y_k}. \tag{4.8}$$

The initial approximation H_0 is chosen to be positive definite. Recursively, H_{k+1} is positive definite (see, for instance, Fletcher, 1970).

Since it is computationally expensive to store the Hessian approximations and perform matrix-vector products in the context of large-scale optimization, we use the limited-memory version of BFGS. Therefore, the approximation of the Hessian only depends on the last p iterations and an initial H_k^0, chosen to be positive definite. The parameter p is called the memory of L-BFGS. The inverse Hessian update becomes

$$\begin{aligned} H_k = {} & (V_{k-1}^\top \dots V_{k-p}^\top) H_k^0 (V_{k-p} \dots V_{k-1}) \\ & + \rho_{k-p} (V_{k-1}^\top \dots V_{k-p+1}^\top) s_{k-p} s_{k-p}^\top (V_{k-p+1} \dots V_{k-1}) \\ & + \rho_{k-p+1} (V_{k-1}^\top \dots V_{k-p+2}^\top) s_{k-p+1} s_{k-p+1}^\top (V_{k-p+2} \dots V_{k-1}) \\ & \vdots \\ & + \rho_{k-1} s_{k-1} s_{k-1}^\top, \end{aligned} \tag{4.9}$$

where $V_k = (I - \rho_k s_k y_k^\top)$.

When the step length α_k in equation (4.4) is not computed from the Wolfe conditions in the line search (Wolfe, 1969, 1971), there is no guarantee that the curvature condition holds. Therefore, we use the *damping* technique, as defined in (Powell, 1978), to ensure that the inverse Hessian approximation remains sufficiently positive definite during the optimization process. It consists in defining a modified gradient displacement vector $\hat{y}_k$ as

$$\hat{y}_k = \theta_k y_k + (1 - \theta_k) B_k s_k, \tag{4.10}$$

where

$$\theta_k = \begin{cases} 1 & \text{if} \quad s_k^\top y_k \geq \eta s_k^\top B_k s_k \\ (1-\eta) \frac{s_k^\top B_k s_k}{s_k^\top B_k s_k - s_k^\top y_k} & \text{if} \quad s_k^\top y_k < \eta s_k^\top B_k s_k \end{cases}, \tag{4.11}$$

with $\eta \in (0, 1)$. Thus,

$$s_k^\top \hat{y}_k \geq \eta s_k^\top B_k s_k \geq \eta \lambda_{\min}(B_k) \|s_k\|^2 = \frac{\eta}{\lambda_{\max}(H_k)} \|s_k\|^2,$$

and the curvature condition is always verified when using s_k and $\hat{y}_k$. We obtain the damped BFGS update :

$$H_{k+1} = (I - \hat{\rho}_k s_k \hat{y}_k^\top) H_k (I - \hat{\rho}_k \hat{y}_k s_k^\top) + \hat{\rho}_k s_k s_k^\top, \quad \text{where} \quad \hat{\rho}_k = \frac{1}{s_k^\top \hat{y}_k}. \tag{4.12}$$

The L-BFGS update is obtained by applying p times the damped BFGS update to H_k^0.

4.3 A new stochastic damped L-BFGS with controlled Hessian norm

Our working assumptions are the same as Assumptions 1-3 from Chapter 3. Namely,

Assumption 4. *The function f is bounded below on $\mathbb{R}^n$, i.e., there exists κ_{low} such that $f(x) \geq \kappa_{low}$ for all $x \in \mathbb{R}^n$.*

Assumption 5. *The function f is continuously differentiable over $\mathbb{R}^n$.*

Assumption 6. *The gradient of f is Lipschitz continuous, i.e., there exists $L > 0$ such that for all x, $y \in \mathbb{R}^n$, $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$.*

In Section 4.3.1, we propose a new definition for $\hat{y}$. Then, we prove that the maximum and minimum eigenvalues of the updated inverse Hessian, using the new $\hat{y}$, are bounded above and below, respectively. In Section 4.3.2, we present our algorithm, that we call Stochastic Damped L-BFGS with Controlled Hessian Norm (SdLBFGS-CHN). Finally, in Section 4.3.3, we propose a variance reduced version of SdLBFGS-CHN.

4.3.1 Bounded inverse Hessian approximation

Our purpose is not only to prove that the inverse Hessian approximation, H_k, stays bounded all along the optimization process, but also to obtain bounds that we are able to compute, or at least compute their approximations. Definition (4.10) of $\hat{y}_k$ would result in an expression of the bounds over H_{k+1} that depends on the maximum and minimum eigenvalues of H_k, which we cannot afford to compute.

Inspired by (Wang et al., 2017), we change the definition of $\hat{y}_k$ in (4.10), and use the following definition instead:

$$\hat{y}_k = \theta_k y_k + (1 - \theta_k)(H^0_{k+1})^{-1} s_k, \tag{4.13}$$

where

$$\theta_k = \begin{cases} 1 & \text{if} \quad s_k^\top y_k \geq \eta s_k^\top (H^0_{k+1})^{-1} s_k \\ (1-\eta)\frac{s_k^\top (H^0_{k+1})^{-1} s_k}{s_k^\top (H^0_{k+1})^{-1} s_k - s_k^\top y_k} & \text{if} \quad s_k^\top y_k < \eta s_k^\top (H^0_{k+1})^{-1} s_k \end{cases}, \tag{4.14}$$

with $\eta \in (0, 1)$.

This new damping strategy ensures that the curvature condition

$$s_k^\top \hat{y}_k \geq \eta s_k^\top (H^0_{k+1})^{-1} s_k \geq \eta \lambda_{\min}(H^0_{k+1})\|s_k\|^2 = \frac{\eta}{\lambda_{\max}(H^0_{k+1})}\|s_k\|^2,$$

is always satisfied since H^0_{k+1} is chosen to be definite positive.

In Lemma 4.3.1, we consider two vectors s and y such that $s^\top y \geq \gamma \|s\|^2$, with $\gamma > 0$. Notice that

$$\gamma\|s\|^2 \leq s^\top y \leq \|s\|\|y\|, \quad \text{and thus} \quad \|s\| \leq \frac{1}{\gamma}\|y\|.$$

We also need $\rho = \frac{1}{s^\top y}$, the inverse of their inner product. It is useful to note that

$$\frac{1}{\|s\|\|y\|} \leq \rho \leq \frac{1}{\gamma}\frac{1}{\|s\|^2}. \tag{4.15}$$

Lemma 4.3.1. *Let s and $y \in \mathbb{R}^n$ such that $s^\top y \geq \gamma\|s\|^2$ with $\gamma > 0$, and such that $\|y\| \leq L_y\|s\|$, with $L_y > 0$. We denote ρ, the inverse of the inner product of these two vectors, i.e. $\rho = \frac{1}{s^\top y}$. Let $A = \mu SS^\top + \rho ss^\top$, where $\mu > 0$, and $S = (I - \rho sy^\top)$. Then,*

$$\lambda_{\min}(A) \geq \min\left(\frac{1}{L_y}, \frac{\mu}{1 + \frac{\mu}{\gamma}L_y^2}\right) > 0, \tag{4.16}$$

and

$$\lambda_{\max}(A) \leq \frac{1}{\gamma} + \max\left(0, \frac{\mu}{\gamma^2}L_y^2 - \frac{\mu}{1 + \frac{\mu}{\gamma}L_y^2}\right). \tag{4.17}$$

Proof. Since A is a real symmetric matrix, the spectral theorem states that its eigenvalues are real and it can be diagonalized by an orthogonal matrix. That means that we can find n orthogonal eigenvectors and n eigenvalues counted with multiplicity.

Consider first the special case where s and y are collinear, i.e. there exists $\theta > 0$ such that $y = \theta s$. Any vector such that $u \in s^\perp$, where $s^\perp = \{x \in \mathbb{R}^n : x^\top s = 0\}$, is an eigenvector of A associated with the eigenvalue μ of multiplicity $n - 1$. Moreover, $s^\top y = \theta\|s\|^2 = \|s\|\|y\|$, $\rho = \frac{1}{\theta\|s\|^2}$ and we have

$$As = \left[\mu\left(I - \rho\theta ss^\top\right)^2 + \rho ss^\top\right]s = \left[\mu\left(1 - \rho\theta\|s\|^2\right)^2 + \rho\|s\|^2\right]s = \rho\|s\|^2 s.$$

Let us call $\lambda = \rho\|s\|^2$, the eigenvalue associated with the eigenvector s. From (4.15) and $\|y\| \leq L_y\|s\|$, we deduce that

$$\frac{1}{L_y} \leq \lambda \leq \frac{1}{\gamma}.$$

In the more general case, suppose that s and y are linearly independent. Any vector u such that $u^\top s = 0 = u^\top y$ satisfies $Au = \mu u$. This provides us with a $(n-2)$-dimensional eigenspace V, associated to the eigenvalue μ of multiplicity $n - 2$. Note that

$$\begin{aligned} As &= \rho\|s\|^2\ (1 + \mu\rho\|y\|^2)s - \rho\|s\|^2\mu y, \\ Ay &= s. \end{aligned}$$

Thus neither s nor y is an eigenvector of A. Now consider an eigenvalue λ associated with an eigenvector u, such that $u \in V^{\perp}$. Since s and y are linearly-independent, we can search for u of the form $u = s + \beta y$ with $\beta > 0$. We obtain the following system of equations

$$\begin{aligned} \rho\|s\|^2 \; (1 + \mu\rho\|y\|^2) + \beta &= \lambda, \\ -\rho\|s\|^2\mu &= \lambda\beta. \end{aligned}$$

We eliminate $\beta = \lambda - \rho\|s\|^2 \; (1 + \mu\rho\|y\|^2)$ and obtain

$$p(\lambda) = 0,$$

where

$$p(\lambda) = \lambda^2 - \lambda\rho\|s\|^2 \; (1 + \mu\rho\|y\|^2) + \rho\|s\|^2\mu.$$

The roots of p must be the two remaining eigenvalues $\lambda_1 \leq \lambda_2$ that we are looking for. In order to obtain (4.16), we need a lower bound on λ_1 whereas to obtain (4.17), we need an upper bound on λ_2.

On the one hand, let l be the tangent to the graph of p at $\lambda = 0$, defined by

$$l(\lambda) = p(0) + p'(0)\lambda = \mu\rho\|s\|^2 - \lambda\rho\|s\|^2 \left(1 + \mu\rho\|y\|^2\right).$$

Its unique root is

$$\bar{\lambda} = \frac{\mu}{1 + \mu\rho\|y\|^2}.$$

From (4.15) and since $\|y\| \leq L_y\|s\|$, we deduce that

$$\bar{\lambda} \geq \frac{\mu}{1 + \frac{\mu}{\gamma}\frac{\|y\|^2}{\|s\|^2}} \geq \frac{\mu}{1 + \frac{\mu}{\gamma}L_y^2}.$$

Since p is convex, it remains above its tangent, and $\bar{\lambda} \leq \lambda_1$.

Finally,

$$\lambda_{\min}(A) \geq \min\left(\tfrac{1}{L_y}, \tfrac{\mu}{1+\frac{\mu}{\gamma}L_y^2}\right) > 0.$$

This proves (4.16).

On the other hand, we note $\Delta = \rho^2\|s\|^4(1 + \mu\rho\|y\|^2)^2 - 4\rho\|s\|^2\mu$. The discriminant Δ must be nonnegative since A is real symmetric, and its eigenvalues are real. We have

$$\lambda_2 = \frac{\rho\|s\|^2(1 + \mu\rho\|y\|^2) + \sqrt{\rho^2\|s\|^4(1 + \mu\rho\|y\|^2)^2 - 4\rho\|s\|^2\mu}}{2}.$$

For any positive a and b such that $a^2 - b > 0$, we have $\sqrt{a^2 - b} \le a - \frac{b}{2a}$. Thus,

$$\lambda_2 \le \rho \|s\|^2 (1 + \mu\rho\|y\|^2) - \frac{\mu}{1 + \mu\rho\|y\|^2}.$$

From (4.15), we deduce that

$$\lambda_2 \le \frac{1}{\gamma} + \frac{\mu}{\gamma^2}\frac{\|y\|^2}{\|s\|^2} - \frac{\mu}{1 + \frac{\mu}{\gamma}\frac{\|y\|^2}{\|s\|^2}}.$$

And since $\|y\| \le L_y \|s\|$, it follows

$$\lambda_2 \le \frac{1}{\gamma} + \frac{\mu}{\gamma^2}L_y^2 - \frac{\mu}{1 + \frac{\mu}{\gamma}L_y^2}.$$

Finally,

$$\lambda_{\max}(A) \le \max\left(\frac{1}{\gamma}, \frac{1}{\gamma} + \frac{\mu}{\gamma^2}L_y^2 - \frac{\mu}{1 + \frac{\mu}{\gamma}L_y^2}\right).$$

This proves (4.17).

$\square$

Let us recall the damped L-BFGS update formula in the stochastic setting. In each iteration, we compute

$$s_k = x_{k+1} - x_k \quad \text{and} \quad y_k = g(x_{k+1}, \xi_k) - g(x_k, \xi_k). \tag{4.18}$$

Notice that we compute the gradient estimates $g(x_{k+1}, \xi_k)$ and $g(x_k, \xi_k)$ using the same mini-batch ξ_k, following the steps of (Schraudolph et al., 2007; Bordes et al., 2009). As mentioned by (Bottou et al., 2018), the use of different mini-batches, ξ_{k+1} and ξ_k, can result in poor curvature estimates. Alternative strategies include decoupling the computation of the step and the Hessian approximation (Bottou et al., 2018), or computing the Hessian approximation using a significant overlap between two consecutive batches (Berahas and Takáč, 2020).

Back to the stochastic damped L-BFGS, we choose the damping parameter θ_k according to (4.14) to ensure that

$$s_k^\top \hat{y}_k \ge \eta s_k^\top (H_{k+1}^0)^{-1} s_k \ge \frac{\eta}{\lambda_{\max}(H_{k+1}^0)}\|s_k\|^2, \tag{4.19}$$

where $B_k = H_k^{-1}$ is the current approximation of the Hessian and $\eta \in (0, 1)$. Then, we apply

p times the damped BFGS update (4.12) to H_k^0:

$$\begin{aligned}
H_{k+1} = {} & (S_k^\top \dots S_{k-p+1}^\top) H_{k+1}^0 (S_{k-p+1} \dots S_k) \\
& + \hat{\rho}_{k-p+1} (S_k^\top \dots S_{k-p+2}^\top) s_{k-p+1} s_{k-p+1}^\top (S_{k-p+2} \dots S_k) \\
& + \hat{\rho}_{k-p+2} (S_k^\top \dots S_{k-p+3}^\top) s_{k-p+2} s_{k-p+2}^\top (S_{k-p+3} \dots S_k) \\
& \vdots \\
& + \hat{\rho}_k s_k s_k^\top,
\end{aligned} \tag{4.20}$$

where $S_k = (I - \hat{\rho}_k s_k \hat{y}_k^\top)$ and H_k^0 is a well chosen positive definite matrix.

To prove that the inverse Hessian approximation is bounded, we use the following assumption:

Assumption 7. *The approximation of the gradient of f, $g(x, \xi)$, is Lipschitz continuous, i.e., there exists $L_g > 0$ such that for all x, $y \in \mathbb{R}^n$, $\|\nabla g(x, \xi) - \nabla g(y, \xi)\| \leq L_g \|x - y\|$.*

This assumption is required to prove convergence and convergence rates for most recent stochastic quasi-Newton methods. It is discussed in great detail in (Yousefian et al., 2017). Assumption 7 is less restrictive than requiring $f(x, \xi)$ to be twice differentiable with respect to x, and the norm of the Hessian $\|\nabla_{xx}^2 f(x, \xi)\|$ to be bounded for any x, ξ, as required in (Wang et al., 2017).

The following theorem shows that the eigenvalues of H_{k+1} are uniformly bounded.

Theorem 4.3.2. *Let $B_k = H_k^{-1}$ and H_{k+1}^0 be two positive definite matrices and $p > 0$. If H_{k+1} is obtained by applying p times the damped BFGS update formula with inexact gradient* (4.20) *to H_{k+1}^0, then there exists two positive constants λ and Λ such that*

$$\lambda_{\min}(H_{k+1}) \geq \lambda \qquad \text{and} \qquad \lambda_{\max}(H_{k+1}) \leq \Lambda. \tag{4.21}$$

Proof. Consider one damped BFGS update (4.12) using s and y defined in (4.18) and $\hat{y}_k$ defined in (4.13), i.e, $p = 1$,

$$H_{k+1} = S_k H_{k+1}^0 S_k^\top + \hat{\rho}_k s_k s_k^\top, \quad \text{where} \quad \hat{\rho}_k = \frac{1}{s_k^\top \hat{y}_k}, \quad S_k = (I - \hat{\rho}_k s_k \hat{y}_k^\top).$$

We have

$$\begin{aligned}
\lambda_{\min}(H_{k+1}) &\geq \lambda_{\min}(\mu_1 S_k S_k^\top + \hat{\rho}_k s_k s_k^\top), \quad \text{where } \mu_1 = \lambda_{\min}(H_{k+1}^0), \\
\lambda_{\max}(H_{k+1}) &\leq \lambda_{\max}(\mu_2 S_k S_k^\top + \hat{\rho}_k s_k s_k^\top), \quad \text{where } \mu_2 = \lambda_{\max}(H_{k+1}^0).
\end{aligned}$$

Because H_{k+1}^0 is positive definite, we have $0 < \mu_1 \leq \mu_2$. Let us show that we can apply Lemma 4.3.1 to

$$A_1 = \mu_1 S_k S_k^\top + \hat{\rho}_k s_k s_k^\top \quad \text{and} \quad A_2 = \mu_2 S_k S_k^\top + \hat{\rho}_k s_k s_k^\top.$$

From (4.19), we obtain

$$s_k^\top \hat{y}_k \geq \eta \lambda_{\min}((H_{k+1}^0)^{-1}) \|s_k\|^2 = \frac{\eta}{\lambda_{\max}(H_{k+1}^0)} \|s_k\|^2 = \frac{\eta}{\mu_1} \|s_k\|^2.$$

Moreover, using Assumption (7) we have:

$$\|\hat{y}_k = \theta_k y_k + (1-\theta_k)(H_{k+1}^0)^{-1} s_k\| \leq \|y_k\| + \|(H_{k+1}^0)^{-1} s_k\| \leq L_g \|s_k\| + \|(H_{k+1}^0)^{-1} s_k\|,$$

then,

$$\|\hat{y}_k\| \leq (L_g + \lambda_{\max}((H_{k+1}^0)^{-1}) \|s_k\| = (L_g + \frac{1}{\lambda_{\min}(H_{k+1}^0)}) \|s_k\|.$$

Therefore, we can apply Lemma (4.3.1) with s_k, $\hat{y}_k$, $\gamma = \frac{\eta}{\mu_1} > 0$, $L_y = L_g + \frac{1}{\lambda_{\min}(H_{k+1}^0)} > 0$ and $\mu = \mu_1 > 0$ for A_1, respectively $\mu = \mu_2 > 0$ for A_2.

Let us note $L_{k+1} = L_g + \frac{1}{\lambda_{\min}(H_{k+1}^0)}$. It follows that:

$$\lambda_{\min}(H_{k+1}) \geq \min\left(\frac{1}{L_{k+1}}, \frac{\lambda_{\min}(H_{k+1}^0)}{1 + \frac{\lambda_{\min}(H_{k+1}^0)\lambda_{\max}(H_{k+1}^0)}{\eta}(L_{k+1})^2} \right),$$
$$\lambda_{\max}(H_{k+1}) \leq \frac{\lambda_{\max}(H_{k+1}^0)}{\eta} + \max\left(0, \frac{\lambda_{\max}(H_{k+1}^0)^3}{\eta^2} L_{k+1}^2 - \frac{\lambda_{\max}(H_{k+1}^0)}{1 + \frac{\lambda_{\max}(H_{k+1}^0)^2}{\eta} L_{k+1}^2} \right).$$

Now, we consider the case where $p > 1$ and let us note

$$H_{k+1}^{h+1} := S_{k-h} H_{k+1}^h S_{k-h}^\top + \hat{\rho}_{k-h} s_{k-h} s_{k-h}^\top, \quad 0 \leq h \leq p-1,$$

where

$$H_{k+1}^{p} := H_{k+1}, \quad \hat{\rho}_{k-h} = \frac{1}{s_{k-h}^{\top}\hat{y}_{k-h}}, \quad S_k = (I - \hat{\rho}_{k-h}s_{k-h}\hat{y}_{k-h}^{\top}).$$

Similarly to the case $p = 1$, we have

$$\lambda_{\min}(H_{k+1}^{h+1}) \geq \lambda_{\min}(\mu_1^h S_{k-h}S_{k-h}^{\top} + \hat{\rho}_{k-h}s_{k-h}s_{k-h}^{\top}), \quad \text{where } \mu_1^h = \lambda_{\min}(H_{k+1}^{h}),$$
$$\lambda_{\max}(H_{k+1}^{h+1}) \leq \lambda_{\max}(\mu_2^h S_{k-h}S_{k-h}^{\top} + \hat{\rho}_{k-h}s_{k-h}s_{k-h}^{\top}), \quad \text{where } \mu_2^h = \lambda_{\max}(H_{k+1}^{h}).$$

Recursively, we have that H_{k+1}^{h} is positive definite, $0 < \mu_1^h \leq \mu_2^h$. We show that we can apply Lemma (4.3.1) to

$$A_1^h = \mu_1^h S_{k-h}S_{k-h}^{\top} + \hat{\rho}_{k-h}s_{k-h}s_{k-h}^{\top} \quad \text{and} \quad A_2^h = \mu_2^h S_{k-h}S_{k-h}^{\top} + \hat{\rho}_{k-h}s_{k-h}s_{k-h}^{\top}.$$

From (4.19), we have

$$s_{k-h}^{\top}\hat{y}_{k-h} \geq \eta\lambda_{\min}((H_{k-h+1}^{0})^{-1})\|s_{k-h}\|^2 = \frac{\eta}{\lambda_{\max}(H_{k-h+1}^{0})}\|s_{k-h}\|^2.$$

Using Assumption (7) we have

$$\|\hat{y}_{k-h} = \theta_{k-h}y_{k-h} + (1-\theta_{k-h})(H_{k-h+1}^{0})^{-1}s_{k-h}\| \leq L_g\|s_{k-h}\| + \|(H_{k-h+1}^{0})^{-1}s_{k-h}\|,$$

then,

$$\|\hat{y}_{k-h}\| \leq (L_g + \frac{1}{\lambda_{\min}(H_{k-h+1}^{0})})\|s_{k-h}\|.$$

We apply Lemma (4.3.1) with $s = s_{k-h}$, $y = \hat{y}_{k-h}$, $\gamma = \frac{\eta}{\lambda_{\max}(H_{k-h+1}^{0})} > 0$, $L_y = L_g + \frac{1}{\lambda_{\min}(H_{k-h+1}^{0})} > 0$ and $\mu = \mu_1^h > 0$ for A_1^h, respectively $\mu = \mu_2^h > 0$ for A_2^h.
Let us note $L_{k-h+1} = L_g + \frac{1}{\lambda_{\min}(H_{k-h+1}^{0})}$ and $\gamma_{k-h+1} = \frac{\eta}{\lambda_{\max}(H_{k-h+1}^{0})}$. Then we have

$$\lambda_{\min}(H_{k+1}^{h+1}) \geq \min\left(\frac{1}{L_{k-h+1}}, \frac{\lambda_{\min}(H_{k+1}^{h})}{1 + \frac{\lambda_{\min}(H_{k+1}^{h})}{\gamma_{k-h+1}}(L_{k-h+1})^2}\right), \tag{4.22}$$

and

$$\lambda_{\max}(H_{k+1}^{h+1}) \leq \frac{1}{\gamma_{k-h+1}} + \max\left(0, \frac{\lambda_{\max}(H_{k+1}^{h})}{\gamma_{k-h+1}^2} L_{k-h+1}^2 - \frac{\lambda_{\max}(H_{k+1}^{h})}{1 + \frac{\lambda_{\max}(H_{k+1}^{h})}{\gamma_{k-h+1}} L_{k-h+1}^2}\right). \tag{4.23}$$

It is clear that we can obtain the lower bound on $\lambda_{\min}(H_{k+1})$ recursively using (4.22). Obtaining the upper bound on $\lambda_{\max}(H_{k+1})$ using (4.23) is trickier. However, we notice that inequality (4.23) implies

$$\lambda_{\max}(H_{k+1}^{h+1}) \leq \frac{1}{\gamma_{k-h+1}} + \frac{\lambda_{\max}(H_{k+1}^{h})}{\gamma_{k-h+1}^2} L_{k-h+1}^2. \tag{4.24}$$

This upper bound is less tight but it allows us to compute $\lambda_{\max}(H_{k+1})$ recursively. □

Now that we obtained the upper (4.24) and lower (4.22) bounds on the maximum and minimum eigenvalues of H_{k+1}, respectively, we use them to define our algorithm in the next section.

4.3.2 Description of the algorithm: SdLBFGS-CHN

First, we discuss the choice of H_{k+1}^0. A common choice for H_{k+1}^0 (Nocedal and Wright, 2006) is

$$H_{k+1}^0 = \gamma_{k+1}^{-1}\mathbb{I}, \quad \text{where} \quad \gamma_{k+1} = \frac{y_k^\top y_k}{s_k^\top y_k},$$

and γ_{k+1} is called the *scaling parameter*. This choice ensures that the search direction is well scaled, enabling the use of a large step size. To prevent H_{k+1}^0 from becoming singular or non-positive definite (since there is no guarantee in the non-convex case that $s_k^\top y_k > 0$), we choose to define H_{k+1}^0 as follows:

$$H_{k+1}^0 = \left(\max(\underline{\gamma}_{k+1}, \min(\gamma_{k+1}, \bar{\gamma}_{k+1}))\right)\mathbb{I}, \quad \text{where} \quad \gamma_{k+1} = \frac{y_k^\top y_k}{s_k^\top y_k}, \tag{4.25}$$

and $0 < \underline{\gamma}_{k+1} < \bar{\gamma}_{k+1}$ can be constants or iteration-dependent.

We use the mini-batch gradient approximation

$$g(x_k, \xi_k) = \frac{1}{m_k} \sum_{i=1}^{m_k} \nabla f_{\xi_{k,i}}(x_k). \tag{4.26}$$

The Hessian-gradient product used to compute the search direction $d_k = -H_k g(x_k, \xi_k)$ can be obtained cheaply by exploiting a recursive algorithm (Nocedal and Wright, 2006), as described in Algorithm 7.

Algorithm 7 Two-loop recursion algorithm for Hessian-gradient product computation

```
Require: Current iterate $x_k$, $g(x_k, \xi_k)$, $\hat{\rho}_i = \frac{1}{s_i^\top \hat{y}_i}$, $s_i$, $\hat{y}_i$ for $i \in \{k-p, \dots, k-1\}$.
 1: Define $g = g(x_k, \xi_k)$
 2: for $i = k-1, k-2, \dots, k-p$ do
 3:     Compute $\nu_i = \hat{\rho}_i s_i^\top g$
 4:     Compute $g = g - \nu_i \hat{y}_i$
 5: end for
 6: Compute $q = H_k^0 g$ using (4.25).
 7: for $i = k-p, k-p+1, \dots, k-1$ do
 8:     Compute $\mu = \hat{\rho}_i \hat{y}_i^\top q$
 9:     Compute $q = q + (\nu_i - \mu) s_i$
10: end for
11: Return $q = H_k \, g(x_k, \xi_k)$
```

Finally, we summarize our complete algorithm, the Stochastic Damped L-BFGS with Controlled Hessian Norm (SdLBFGS-CHN), in Algorithm 8. Note that the notations λ_k and Λ_k do not refer to the k-th eigenvalue and that the index k refers to the iteration number only.

In step 5 of the algorithm, we compute an estimate of the upper and lower bounds on $\lambda_{\max}(H_{k+1})$ and $\lambda_{\min}(H_{k+1})$, respectively. The only unknown quantity in the expressions of Λ in (4.24) and λ in (4.23) is the Lipschitz constant L_g. We estimate this Lipschitz constant at each iteration k by defining $L_{g,k} \approx \|y_k\|/\|s_k\|$. There are several strategies that we can consider when the bounds are not within the limits that we set as hyperparameters. Two possible strategies are presented in Algorithms 9 and 10. Again, note that the notations λ_k and Λ_k do not refer to the k-th eigenvalue and that the index k refers to the iteration number only.

4.3.3 Variance reduced SdLBFGS-CHN

Motivated by the success of recent methods combining variance reduction with stochastic L-BFGS (Gower et al., 2016; Moritz et al., 2016; Wang et al., 2017), we propose a new variance reduced variant of our algorithm. Not only would this accelerate the convergence, since can choose a constant step size, but it also improves the quality of the curvature approximation.

Algorithm 8 Stochastic Damped L-BFGS with Controlled Hessian Norm - SdLBFGS-CHN

```
Require: $x_0 \in \mathbb{R}^n$
 1: Choose a step size sequence $\{\alpha_k > 0\}_{k\geq 0}$, batch size sequence $\{m_k > 0\}_{k\geq 0}$, upper bound
    limit $\lambda_{\max}$, lower bound limit $\lambda_{\min}$, memory parameter $p$, total number of epochs $N_{\text{epochs}}$,
    and the sequences $\underline{\gamma}_{k+1}$ and $\bar{\gamma}_{k+1}$, such that $0 < \underline{\gamma}_{k+1} < \bar{\gamma}_{k+1}$.
 2: Set $t = 0$, $k = 0$.
 3: Sample batch $\xi_k$ of size $m_k$. Compute $g(x_k, \xi_k)$.
 4: If $k = 0$, define $d_k = -g(x_k, \xi_k)$, otherwise compute $d_k = -H_k g(x_k, \xi_k)$ using Algorithm 7.
    Then define $x_{k+1} = x_k + \alpha_k d_k$.
 5: Compute $s_k$, $y_k$ as in (4.18), and compute $\hat{y}_k$ as in (4.13).
 6: Estimate $\Lambda_k$ and $\lambda_k$ in (4.21) recursively using (4.24) and (4.23). If $\Lambda_k > \lambda_{\max}$ or
    $\lambda_k < \lambda_{\min}$, use strategy 9 or 10.
 7: if the epoch is completed and $t = N_{\text{epochs}} - 1$ then
 8:     Terminate with the approximate solution $x_{k+1}$.
 9: else
10:     if the epoch is completed and $t < N_{\text{epochs}} - 1$ then
11:         Increment $t$ by one and shuffle the dataset.
12:     end if
13:     Increment $k$ by one and go to step 3.
14: end if
```

Algorithm 9 Hessian update strategy n.1 when bounds are not acceptable

```
Require: $\Lambda_k$, $\lambda_k$, $\lambda_{\max}$ and $\lambda_{\min}$
 1: if $\Lambda_k > \lambda_{\max}$ or $\lambda_k < \lambda_{\min}$ then
 2:     delete stored values of $s_i$, $y_i$ and $\hat{y}_i$, $i \in \{k - p + 1, \dots, k\}$.
 3:     in the next iteration, use search direction $d_{k+1} = -g(x_{k+1}, \xi_{k+1})$.
 4: end if
```

Algorithm 10 Hessian update strategy n.2 when bounds are not acceptable

```
Require: $\Lambda_k$, $\lambda_k$, $\lambda_{\max}$ and $\lambda_{\min}$
 1: if $\Lambda_k > \lambda_{\max}$ or $\lambda_k < \lambda_{\min}$ then
 2:     delete stored values of $s_i$, $y_i$ and $\hat{y}_i$, $i \in \{k - p + 1, \dots, k - 1\}$.
 3:     in the next iteration, use search direction $d_{k+1} = -H_{k+1} g(x_{k+1}, \xi_{k+1})$, where
        $H_{k+1} g(x_{k+1}, \xi_{k+1})$ is computed using the last pair $(s_k, \hat{y}_k)$.
 4: end if
```

We apply an SVRG-like type of variance reduction. The new version of SdLBFGS-CHN is described in Algorithm 11.

Algorithm 11 Variance reduced SdLBFGS-CHN - VR-SdLBFGS-CHN

```
Require: $x_0 \in \mathbb{R}^n$
 1: Choose a step size sequence $\{\alpha_k > 0\}_{k\geq 0}$, batch size sequence $\{m_k > 0\}_{k\geq 0}$, upper bound
    limit $\lambda_{\max}$, lower bound limit $\lambda_{\min}$, memory parameter $p$, total number of epochs $N_{\text{epochs}}$,
    and the sequences $\underline{\gamma}_{k+1}$ and $\bar{\gamma}_{k+1}$, such that $0 < \underline{\gamma}_{k+1} < \bar{\gamma}_{k+1}$.
 2: Set $t = 0$, $k = 0$.
 3: Define $x_k^t = x_k$ and compute the full gradient $\nabla f(x_k^t)$.
 4: Sample batch $\xi_k$ of size $m_k$ and compute $g(x_k, \xi_k)$ and $g(x_k^t, \xi_k)$.
 5: Define $\tilde{g}(x_k, \xi_k) = g(x_k, \xi_k) - g(x_k^t, \xi_k) + \nabla f(x_k^t)$.
 6: If $k = 0$, define $d_k = -\tilde{g}(x_k, \xi_k)$, otherwise compute $d_k = -H_k\tilde{g}(x_k, \xi_k)$ using Algorithm 7.
    Then define $x_{k+1} = x_k + \alpha_k d_k$.
 7: Compute $s_k$, $y_k$ as in (4.18), and $\hat{y}_k$ as in (4.13) using $\tilde{g}(x_k, \xi_k)$.
 8: Estimate $\Lambda_k$ and $\lambda_k$ in (4.21) recursively using (4.24) and (4.23). If $\Lambda_k > \lambda_{\max}$ or
    $\lambda_k < \lambda_{\min}$, use strategy 9 or 10.
 9: if the epoch is completed and $t = N_{\text{epochs}} - 1$ then
10:     Terminate with the approximate solution $x_{k+1}$.
11: else if the epoch is completed and $t < N_{\text{epochs}} - 1$ then
12:     Increment $t$ by one, shuffle the dataset, increment $k$ by one and go to step 3.
13: else
14:     Increment $k$ by one and go to step 4.
15: end if
```

Notice that the full gradient is computed once in every epoch in step 3. As mentioned in the literature review, Chapter 2, $(g(x_k^t, \xi_k) - \nabla f(x_k^t))$ can be seen as the bias in the gradient estimation $g(x_k, \xi_k)$, and it is used here to correct the gradient approximation in step 5.

4.4 Convergence and Complexity Analysis

Since (Wang et al., 2017) presented a general framework for stochastic quasi-Newton methods, we show that the results of their convergence analysis and iteration complexity apply to SdLBFGS-CHN.

We consider the assumptions 4, 5 and 6. Here are the additional assumptions used by (Wang et al., 2017) to prove the global convergence:

Assumption 8. *The full-gradient $\nabla f(x_k)$ and the gradient approximation $g(x_k, \xi_k)$ verify, for every iteration k,*

$$\mathbb{E}_{\xi_k}\left[g(x_k, \xi_k)\right] = \nabla f(x_k), \tag{4.27}$$

and

$$\mathbb{E}_{\xi_k}\left[\|g(x_k,\xi_k)-\nabla f(x_k)\|^2\right] \leq \sigma^2, \tag{4.28}$$

where $\sigma > 0$, and for every k, the random variable ξ_k is independent of $\{x_1, \dots, x_k\}$.

Assumption 9. *There exists two positive constants, λ and Λ, such that, for every $k \geq 0$*

$$\lambda \mathbb{I} \preceq H_k \preceq \Lambda \mathbb{I}. \tag{4.29}$$

Assumption 10. *H_k depends only on $\{\xi_i\}_{i=1}^{k-1}$.*

Assumption 11. *The step size sequence $\{\alpha_k > 0\}_{k\geq 0}$ satisfies*

$$\sum_{k=0}^{\infty} \alpha_k = +\infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 < +\infty. \tag{4.30}$$

First, notice that equation (4.27) of Assumption 8 is satisfied, since we use the mini-batch gradient (4.26). Theorem 4.3.2 proves that Assumption 9 is satisfied for our algorithm. Assumption 10 is satisfied by definition of H_k (4.8). Finally, we choose $\alpha_k = c/(k+1)$, where $c > 0$ is a constant, to satisfy Assumption 11.

Therefore, we can announce the following convergence theorems of SdLBFGS-CHN:

Theorem 4.4.1 (Wang et al., 2017, Theorem 2.1)**.** *Assume that we use a fixed batch size $m_k = m$ and that Assumptions 4-11 hold for $\{x_k\}$ generated by SdLBFGS-CHN. If the step size $\alpha_k \leq \frac{\lambda}{L\Lambda}$, for all $k \geq 0$, then*

$$\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0, \quad \text{with probability} \quad 1. \tag{4.31}$$

Moreover, there exists a positive constant M_f, such that

$$\mathbb{E}[f(x_k)] \leq M_f, \quad \forall k \geq 0. \tag{4.32}$$

Theorem 4.4.2 (Wang et al., 2017, Theorem 2.3)**.** *Assume that we use a fixed batch size $m_k = m$ and that Assumptions 4-11 hold for $\{x_k\}$ generated by SdLBFGS-CHN.*

If the step size is chosen as

$$\alpha_k = \frac{\lambda}{L\Lambda^2} k^{-\beta}, \quad \forall k \geq 0, \quad \text{with,} \quad \beta \in (0.5, 1), \tag{4.33}$$

then,

$$\frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\left[\|\nabla f(x_k)\|^2\right] \le \frac{2L(M_f-\kappa_{low})\Lambda^2}{\lambda^2}T^{\beta-1}+\frac{\sigma^2}{(1-\beta)m}(T^{-\beta}-T^{-1}), \tag{4.34}$$

where T is the number of iterations.

Moreover, for a given $\epsilon \in (0,1)$, the number of iterations T needed to guarantee that

$$\frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\left[\|\nabla f(x_k)\|^2\right] \le \epsilon, \tag{4.35}$$

is at most $\mathcal{O}(\epsilon^{-\frac{1}{1-\beta}})$.

4.5 Experimental results

We compare the performance of our algorithm to that of SdLBFGS-VR by (Wang et al., 2017) and to SGD for solving a multi-class classification problem. For this purpose, we train a modified version[1] of the deep neural network model DavidNet[2] that was proposed by David C. Page, on CIFAR-10 (Krizhevsky, 2009). The latter is a dataset that contains 60,000 colour images with labels in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck), each containing 6,000 images. We use 50,000 images for the training task and 10,000 images for the validation task.

Note that we also compared the performance of SdLBFGS-VR and VR-SdLBFGS-CHN in solving other problems, such as a logistic regression problems using the MNIST dataset (LeCun et al., 2010) and a non-convex support vector machine problem with a sigmoid loss function using the RCV1 dataset (Lewis et al., 2004). The performance of the two algorithms is **on par** because these problems **are not highly non-convex or ill-conditioned**. By contrast, training DavidNet on CIFAR-10 **is a highly non-convex problem**, for which the Hessian approximation becomes **ill-conditioned** during the training. Therefore, we only present the results for the latter in this section.

Details of the experiments:

- since the bound on the maximum eigenvalue of H_{k+1} is loose, we apply our Hessian norm control using the bound on the minimum eigenvalue of H_{k+1} only. We have $\lambda_{\min}(H_{k+1}) = \frac{1}{\lambda_{\max}(B_{k+1})} = \frac{1}{\|B_{k+1}\|}$. Hence, controlling the bound on $\lambda_{\min}(H_{k+1})$ enables

[1]FastResNet Hyperparameters tuning with Ax on CIFAR10

[2]https://myrtle.ai/learn/how-to-train-your-resnet-4-architecture/

us to control the norm of the matrix B_{k+1} from becoming very large and its inverse from becoming singular.

- for the definition of H^0_{k+1} in (4.25), we choose $\underline{\gamma}_{k+1}$ and $\bar{\gamma}_{k+1}$ to be constant, independent of the iteration.

Numerical values:

- For all algorithms: we train the network for 20 epochs and use a batch size of 256 samples.
- For SGD, we choose a step size equal to 0.001.
- For both SdLBFGS-VR and VR-SdLBFGS-CHN, the memory parameter p is equal to 10, the minimal scaling parameter $\underline{\gamma}$ is equal to 0.1, the constant step size α is equal to 0.1 and η is equal to 0.25.
- For VR-SdLBFGS-CHN, we use a maximal scaling parameter $\bar{\gamma}$ that is equal to 10^5 and a lower bound limit $\lambda_{\min}$ that is equal to 10^{-5}.

Figure 4.1 shows that VR-SdLBFGS-CHN outperforms SdLBFGS-VR for the training of the neural network, both outperforming SGD. We see that for the validation accuracy, VR-SdLBFGS-CHN has an edge over SdLBFGS-VR, both outperforming SGD again.

More importantly, we see that the performance of VR-SdLBFGS-CHN is more stable than that of SdLBFGS-VR. To further investigate this observation, we plot the evolution of the estimation of lower bound on the smallest eigenvalue for the first 5 epochs (an epoch is equivalent to 195 iterations). Figure 4.2 shows that during the first epochs, the lower bound for SdLBFGS-VR takes very small values. This might imply an ill-conditioning issue and explain the instability in the behaviour of SdLBFGS-VR. Therefore, we demonstrate that our technique, which aims to control the quality of the Hessian approximation, provides a more stable behaviour for highly non-convex and ill-conditioned problems.

4.6 Conclusion

In this chapter, we tackled the problem of using the stochastic damped L-BFGS algorithm to solve optimization problems in the non-convex setting, where there are no guarantees that the Hessian matrix approximation remains well-conditioned and non-singular throughout the optimization procedure. For this purpose, we introduced a new stochastic damped L-BFGS algorithm that monitors the quality of the inverse Hessian approximation during the

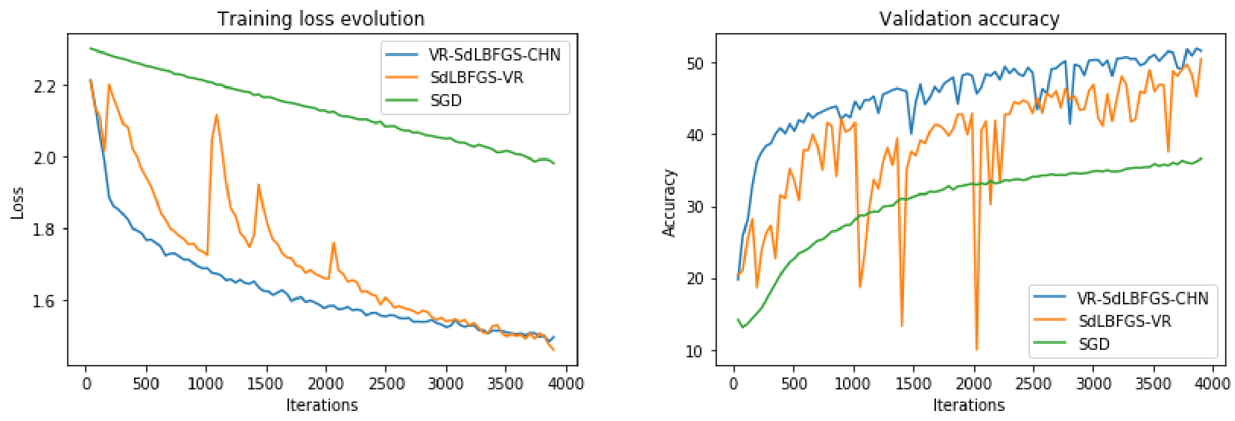

Figure 4.1 Evolution of the training loss (left) and accuracy (right) for training a modified DavidNet on CIFAR-10.

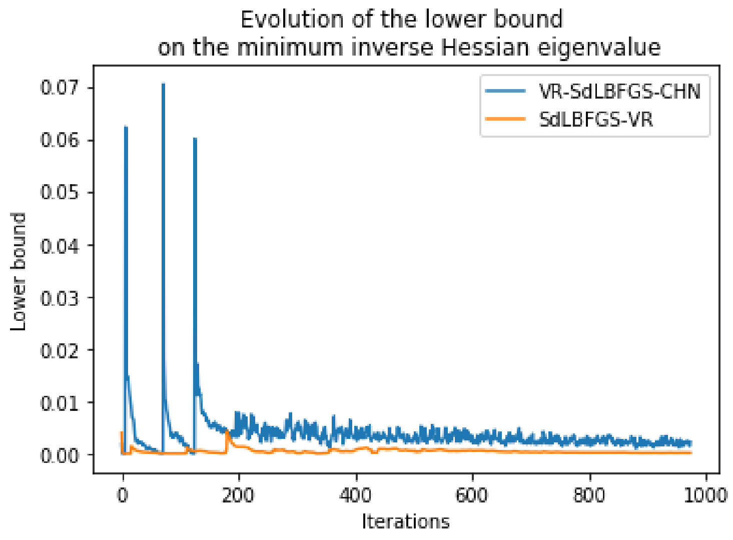

Figure 4.2 Evolution of the lower bound on the smallest eigenvalues of the inverse Hessian approximation for training a modified DavidNet on CIFAR-10.

optimization by estimating the bounds on its largest and smallest eigenvalues. Our work is the first to address the Hessian singularity problem by approximating and leveraging these bounds. Moreover, we defined a new formula of the initial inverse Hessian approximation that makes the algorithm more stable. Additionally, we introduced a variance-reduced version of our algorithm in order to improve the quality of the curvature approximation and accelerate the convergence. Our algorithm converges almost-surely to a stationary point, and numerical experiments show that it is more robust to ill-conditioned problems and more suitable to the highly non-convex context of deep learning.

CHAPTER 5 CONCLUSION AND RECOMMENDATIONS

In this thesis, we presented novel first- and second-order algorithms in order to solve convex and non-convex large-scale optimization problems, especially arising in the context of machine learning. In this chapter, we discuss our contributions in Section 5.1, the limitations of our work in Section 5.2, and finally, we offer recommendations for interesting research venues in Section 5.3.

5.1 Advancement of knowledge

In Chapter 3, we proposed 3 novel first-order optimization algorithms, namely

- ARIGAF: this algorithm is a new version of ARIG (Bonniot, 2018) that is adapted to the stochastic setting. We show that ARIGAF enjoys the same theoretical guarantees as ARIG. This new algorithm defines a general transition framework from deterministic and semi-deterministic trust-region methods to fully-stochastic trust-region methods.
- ARIG+: it is the combination of the stochastic version of ARIG with an adaptive sampling technique. The latter is used to ensure that the accuracy condition on the gradient is satisfied. ARIG+ outperforms SGD for solving a logistic regression problem using the MNIST dataset.
- ARAS: this algorithm leverages the two-phase nature of stochastic optimization procedures in order to incorporate both adaptive regularization and adaptive sampling. In contrast with ARIG+, the step size in ARAS converges to zero in the second phase of the optimization, which guarantees the overall convergence of the algorithm. To the best of our knowledge, no previous work attempted to use Pflug diagnostic (Chee and Toulis, 2018) in the way we propose.

Recall Table 5.1 that summarizes the properties of the 3 novel algorithms, as well as ARIG.

In Chapter 4, we used a less restrictive assumption, in comparison to related works, to prove that the L-BFGS inverse Hessian approximation is bounded. We then introduced a new stochastic damped L-BFGS algorithm, SdLBFGS-CHN, and its variance-reduced version VR-SdLBFGS-CHN. The algorithm incorporates an estimation of the bounds on the smallest and largest eigenvalues of the Hessian approximation to control its quality. We also presented a new formula to define the initial inverse Hessian approximation, which makes

Table 5.1 Comparison of the characteristics of ARIG, ARIGAF, ARIG+ and ARAS.

	ARIG	ARIGAF	ARIG+	ARAS
Stochasticity	✗	✓	✓	✓
Theoretical guarantees	✓	✓	✗	?
Adaptive step size	✓	✓	✓	✓
Adaptive batch size	✗	✗	✓	✓
Practical for machine learning	✗	✗	✓	✓
Conceivable overall convergence	✓	✓	✗	✓

the algorithm more stable. Numerical experiments demonstrate that our algorithm is more robust to ill-conditioning. To the best of our knowledge, no other work proposed to solve the ill-conditioning problem via monitoring the quality of the Hessian approximation using the bounds on its eigenvalues. The proposed algorithm converges almost surely.

5.2 Limits and constraints

Our algorithms in Chapters 3 and 4 are adapted to both convex and non-convex settings. However, there are some limits and constraints related to our methods, namely

- For our novel first-order methods, we propose to use adaptive sampling to satisfy the accuracy condition on the gradient estimate. However, we do not propose a practical method to verify the accuracy of the function approximation. Moreover, our adaptive sampling technique relies on the assumption that the batch size increase is gradual, which might not be true for all types of problems.
- For our novel second-order algorithm, we estimate the bounds on the eigenvalues recursively. The recursive approach loosens the bounds and makes the approximation less reliable. Moreover, we estimate the stochastic gradient's Lipschitz constant using an empirical heuristic without guarantees about its quality.

5.3 Recommendations

It is conceivable that very large batches would be needed to maintain the accuracy condition on the function estimates in stochastic trust-region methods (Paquette, 2020). There are two possible solutions to this problem. The first solution consists of not taking the approximation's accuracy into account, the same way we did for ARIG+ and ARAS, inspired by (Curtis et al., 2019) who also dropped the accuracy condition. In this case, a new convergence analysis would be needed for ARIG+ and ARAS. The second option is to incorporate an aggregation-like variance reduction technique that would reduce the variance in the function approximations without requiring the use of enormous batches.

For our second-order method, one can try to replace the SVRG-like variance reduction technique with adaptive sampling. Several works proposed to combine adaptive sampling with stochastic quasi-Newton methods, such as (Jalilzadeh et al., 2018), (Bollapragada et al., 2018b), and (Bollapragada and Wild, 2019). To weaken the Lipschitz condition on the stochastic gradients, one can use the smoothing technique proposed by (Yousefian et al., 2017). The technique only applies to the strongly convex case, whereas our purpose is to solve highly non-convex problems. Therefore, it needs to be adapted to our case. Finally, one can incorporate Nesterov's momentum term in the update in order to accelerate our algorithm (Ninomiya, 2017), as well as its variance-reduced version (Yasuda et al., 2019).

REFERENCES

Z. Allen-Zhu and E. Hazan, “Variance reduction for faster non-convex optimization,” in *International conference on machine learning*, 2016, pp. 699–707.

H. Asi and J. C. Duchi, “The importance of better models in stochastic optimization,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 46, pp. 22 924–22 930, 2019.

A. Bahamou and D. Goldfarb, “A dynamic sampling adaptive-sgd method for machine learning,” *arXiv preprint arXiv:1912.13357*, 2019.

A. S. Bandeira, K. Scheinberg, and L. N. Vicente, “Convergence of trust-region methods based on probabilistic models,” *SIAM Journal on Optimization*, vol. 24, no. 3, pp. 1238–1264, 2014.

A. S. Berahas and M. Takáč, “A robust multi-batch l-bfgs method for machine learning,” *Optimization Methods and Software*, vol. 35, no. 1, pp. 191–219, 2020.

E. G. Birgin, J. Gardenghi, J. M. Martínez, S. A. Santos, and P. L. Toint, “Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models,” *Mathematical Programming*, vol. 163, no. 1-2, pp. 359–368, 2017.

J. Blanchet, C. Cartis, M. Menickelly, and K. Scheinberg, “Convergence rate analysis of a stochastic trust-region method via supermartingales,” *INFORMS journal on optimization*, vol. 1, no. 2, pp. 92–119, 2019.

R. Bollapragada and S. M. Wild, “Adaptive sampling quasi-newton methods for derivative-free stochastic optimization,” *arXiv preprint arXiv:1910.13516*, 2019.

R. Bollapragada, R. Byrd, and J. Nocedal, “Adaptive sampling strategies for stochastic optimization,” *SIAM Journal on Optimization*, vol. 28, no. 4, pp. 3312–3343, 2018.

R. Bollapragada, D. Mudigere, J. Nocedal, H.-J. M. Shi, and P. T. P. Tang, “A progressive batching l-bfgs method for machine learning,” *arXiv preprint arXiv:1802.05374*, 2018.

T. Bonniot, “Convergence and complexity of unconstrained optimization methods with inexact gradients,” Master’s thesis, ENSEEIHT, Toulouse, France, September 2018, intership report, supervised by S. Gratton, D. Orban and Ph. Toint).

A. Bordes, L. Bottou, and P. Gallinari, “Sgd-qn: Careful quasi-newton stochastic gradient descent,” *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1737–1754, 2009.

L. Bottou, “Online learning and stochastic approximations,” *On-line learning in neural networks*, vol. 17, no. 9, p. 142, 1998.

——, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.

L. Bottou and O. Bousquet, “The tradeoffs of large scale learning,” in *Advances in neural information processing systems*, 2008, pp. 161–168.

L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

C. G. Broyden, “The convergence of a class of double-rank minimization algorithms 1. general considerations,” *IMA Journal of Applied Mathematics*, vol. 6, no. 1, pp. 76–90, 1970.

R. H. Byrd, G. M. Chin, J. Nocedal, and Y. Wu, “Sample size selection in optimization methods for machine learning,” *Mathematical programming*, vol. 134, no. 1, pp. 127–155, 2012.

R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, “A stochastic quasi-newton method for large-scale optimization,” *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1008–1031, 2016.

C. Cartis, N. I. Gould, and P. L. Toint, “Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results,” *Mathematical Programming*, vol. 127, no. 2, pp. 245–295, 2011.

——, “Adaptive cubic regularisation methods for unconstrained optimization. part ii: worst-case function-and derivative-evaluation complexity,” *Mathematical programming*, vol. 130, no. 2, pp. 295–319, 2011.

A. Cauchy, “Méthode générale pour la résolution des systemes d’équations simultanées,” *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

K.-H. Chang, L. J. Hong, and H. Wan, “Stochastic trust-region response-surface method (strong)—a new response-surface framework for simulation optimization,” *INFORMS Journal on Computing*, vol. 25, no. 2, pp. 230–243, 2013.

T. Chavdarova, S. Stich, M. Jaggi, and F. Fleuret, “Reducing noise in gan training with variance reduced extragradient,” 2019, preprint, to appear in arXiv.

J. Chee and P. Toulis, "Convergence diagnostics for stochastic gradient descent with constant learning rate," in *International Conference on Artificial Intelligence and Statistics*, 2018, pp. 1476–1485.

H. Chen, H.-C. Wu, S.-C. Chan, and W.-H. Lam, "A stochastic quasi-newton method for large-scale nonconvex optimization with applications," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

R. Chen, M. Menickelly, and K. Scheinberg, "Stochastic optimization using a trust-region method and random models," *Mathematical Programming*, vol. 169, no. 2, pp. 447–487, 2018.

J. Collins, J. Sohl-Dickstein, and D. Sussillo, "Capacity and trainability in recurrent neural networks," *arXiv preprint arXiv:1611.09913*, 2016.

A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods.* SIAM, 2000.

F. E. Curtis and K. Scheinberg, "Adaptive stochastic optimization," *arXiv preprint arXiv:2001.06699*, 2020.

F. E. Curtis, K. Scheinberg, and R. Shi, "A stochastic trust region algorithm based on careful step normalization," *Informs Journal on Optimization*, vol. 1, no. 3, pp. 200–220, 2019.

A. Defazio and L. Bottou, "On the ineffectiveness of variance reduced optimization for deep learning," *arXiv preprint arXiv:1812.04529*, 2018.

A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in Neural Information Processing Systems*, 2014, pp. 1646–1654.

J. E. Dennis and J. J. Moré, "A characterization of superlinear convergence and its application to quasi-newton methods," *Mathematics of computation*, vol. 28, no. 126, pp. 549–560, 1974.

J. E. Dennis Jr and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations.* Siam, 1996, vol. 16.

J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

C. Fang, C. J. Li, Z. Lin, and T. Zhang, “Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator,” in *Advances in Neural Information Processing Systems*, 2018, pp. 689–699.

R. Fletcher, “A new approach to variable metric algorithms,” *The computer journal*, vol. 13, no. 3, pp. 317–322, 1970.

J. Freund and R. Walpole, “Mathematical statistics. prentice-hall,” *Inc., New Jersey, USA*, 1971.

M. P. Friedlander and M. Schmidt, “Hybrid deterministic-stochastic methods for data fitting,” *SIAM Journal on Scientific Computing*, vol. 34, no. 3, pp. A1380–A1405, 2012.

M. C. Fu, “Optimization for simulation: Theory vs. practice,” *INFORMS Journal on Computing*, vol. 14, no. 3, pp. 192–215, 2002.

D. Goldfarb, “A family of variable-metric methods derived by variational means,” *Mathematics of computation*, vol. 24, no. 109, pp. 23–26, 1970.

N. Gould, D. Orban, and P. Toint, “Numerical methods for large-scale nonlinear optimization,” *Acta Numerica*, vol. 14, p. 299, 2005.

R. Gower, D. Goldfarb, and P. Richtárik, “Stochastic block bfgs: Squeezing more curvature out of data,” in *International Conference on Machine Learning*, 2016, pp. 1869–1878.

F. S. Hashemi, S. Ghosh, and R. Pasupathy, “On adaptive sampling rules for stochastic recursions,” in *Proceedings of the Winter Simulation Conference 2014*. IEEE, 2014, pp. 3959–3970.

A. Jalilzadeh, A. Nedić, U. V. Shanbhag, and F. Yousefian, “A variable sample-size stochastic quasi-newton method for smooth and nonsmooth stochastic convex optimization,” in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4097–4102.

R. Johnson and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” in *Advances in neural information processing systems*, 2013, pp. 315–323.

N. S. Keskar and R. Socher, “Improving generalization performance by switching from adam to sgd,” *arXiv preprint arXiv:1712.07628*, 2017.

D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015.

K. C. Kiwiel, “Convergence and efficiency of subgradient methods for quasiconvex minimization,” *Mathematical programming*, vol. 90, no. 1, pp. 1–25, 2001.

A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.

J. Larson and S. C. Billups, “Stochastic derivative-free optimization using a trust region framework,” *Computational Optimization and applications*, vol. 64, no. 3, pp. 619–645, 2016.

Y. LeCun, L. Bottou, G. B. Orr, K.-R. Müller *et al.*, “Neural networks: Tricks of the trade,” *Springer Lecture Notes in Computer Sciences*, vol. 1524, no. 5-50, p. 6, 1998.

Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, “Rcv1: A new benchmark collection for text categorization research,” *Journal of machine learning research*, vol. 5, no. Apr, pp. 361–397, 2004.

D. C. Liu and J. Nocedal, “On the limited memory bfgs method for large scale optimization,” *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

A. Mokhtari and A. Ribeiro, “Res: Regularized stochastic bfgs algorithm,” *IEEE Transactions on Signal Processing*, vol. 62, no. 23, pp. 6089–6104, 2014.

P. Moritz, R. Nishihara, and M. Jordan, “A linearly-convergent stochastic l-bfgs algorithm,” in *Artificial Intelligence and Statistics*, 2016, pp. 249–258.

N. Murata, “A statistical study of on-line learning,” *Online Learning and Neural Networks. Cambridge University Press, Cambridge, UK*, pp. 63–92, 1998.

Y. Nesterov, *Introductory lectures on convex optimization: A basic course.* Springer Science & Business Media, 2013, vol. 87.

Y. E. Nesterov, “A method for solving the convex programming problem with convergence rate o (1/kˆ 2),” in *Dokl. akad. nauk Sssr*, vol. 269, 1983, pp. 543–547.

L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, “Sarah: A novel method for machine learning problems using stochastic recursive gradient,” *arXiv preprint arXiv:1703.00102*, 2017.

——, “Stochastic recursive gradient algorithm for nonconvex optimization,” *arXiv preprint arXiv:1705.07261*, 2017.

H. Ninomiya, “A novel quasi-newton-based optimization for neural network training incorporating nesterov’s accelerated gradient,” *Nonlinear Theory and Its Applications, IEICE*, vol. 8, no. 4, pp. 289–301, 2017.

J. Nocedal, “Updating quasi-newton matrices with limited storage,” *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

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

C. Paquette, Personal Communication, 2020.

R. Pasupathy, P. Glynn, S. Ghosh, and F. Hashemi, “On sampling rates in stochastic recursions,” *Under Review*, 2015.

G. C. Pflug, “Gradient estimates for the performance of markov chains and discrete event processes,” *Annals of Operations Research*, vol. 39, no. 1, pp. 173–194, 1992.

——, “On the determination of the step size in stochastic quasigradient methods,” 1983.

——, “Non-asymptotic confidence bounds for stochastic approximation algorithms with constant step size,” *Monatshefte für Mathematik*, vol. 110, no. 3-4, pp. 297–314, 1990.

B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

M. J. Powell, “Algorithms for nonlinear constraints that use lagrangian functions,” *Mathematical programming*, vol. 14, no. 1, pp. 224–248, 1978.

E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.

S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, “Stochastic variance reduction for nonconvex optimization,” in *International conference on machine learning*, 2016, pp. 314–323.

S. J. Reddi, S. Kale, and S. Kumar, “On the Convergence of Adam and Beyond,” in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/pdf?id=ryQu7f-RZ

H. Robbins and S. Monro, “A stochastic approximation method,” *The annals of mathematical statistics*, pp. 400–407, 1951.

N. L. Roux, M. Schmidt, and F. R. Bach, “A stochastic gradient method with an exponential convergence rate for finite training sets,” in *Advances in Neural Information Processing Systems*, 2012, pp. 2663–2671.

N. N. Schraudolph, J. Yu, and S. Günter, “A stochastic quasi-newton method for online convex optimization,” in *Artificial intelligence and statistics*, 2007, pp. 436–443.

S. Shalev-Shwartz and T. Zhang, “Stochastic Dual Coordinate Ascent methods for regularized loss minimization,” *Journal of Machine Learning Research*, vol. 14, no. Feb, pp. 567–599, 2013.

D. F. Shanno, “Conditioning of quasi-newton methods for function minimization,” *Mathematics of computation*, vol. 24, no. 111, pp. 647–656, 1970.

T. Tieleman and G. Hinton, “Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning,” *COURSERA Neural Networks Mach. Learn*, 2012.

B. Verweij, S. Ahmed, A. J. Kleywegt, G. Nemhauser, and A. Shapiro, “The sample average approximation method applied to stochastic routing problems: a computational study,” *Computational Optimization and Applications*, vol. 24, no. 2-3, pp. 289–333, 2003.

X. Wang, S. Ma, D. Goldfarb, and W. Liu, “Stochastic quasi-newton methods for nonconvex stochastic optimization,” *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 927–956, 2017.

Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh, “Spiderboost and momentum: Faster variance reduction algorithms,” in *Advances in Neural Information Processing Systems*, 2019, pp. 2406–2416.

P. Wolfe, “Convergence conditions for ascent methods,” *SIAM review*, vol. 11, no. 2, pp. 226–235, 1969.

——, “Convergence conditions for ascent methods. ii: Some corrections,” *SIAM review*, vol. 13, no. 2, pp. 185–188, 1971.

S. Yasuda, S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya, and H. Asai, “A stochastic variance reduced nesterov’s accelerated quasi-newton method,” in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 1874–1879.

F. Yousefian, A. Nedić, and U. V. Shanbhag, “A smoothing stochastic quasi-newton method for non-lipschitzian stochastic optimization problems,” in *2017 Winter Simulation Conference (WSC)*. IEEE, 2017, pp. 2291–2302.

B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *arXiv preprint arXiv:1611.01578*, 2016.