2017

Efficient Methods For Large-Scale Empirical Risk Minimization

Aryan Mokhtari
University of Pennsylvania, aryanm@seas.upenn.edu

Efficient Methods For Large-Scale Empirical Risk Minimization

Abstract
Empirical risk minimization (ERM) problems express optimal classifiers as solutions of optimization problems in which the objective is the sum of a very large number of sample costs. An evident obstacle in using traditional descent algorithms for solving this class of problems is their prohibitive computational complexity when the number of component functions in the ERM problem is large. The main goal of this thesis is to study different approaches to solve these large-scale ERM problems.

We begin by focusing on incremental and stochastic methods which split the training samples into smaller sets across time to lower the computation burden of traditional descent algorithms. We develop and analyze convergent stochastic variants of quasi-Newton methods which do not require computation of the objective Hessian and approximate the curvature using only gradient information. We show that the curvature approximation in stochastic quasi-Newton methods leads to faster convergence relative to first-order stochastic methods when the problem is ill-conditioned. We culminate with the introduction of an incremental method that exploits memory to achieve a superlinear convergence rate. This is the best known convergence rate for an incremental method.

An alternative strategy for lowering the prohibitive cost of solving large-scale ERM problems is decentralized optimization whereby samples are separated not across time but across multiple nodes of a network. In this regime, the main contribution of this thesis is in incorporating second-order information of the aggregate risk corresponding to samples of all nodes in the network in a way that can be implemented in a distributed fashion. We also explore the separation of samples across both, time and space, to reduce the computational and communication cost for solving large-scale ERM problems. We study this path by introducing a decentralized stochastic method which incorporates the idea of stochastic averaging gradient leading to a low computational complexity method with a fast linear convergence rate.

We then introduce a rethinking of ERM in which we consider not a partition of the training set as in the case of stochastic and distributed optimization, but a nested collection of subsets that we grow geometrically. The key insight is that the optimal argument associated with a training subset of a certain size is not that far from the optimal argument associated with a larger training subset. Based on this insight, we present adaptive sample size schemes which start with a small number of samples and solve the corresponding ERM problem to its statistical accuracy. The sample size is then grown geometrically and use the solution of the previous ERM as a warm start for the new ERM. Theoretical analyses show that the use of adaptive sample size methods reduces the overall computational cost of achieving the statistical accuracy of the whole dataset for a broad range of deterministic and stochastic first-order methods. We further show that if we couple the adaptive sample size scheme with Newton's method, it is possible to consider subsequent doubling of the training set and perform a single Newton iteration in between. This is possible because of the interplay between the statistical accuracy and the quadratic convergence region of these problems and yields a method that is guaranteed to solve an ERM problem by performing just two passes over the dataset.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Electrical & Systems Engineering

First Advisor
Alejandro Ribeiro

Keywords
Adaptive sample size algorithms, Decentralized methods, Empirical risk minimization, Optimization, Stochastic methods

Subject Categories
Electrical and Electronics

EFFICIENT METHODS FOR LARGE-SCALE EMPIRICAL RISK MINIMIZATION

Aryan Mokhtari

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
2017

Supervisor of Dissertation

Alejandro Ribeiro, Rosenbluth Associate Professor of Electrical and Systems Engineering

Graduate Group Chairperson

Alejandro Ribeiro, Rosenbluth Associate Professor of Electrical and Systems Engineering

Dissertation Committee

Ali Jadbabaie, JR East Professor of Engineering, Massachusetts Institute of Technology
Asu Ozdaglar, Professor of Electrical Engineering and Computer Science, Massachusetts Institute of Technology
Gesualdo Scutari, Associate Professor of School of Industrial Engineering, Purdue University

EFFICIENT METHODS FOR LARGE-SCALE EMPIRICAL RISK MINIMIZATION

To my wife: Solmaz.

Acknowledgments

The years that I spent at UPenn as a Ph.D. student were an enjoyable experience, mainly because of my advisor, collaborators, and friends. I am extremely grateful to be able to appreciate and acknowledge their crucial influence on writing this thesis.

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Alejandro Ribeiro. This thesis without Alejandro's comments, suggestions, and support would not have been possible. I want to thank him for his significant impact in shaping my career and more importantly my personality. It has been my great pleasure to have the privilege of being his student. I believe that the ultimate goal of a teacher or advisor is to touch the lives of her/his students, and Alejandro was successful in positively influencing my life at a stage that I will always regard fondly.

I would like to thank my committee members Prof. Ali Jadbabaie, Prof. Asu Ozdaglar, and Prof. Gesualdo Scutari for their insightful comments, valuable criticism, and constant encouragement.

Indeed, writing this thesis would not have been possible without the joint effort of my collaborators. I would like to thank Prof. Qing Ling, Prof. Thomas Hofmann, Dr. Wei Shi, Dr. Aurelien Lucchi, Hadi Daneshmand, and Mark Eisen for their contributions to various elements of this thesis. Also, I would like to thank Prof. Geert Leus, Prof. Ali Jadbabaie, Prof. Gesualdo Scutari, Prof. Mert Gurbuzbalaban, Dr. Alec Koppel, Dr. Shahin Shahrampour, Dr. Andrea Simonetto, Santiago Paternain, and Tianyi Chen for giving me the opportunity to collaborate with them during my graduate study. Although none of the technical content in this thesis emerged from these collaborations, their positive impact on my research experience which led to writing this thesis is invaluable.

I would like to thank my friends both here and back home in Iran. I am especially grateful to my lab-mates who made this arduous journey an enjoyable experience. Special thanks to my parents, Shahnaz and Reza. Thank you is an insufficient phrase for your consistent support, limitless sacrifices, and indescribable love. To my brother, Ali, who is an extraordinary friend and role model for me. To my aunt, Fatemeh (Nasrin) Mohseni-Mofidi, thanks for your constant encouragement and support. Last but not least, to the love of my life Solmaz. Thank you, for always being my biggest cheerleader and more

importantly for believing in me more than myself. This thesis would not have been written without your unconditional love and support. Dedicating this thesis to you is the least I could do to acknowledge your role in writing this thesis and appreciate your kindness and love.

Aryan Mokhtari, Philadelphia, July 2017

ABSTRACT

EFFICIENT METHODS FOR LARGE-SCALE EMPIRICAL RISK MINIMIZATION

Aryan Mokhtari
Alejandro Ribeiro

Empirical risk minimization (ERM) problems express optimal classifiers as solutions of optimization problems in which the objective is the sum of a very large number of sample costs. An evident obstacle in using traditional descent algorithms for solving this class of problems is their prohibitive computational complexity when the number of component functions in the ERM problem is large. The main goal of this thesis is to study different approaches to solve these large-scale ERM problems.

We begin by focusing on incremental and stochastic methods which split the training samples into smaller sets across time to lower the computation burden of traditional descent algorithms. We develop and analyze convergent stochastic variants of quasi-Newton methods which do not require computation of the objective Hessian and approximate the curvature using only gradient information. We show that the curvature approximation in stochastic quasi-Newton methods leads to faster convergence relative to first-order stochastic methods when the problem is ill-conditioned. We culminate with the introduction of an incremental method that exploits memory to achieve a superlinear convergence rate. This is the best known convergence rate for an incremental method.

An alternative strategy for lowering the prohibitive cost of solving large-scale ERM problems is decentralized optimization whereby samples are separated not across time but across multiple nodes of a network. In this regime, the main contribution of this thesis is in incorporating second-order information of the aggregate risk corresponding to samples of all nodes in the network in a way that can be implemented in a distributed fashion. We also explore the separation of samples across both, time and space, to reduce the computational and communication cost for solving large-scale ERM problems. We study this path by introducing a decentralized stochastic method which incorporates the idea of stochastic averaging gradient leading to a low computational complexity method with a fast linear convergence rate.

We then introduce a rethinking of ERM in which we consider not a partition of the training set as in the case of stochastic and distributed optimization, but a nested collection of subsets that we grow geometrically. The key insight is that the optimal argument associated with a training subset of a certain size is not that far from the optimal argument associated with a larger training subset. Based on this insight, we present adaptive sample size schemes which start with a small number of samples and solve the corresponding ERM

problem to its statistical accuracy. The sample size is then grown geometrically and use the solution of the previous ERM as a warm start for the new ERM. Theoretical analyses show that the use of adaptive sample size methods reduces the overall computational cost of achieving the statistical accuracy of the whole dataset for a broad range of deterministic and stochastic first-order methods. We further show that if we couple the adaptive sample size scheme with Newton's method, it is possible to consider subsequent doubling of the training set and perform a single Newton iteration in between. This is possible because of the interplay between the statistical accuracy and the quadratic convergence region of these problems and yields a method that is guaranteed to solve an ERM problem by performing just two passes over the dataset.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A large fraction of machine learning methods requires the solution of an empirical risk minimization problem (ERM) which is expressed as the minimization of a sum of individual costs associated with individual elements of a training set. For example, support vector machines intend to determine a hyperplane that separates samples with different labels by minimizing the average hinge loss associated with a given training set. In logistic regression, we aim to find a linear classifier that minimizes an average negative log likelihood probability computed based on a set of training samples. Dictionary learning aims at finding a dictionary in which some training data admits a sparse representation by solving a finite sum optimization problem. A feature common to modern versions of this class of problems is the very large size of the respective training sets.

As the objective function in the ERM problem is convex, any descent method can be utilized to solve this problem. However, an evident obstacle in using traditional descent algorithms for solving this class of problems is their prohibitive computation complexity which is proportional to the number of component functions in the ERM problem, i.e., the number of available samples in the given dataset. The main goal of this thesis is to study different approaches to solve large-scale ERM problems.

We first explore the idea of separating the training set into smaller subsets across time. In this scheme at each iteration, only a subset of samples – chosen either randomly or cyclically – is used to update the estimate for the minimizer of the empirical risk. When training examples are processed sequentially over time, we are in the realm of stochastic optimization methods. In this regime, the main contribution of this thesis is studying the application of quasi-Newton methods to accelerate the state-of-the-art first order methods by approximating the curvature information of the empirical risk.

The second direction considered in this thesis for training massive training sets is distributing samples to distinct nodes of a network, which we can interpret as separating samples over space. In this regime, the main contribution of this thesis is in incorporating

the second-order information of the aggregate risk associated with samples of all nodes in the network in a way that can be implemented in a distributed fashion, i.e., each node only exchanges information with its neighboring nodes to update its estimate of the global minimizer. Indeed, the idea of separating samples both across time and space can be executed to enhance the required computational and communication cost for solving large-scale empirical risk minimization problems. This path, which is also referred to as decentralized stochastic optimization, is studied in this thesis.

The third and last path considered in this thesis is a rethinking of ERM in which we consider not a partition of the training set as in the case of distributed and stochastic optimization, but a nested collection of subsets that grows geometrically. The key insight is that the optimal argument associated with a training subset of a certain size is not that far from the optimal argument corresponding to a larger training subset since the samples are drawn from a common (unknown) distribution. This means that solutions for an element of the geometric sequence can be used as warm starts for the solution of the subsequent element.

1.1 Context and background

To formally introduce the problem formulation used in this thesis, consider a decision vector $\mathbf{w} \in \mathbb{R}^p$, a random variable $\boldsymbol{\Theta} \in \mathbb{R}^d$ with realizations $\boldsymbol{\theta}$ and a convex loss function $f(\mathbf{w}, \boldsymbol{\theta})$. We aim to find the optimal argument that minimizes the optimization problem

$$\mathbf{w}^* := \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} F(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \mathbb{E}_{\boldsymbol{\Theta}}[f(\mathbf{w}, \boldsymbol{\Theta})] = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \int_{\boldsymbol{\Theta}} f(\mathbf{w}, \boldsymbol{\Theta}) P(d\boldsymbol{\theta}), \tag{1.1}$$

where $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\Theta}}[f(\mathbf{w}, \boldsymbol{\Theta})]$ is defined as the expected loss, and P is the probability distribution of the random variable $\boldsymbol{\Theta}$. The optimization problem in (1.1) cannot be solved since the distribution P is unknown. However, we have access to a training set $\mathcal{T} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$ containing N independent samples $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ drawn from P, and, therefore, we attempt to minimize the empirical loss associated with the training set $\mathcal{T} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$, which is equivalent to minimizing the problem

$$\mathbf{w}_N^\dagger := \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{w}, \boldsymbol{\theta}_i), \tag{1.2}$$

We refer to the loss $(1/N)\sum_{i=1}^{N} f(\mathbf{w}, \boldsymbol{\theta}_i)$ in (1.2) as the empirical loss associated to the training set $\mathcal{T}$. One may also interpret the ERM problem in (1.2), which typically appears in machine leaning applications [14, 15, 26, 108], as a finite sum minimization problem of N smooth and convex functions. In particular, if we define the component function $f_i : \mathbb{R}^p \to \mathbb{R}$

for $i = 1, \ldots, N$ as $f_i(\mathbf{w}) = f(\mathbf{w}, \boldsymbol{\theta}_i)$ and the global objective function f as the average of the component functions f_i, the optimization problem in (1.2) can be written as

$$\mathbf{w}^* := \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}}\, f(\mathbf{w}) := \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}}\, \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{w}). \tag{1.3}$$

Indeed, the problem formulation in (1.3), which is referred to as finite sum minimization, is more general, and ERM in (1.2) is a special case of the optimization problem in (1.3). This class of problems arises in many application domains such as control [22, 25, 56] and wireless communications [96, 97, 103].

In the first two parts of the thesis, we introduce a set of algorithms that can be executed to solve the more general setting in (1.3), and, of course, these approaches can be used to solve the ERM problem in (1.2). In the third part of the thesis, however, we study a novel approach for training large-scale ERM problems which is specifically designed for the setting in (1.2), and the results in this part of the thesis cannot be generalized to the finite sum minimization problem in (1.3).

In the following sections, we overview the state-of-the-art methods for solving large-scale ERM problems and overview the algorithms that are presented in this thesis to achieve faster convergence rates.

1.1.1 Stochastic methods

Classic deterministic optimization algorithms such as gradient descent, Newton's method, and quasi-Newton methods are built on the assumption that the gradient of the cost function is empirically computable. In the particular case of Newton's method, evaluation of the Hessian and its inverse should be feasible as well. However, the computational complexity of the gradient and Hessian evaluations, which are required to implement conventional descent methods, increase linearly by the size of the training set. Stochastic approximation methods are constructed on the idea of replacing gradients by their stochastic gradient approximations, arise as a natural alternative to overcome the high computational complexity of deterministic algorithms [15, 49, 82, 104, 107, 130]. However, the slow convergence time of SGD has limited its practical appeal and fostered the search for alternatives. In this regard, it has to be noted that SGD is slow because of both, the use of gradients as descent directions which is problematic in functions with large condition numbers, and the replacement of gradients by random estimates with potentially large variances. Alternatives to deal with both of these problems have been aggressively developed over the last decade.

Alternatives to reduce randomness in SGD have been proposed to render the convergence times of SGD closer to the convergence times of gradient descent. Some early methods make use of memory to either smooth iterates [88] or stochastic gradients [101]. More recent

developments have focused on hybrid approaches that use both, gradients and stochastic gradients, or update descent directions so that they become progressively closer to gradients [47, 104, 129]. Inasmuch as they succeed in reducing randomness, these algorithms end up exhibiting the asymptotic convergence rate of gradient descent which is faster than the asymptotic convergence rate of SGD. Although they improve asymptotic convergence rates, the latter methods are still often slow in practice. This is not unexpected. Reducing randomness is of no use when the function $F(\mathbf{w})$ has a challenging curvature profile. In these ill conditioned functions SGD is limited by the slow convergence times of (deterministic) gradient descent.

A parallel line of research has attempted to accelerate the convergence of SGD by correcting its curvature estimate. The natural solution to deal with these ill-conditioned functions is Newton's method, which makes use of the Hessian of the objective. However, stochastic estimates of Newton steps are not computationally cheap to compute. This issue motivated the use of stochastic quasi-Newton methods. In deterministic settings, quasi-Newton methods, which do not require computation of the objective Hessian and approximate the curvature using only gradient information, have been successful in achieving a super-linear convergence rate which outperforms the linear convergence rate of gradient descent [24, 32, 87, 89]. This success has resulted in the development of the stochastic quasi-Newton methods. An important observation here is that in trying to adapt to the changing curvature of the objective, stochastic quasi-Newton methods may end up exacerbating the problem. Indeed, since Hessian estimates are stochastic, it is possible to end up with almost singular Hessian estimates. The corresponding small eigenvalues then result in a catastrophic amplification of the noise which nullifies progress made towards convergence. This is not a minor problem. In oBFGS this possibility precludes convergence analyses [12, 105] and may result in erratic numerical behavior; see, e.g., Figure 2.12. As a matter of fact, the main motivation for the introduction of RES, presented in Chapter 2, is to avoid this catastrophic noise amplification so as to retain smaller convergence times while ensuring that optimal arguments are found almost surely [70]. However valuable, the convergence guarantees of RES are tainted by an iteration cost of order $O(p^3)$ which precludes its use in problems where p is very large. In deterministic settings this problem is addressed by limited memory (L)BFGS [55] which can be easily generalized to develop the oLBFGS algorithm [105]. Numerical tests of oLBFGS are promising but theoretical convergence characterizations are still lacking. The main contribution of Chapter 3 is to show that the sequence of iterates generated by oLBFGS converges with probability 1 to optimal arguments across realizations of the random variables, while its computational complexity is of order $O(p)$ [72].

Notwithstanding these stochastic quasi-Newton methods are successful in expanding the

application of quasi-Newton methods to stochastic settings and enhance the convergence time of SGD in ill-conditioned problems, the best-proven convergence rate for them is a sublinear rate. This slow asymptotic performance is inherited from the stochastic approximation of gradient which does not vanish even near the optimal solution without randomness reduction techniques.

Considering these two research trusts which handle the issues of randomness and curvature estimation in the update of SGD, it seems natural to combine the idea of randomness reduction and quasi-Newton methods to design a low computation cost method that recovers the superlinear convergence rate of deterministic quasi-Newton methods. The stochastic quasi-Newton methods in [58, 78] attempt to achieve this goal by using the variance reduction technique proposed in [45]; however, they can not achieve better than a linear convergence rate. In Chapter 4, the presented incremental quasi-Newton method (IQN) succeeds to achieve a superlinear convergence rate by incorporating the averages of variables, gradients, and Hessian approximations in conjunction with a corrected Taylor expansion for approximating the individual component functions [60, 61]. The IQN method is the first incremental quasi-Newton method to achieve a superlinear convergence rate.

1.1.2 Decentralized methods

Decentralized (Distributed) optimization algorithms are used to solve the problem of minimizing a global cost function over a set of nodes in situations where the objective function is defined as a sum of local functions. This class of algorithms can be used to solve the problem in (1.3) if the objective functions at nodes are the risk associated with a subset of samples. Specifically, consider a connected network of size V where each node v has access to a local objective function $f_v : \mathbb{R}^p \to \mathbb{R}$. The local objective function $f_v(\mathbf{w})$ is defined as the average risk of q_v functions (samples) $\{f_{v,i}(\mathbf{w})\}_{i=1}^{q_v}$ that can be individually evaluated at node v. Agents cooperate to solve the global optimization problem

$$\tilde{\mathbf{w}}^* := \operatorname*{argmin}_{\mathbf{w}} \sum_{v=1}^{V} f_v(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \sum_{v=1}^{V} \sum_{i=1}^{q_v} f_{v,i}(\mathbf{w}). \tag{1.4}$$

The formulation in (1.4) models a training set with a total of $N = \sum_{v=1}^{V} q_v$ training samples that are distributed among the V agents for parallel processing conducive to the determination of the optimal classifier $\tilde{\mathbf{w}}^*$ [6, 26, 118].

In all distributed methods the first step is replicating the decision variable $\mathbf{w}$ at each node. In other words, we consider the case that each node v has access to a local variable $\mathbf{w}_v$ and tries to achieve the minimum of its local objective functions $f_v(\mathbf{w}_v)$, while keeping its variable equal to the variables $\mathbf{w}_u$ of its neighbors $u \in \mathcal{N}_v$. This alternative formulation

can be written as

$$\{\mathbf{w}_v^*\}_{v=1}^V := \underset{\{\mathbf{w}_v\}_{v=1}^V}{\operatorname{argmin}} \sum_{v=1}^V f_v(\mathbf{w}_v), \quad \text{s.t. } \mathbf{w}_v = \mathbf{w}_u, \quad \text{for all } v, u \in \mathcal{N}_v. \tag{1.5}$$

Since the network is connected, the constraints $\mathbf{w}_v = \mathbf{w}_u$ for all v and $u \in \mathcal{N}_v$ imply that (1.4) and (1.5) are equivalent and we have $\mathbf{w}_v^* = \tilde{\mathbf{w}}^*$ for all v.

There are different algorithms to solve (1.5) in a distributed manner. The most popular choice is decentralized gradient descent (DGD) [80,126] and accelerated distributed gradient descent method [44, 90] which can be abstracted as combinations of local descent steps followed by variable exchanges and averaging of information among neighbors. Although the implementation of DGD is simple, its convergence could be arbitrary slow in ill-conditioned problems since as in other first-order methods it operates on gradient information only. This is not surprising because gradient descent methods in centralized settings where the aggregate function gradient is available at a single server (node) have the same difficulties in problems with skewed curvature.

This issue is addressed in centralized optimization by Newton's method that uses second order information to determine a descent direction adapted to the objective's curvature. However, second order methods are not practical in distributed settings since they require access to global information. In Chapter 5, an approximate Newton's method is presented to solve the problem in (1.5) in a distributed manner. It is done by introducing Network Newton (NN), a method that relies on distributed approximations of Newton steps for the global cost function $\sum_{v=1}^V f_v(\mathbf{w})$ to accelerate the convergence of DGD. The presented NN method approximates the Newton step of a penalized version of (1.5) by truncating the Taylor series of the exact Newton step. Convergence analysis of the NN method guarantees a global linear convergence method, while the rate of convergence is quadratic in a neighborhood of the optimal argument. This convergence property makes NN the first distributed algorithm that solves problem (1.5) faster than a linear rate when the functions are strongly convex and smooth.

The NN method is successful in improving the convergence rate of DGD; however, both of these methods solve a penalized version of (1.5) where the accuracy of the optimal solution depends on the penalty factor; see Chapter 5. This issue can be resolved by solving problem (1.5) in the dual domain. Dual domain methods build on the fact that the dual function of (1.5) has a gradient with separable structure. The use of plain dual gradient descent is possible but generally slow to converge [8, 95, 102]. In centralized optimization, better convergence speeds are attained by the method of multipliers (MM) that adds a quadratic augmentation term to the Lagrangian [7,40], or the proximal (P)MM that adds an additional

term to keep iterates close. In either case, the quadratic term that is added to construct the augmented Lagrangian makes distributed computation of primal gradients impossible. This issue is most often overcome with the use of decentralized (D) versions of the alternating direction method of multipliers (ADMM) [19,103,112]. Besides the ADMM, other methods that use different alternatives to approximate the gradients of the dual function have also been proposed [28,43,79,100,115,117,122]. The convergence rates of these methods have not been studied except for the DADMM and its variants that are known to converge linearly to the optimal argument when the local functions are strongly convex and their gradients are Lipschitz continuous [54, 77, 112]. An important observation here is that while all of these methods try to approximate the MM or the PMM, the performance penalty entailed by the approximation has not been studied.

In Chapter 6 an exact second order method (ESOM) is presented which uses quadratic approximations of the augmented Lagrangians of the problem in (1.5) and leads to a set of separable subproblems. As in the primal domain, computation of the dual function Hessian requires global communication and is impractical. ESOM overcomes this issue by using the Hessian inverse approximation technique introduced in the design of NN. The convergence analysis of ESOM as an approximation of the PMM shows that it converges at the same rate as PMM and the gap between the convergence guarantees of these methods decreases as the iterates approach the optimal argument. This indicates that the convergence paths of ESOM and PMM are very close, while PMM can not be implemented in a distributed fashion and ESOM is a distributed algorithm.

Although in decentralized optimization training samples are divided among different processors (nodes) to reduce the required computational capacity for each processor, there still could be cases that the number of assigned samples to each processor is beyond its computational capacity. To be more precise, all of the mentioned distributed algorithms require the computationally costly evaluation of the local gradients $\nabla f_v(\mathbf{w}) = \sum_{v=1}^{q_v} \nabla f_{v,i}(\mathbf{w})$. In the case that the number of assigned samples N/V to each processor is still large, the computation of local gradients might be beyond the computational capacity of each processor. In this regime, which is not seldom in *large-scale* optimization, we can reduce the computation cost by the use of stochastic decentralized algorithms that substitute the local gradients with their stochastic approximations. In other words, the stochastic decentralized methods separate samples across both time and space by training a subset of available samples at each processor.

The use of distributed stochastic methods reduces the computational cost per iteration of deterministic distributed algorithms but results in sublinear convergences rates of order $O(1/t)$ even if the corresponding deterministic algorithm exhibits linear convergence [9, 48, 92, 113]. This is a drawback that also exists in centralized stochastic optimization where

linear convergence rates in expectation are established by decreasing the variance of the stochastic gradient approximation [31, 45, 47, 49, 104, 109]. In Chapter 7, the decentralized double stochastic averaging gradient (DSA) method is presented which incorporates the idea of stochastic gradient averaging in [31] to approximate local gradients by a low computation cost gradient averaging [71,73]. This modification leads to a linear convergence rate for DSA which is the first decentralized stochastic method that achieves linear convergence rate.

1.1.3 Adaptive sample size algorithms

Perhaps most of the methods designed for solving large-scale ERM problems can be applied to solve finite sum minimization (FSM) problems, including the mentioned stochastic and decentralized methods. However, by focusing on designing methods for solving the general FSM problem we end up ignoring some fundamental properties of ERM which can be leveraged to solve ERM problems more efficiently. ERM problems have two specific qualities that come from the fact that ERM is a proxy for statistical loss minimization. The first property is that since the empirical risk and the statistical loss have different minimizers, there is no reason to solve ERM beyond the expected difference between the two objectives. This so-called *statistical accuracy* V_N is a constant of order $\mathcal{O}(1/N^\alpha)$ where α is a constant from the interval $[0.5, 1]$ depending on the regularity of the loss function. The second important property of ERM is that the component functions are drawn from a common distribution. This implies that if we consider subsets of the training set, the respective empirical risk functions are not that different from each other and, indeed, their differences are related to the statistical accuracy of the subset.

The relationship of ERM to statistical loss minimization suggests that ERM problems have more structure than FSM problems. This is not exploited by most existing methods which, albeit used for ERM, are in fact designed for FSM. The goal of adaptive sample size methods is to exploit the relationship between ERM and statistical loss minimization to achieve lower overall computational complexity for a broad class of first-order methods applied to ERM. The main idea of adaptive sample size methods is to start with a small subset of samples and solve the problem to within its statistical accuracy. Then, increase the size of training samples and use the current approximate optimal solution as a warm start for the new training set.

The technique presented in Chapters 8 and 9 uses subsamples of the training set containing $n \leq N$ component functions that we grow geometrically. In particular, we start by a small number of samples and minimize the corresponding empirical risk added by a regularization term of order V_n up to its statistical accuracy. Note that, based on the first property of ERM, the added adaptive regularization term does not modify the required accuracy while it makes the problem strongly convex and improves the problem condition

number. After solving the subproblem, we double the size of the training set and use the solution of the problem with n samples as a warm start for the problem with $2n$ samples. This is a reasonable initialization since based on the second property of ERM the functions are drawn from a joint distribution, and, therefore, the optimal values of the ERM problems with n and $2n$ functions are not that different from each other. The proposed approach succeeds in exploiting the two properties of ERM problems to improve complexity bounds of first-order methods and second-order methods.

1.2 Thesis outline and contributions

The first approach studied in this thesis in dealing with massive ERM problems is splitting samples across time. In Part I of the thesis, we explore this direction by studying the use of stochastic quasi-Newton methods for improving convergence rate of stochastic gradient descent. In this part, the challenges in designing convergent stochastic quasi-Newton methods have studied, as well as, the techniques to reduce the computational complexity of these methods. Finally, we close this part of the thesis by explaining the difficulties in designing a superlinearly convergent quasi-Newton method in the stochastic domain.

In Part II of the thesis, we explore the use of decentralized methods for solving large-scale ERM problems which can be also interpreted as splitting samples across space. In this part, we recap the state-of-the-art methods for solving distributed optimization problems and presents methods that approximate Newton step, both in primal and dual domains, to accelerate the convergence of first-order distributed methods. We further examine the idea of separating samples both in space and time to reduce the computational burden required at each node when we deal with massive datasets.

In Part III of the thesis, we introduce adaptive sample size methods for solving ERM problems which is constructed on the idea of starting with a small subset of samples and increasing the size of training set geometrically. We study the use of adaptive sample size scheme for both first-order and second-order methods and show that it leads to reduction in overall computational cost to achieve the statistical accuracy of the full dataset.

Chapter 2 opens the first part of the thesis on stochastic methods. In this chapter we introduce RES [68, 70] a stochastic regularized version of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method, to solve problems with the generic structure in (1.2). The proposed regularization avoids the near-singularity problems of more straightforward extensions and yields an algorithm with provable convergence guarantees when the functions are strongly convex. The fundamental idea of BFGS is to continuously satisfy a secant condition that captures information on the curvature of the function being minimized while staying close to previous curvature estimates. To regularize deterministic BFGS we retain the secant condition but modify the proximity condition so that eigenvalues of the Hessian

approximation matrix stay above a given threshold. This regularized version is leveraged to introduce the regularized stochastic BFGS algorithm. Regularized stochastic BFGS differs from standard BFGS in the use of a regularization to make a bound on the largest eigenvalue of the Hessian inverse approximation matrix and in the use of stochastic gradients in lieu of deterministic gradients for both, the determination of descent directions and the approximation of the objective function's curvature.

By proving lower and upper bounds on the approximate Hessians of the component functions it can be guaranteed that the sequence of iterates $\mathbf{w}^t$ genearaed by RES converges to the optimal argument $\mathbf{w}^*$ with probability 1 over realizations of the sample functions. This result indicates that RES is a globally convergent stochastic quasi-Newton method. We complement this result with a characterization of the convergence rate which is shown to be at least of order $O(1/t)$ in expectation. Advantages of RES relative to SGD are significant, as we establish in numerical results for the minimization of a family of quadratic objective functions of varying dimensionality and condition number. As we vary the condition number we observe that for well conditioned objectives RES and SGD exhibit comparable performance, whereas for ill conditioned functions RES outperforms SGD by an order of magnitude. As we vary problem dimension we observe that SGD becomes unworkable for large dimensional problems. RES however, exhibits manageable degradation as the number of iterations required for convergence doubles when the problem dimension increases by a factor of ten. We also study the performance of RES on an important example of ERM problems which are support vector machines (SVMs) [13, 15, 120]. We adapt RES for SVM problems and show the improvement relative to SGD in convergence time and stability through numerical analysis [69].

Chapter 3 studies the convergence properties and numerical behavior of the online Limited memory BFGS method proposed in [105], which can be considered as a practical variant of RES in Chapter 2. However valuable, the convergence guarantees of RES are tainted by an iteration cost of order $O(p^3)$ which precludes its use in problems where p is very large. In deterministic settings this problem is addressed by limited memory (L)BFGS [55] which can be easily generalized to develop the oLBFGS algorithm [105]. The fundamental idea in BFGS and oLBFGS is to continuously satisfy a secant condition while staying close to previous curvature estimates. They differ in that BFGS uses all past gradients to estimate curvature while oLBFGS uses a fixed moving window of past gradients. The use of this window reduces memory and computational cost. The difference between LBGS and oLBFGS is the use of stochastic gradients in lieu of their deterministic counterparts.

In this chapter we present the oLBFGS algorithm and show that it converges with probability 1 to optimal arguments across realizations of the random variables [72]. This is the same convergence guarantee provided for RES, while the computational coplexity of

oLBFGS per iteration is of order $O(p)$. Convergence guarantees for oLBFGS do not require such measures. To do so, we first show that under the assumption that the component functions are strongly convex and Lipschitz continuous the trace and determinant of the Hessian approximations computed by oLBFGS are upper and lower bounded, respectively. These bounds are then used to limit the range of variation of the ratio between the Hessian approximations' largest and smallest eigenvalues. In turn, this condition number limit is shown to be sufficient to prove convergence to the optimal argument $\mathbf{w}^*$ with probability 1 over realizations of the sample functions. We complement this almost sure convergence result with a characterization of the convergence rate which is shown to be at least sublinear of order $O(1/t)$ in expectation.

To show the advantage of using oLBFGS as an adaptive reconditioning strategy we develop its application to the training of SVMs and perform a comparative numerical analysis with synthetic data. The conclusions of this numerical analysis are that oLBFGS performs as well as oBFGS and RES while outperforming SGD when convergence is measured with respect to the number of feature vectors processed. In terms of computation time, oLBFGS outperforms all three methods, SGD, and RES. The advantages of oLBFGS grow with the dimension of the feature vector and can be made arbitrarily large. To further substantiate numerical claims we use oLBFGS to train a logistic regressor to predict the click through rate in a search engine advertising problem. The logistic regression uses a heterogeneous feature vector with 174,026 binary entries that describe the user, the search, and the advertisement. Being a large scale problem with heterogeneous data, the condition number of the logistic log likelihood objective is large and we expect to see significant advantages of oLBFGS relative to SGD. This expectation is fulfilled. The oLBFGS algorithm trains the regressor using less than 1% of the data required by SGD to obtain similar classification accuracy.

Chapter 4 presents the incremental quasi-Newton (IQN) method which has the unique properties of low computational complexity and superlinear convergence rate. Note that the issue of proving convergence of stochastic quasi-Newton methods is tackled by RES and oLBFGS in Chapters 2 and 3, respectively. Although these methods are successful in expanding the application of quasi-Newton methods to stochastic settings, their convergence rate is sublinear. This is not better than the convergence rate of SGD and, as is also the case in SGD, is a consequence of the stochastic approximation noise which necessitates the use of diminishing stepsizes. The stochastic quasi-Newton methods in [58, 78] resolve this issue by using the variance reduction technique proposed in [45]. The fundamental idea of the work in [45] is to reduce the noise of the stochastic gradient approximation by computing the exact gradient in an outer loop to use it in an inner loop for gradient approximation. The methods in [58, 78], which incorporate the variance reduction scheme presented in [45] into

the update of quasi-Newton methods, are successful in achieving a linear convergence rate. Note that the use of variance reduction in stochastic quasi-Newton methods led to linear convergence and did not recover a superlinear rate. Hence, a fundamental question remains unanswered: Is it possible to design an incremental quasi-Newton method that recovers the superlinear convergence rate of deterministic quasi-Newton algorithms? In Chapter 4, we show that the answer to this open problem is positive by proposing an incremental quasi-Newton method (IQN) with a local superlinear convergence rate [60, 61]. This is the first quasi-Newton method to achieve superlinear convergence while having a per iteration cost independent of the number of functions N – the cost per iteration is of order $\mathcal{O}(p^2)$.

There are three major differences between the IQN method and state-of-the-art incremental (stochastic) quasi-Newton methods that lead to the former's superlinear convergence rate. First, the proposed IQN method uses the aggregated information of variables, gradients, and Hessian approximation matrices to reduce the noise of approximation for both gradients and Hessian approximation matrices. This is different to the variance-reduced stochastic quasi-Newton methods in [58, 78] that attempt to reduce only the noise of gradient approximations. Second, in IQN the index of the updated function is chosen in a cyclic fashion, rather than the random selection scheme used in the incremental methods in [23, 70, 72, 105]. The cyclic routine in IQN allows to bound the error at each iteration as a function of the errors of the last N iterates, something that is not possible when using a random scheme. To explain the third and most important difference we point out that the form of quasi-Newton updates is the solution of a local second order Taylor approximation of the objective. It is possible to understand stochastic quasi-Newton methods as an analogous approximation of individual sample functions. However, it turns out that the state-of-the-art stochastic quasi-methods evaluate the linear and quadratic terms of the Taylor's expansion at different points yielding and inconsistent approximation (Remark 4.7). The IQN method utilizes a consistent Taylor series which yields a more involved update which we nonetheless show can be implemented with the same computational cost. These three properties together lead to an incremental quasi-Newton method with a local superlinear convergence rate.

Chapter 5 launches the second part of the thesis on the use of decentralized (distributed) optimization methods for solving large-scale ERM problems on a network using multiple processors (nodes). There are different algorithms to solve (1.4) in a distributed manner. The most popular choices are decentralized gradient descent (DGD) [44, 80, 111, 126], distributed implementations of the alternating direction method of multipliers [19, 27, 77, 103, 112], and decentralized dual averaging [33, 119]. Although there are substantial differences between them, these methods can be generically abstracted as combinations of local descent steps followed by variable exchanges and averaging of information

among neighbors. A feature common to all of these algorithms is the slow convergence rate in ill-conditioned problems since they operate on first order information only. This is not surprising because gradient descent methods in centralized settings where the aggregate function gradient is available at a single server have the same difficulties in problems with skewed curvature.

The goal of Chapter 5 is to accelerate the convergence rate of DGD in solving (1.4) by incorporating the second information of the objective objective function. In achieving this goal, the first step is to reinterpret DGD as an algorithm that utilizes gradient descent to solve a penalized version of (1.5) which is a distributed representation of the problem in (1.4). This reinterpretation explains linear convergence of DGD to a neighborhood of the optimal argument $\mathbf{w}^*$. The volume of this neighborhood is given by the relative weight of the penalty function and the original objective which is controlled by a penalty coefficient.

If DGD uses gradient descent to solve the penalized objective function, it seems natural to use Newton's method to achieves faster convergence. Alas, distributed computation of Newton steps for the penalized problem requires global communication between all nodes in the network and is therefore impractical (Section 5.3). To resolve this issue, the presented Network Newton (NN) method approximates the Newton step of the penalized objective function by truncating the Taylor series of the exact Newton step [66]. This approximation results in a family of methods indexed by the number of terms of the Taylor expansion that are kept in the approximation. The method that results from keeping K of these terms is termed NN-K. A fundamental observation here is that the Hessian of the penalized function has a sparsity structure that is the same sparsity pattern of the graph. Thus, when computing terms in the Hessian inverse expansion, the first order term is as sparse as the graph, the second term is as sparse as the two hop neighborhood, and, in general, the k-th term is as sparse as the k-hop neighborhood of the graph. Thus, implementation of the NN-K method requires aggregating information from K hops away. Increasing K makes NN-K arbitrarily close to Newton's method at the cost of increasing the communication overhead of each iteration.

In the convergence analysis of NN, it is shown that a measure of the error between the Hessian inverse approximation utilized by NN-K and the actual inverse Hessian decays exponentially with the method index K. This exponential decrease hints that using a small value of K should suffice in practice. Convergence analysis of NN shows that its convergence rate is at least linear. It follows from this convergence analysis that larger penalty coefficients result in faster convergence that comes at the cost of increasing the distance between the optimal solutions of the original and penalized objectives. Further, it is shown that for all iterations except the first few, a weighted gradient norm associated with NN-K iterates follows a decreasing path akin to the path that would be followed by

Newton iterates. The only difference between these residual paths is that the NN-K path contains a term that captures the error of the Hessian inverse approximation. Leveraging this similarity, it is possible to show that the rate of convergence is quadratic in a specific interval whose length depends on the order K of the selected network Newton method (Theorem 8). Existence of this quadratic convergence phase explains why NN-K methods converge faster than DGD. It is also worth remarking that the error in the Hessian inverse approximation can be made arbitrarily small by increasing the method's order K and, as a consequence, the quadratic phase can be made arbitrarily large.

Chapter 6 continues development of distributed method for solving ERM problems by extending the idea of using second-order information into primal-dual methods which have exact convergence. This extension is necessary since the NN method in Chapter 5 solves a penalized version of (1.5) where the accuracy of the optimal solution depends on the penalty factor. Among the methods that solve problem (1.5) in the dual domain the method of multipliers (MM) that adds a quadratic augmentation term to the Lagrangian of the problem [7, 40], or the proximal (P)MM that adds an additional term to keep iterates close have the best performance. However, in either case, the quadratic term that is added to construct the augmented Lagrangian makes distributed computation of primal gradients impossible. This chapter studies the exact second order method (ESOM) which uses quadratic approximations of the augmented Lagrangians of (1.5) to approximate the PMM which leads to a set of separable subproblems [74,76]. The ESOM algorithm is second-order since it uses a quadratic approximation of the augmented Lagrangian in the primal update. It is exact since, as the PMM, it solves the augmented Lagrangian and converges to the exact solution of (1.5). And it is distributed since the update of ESOM can be implemented locally and without the need for global communication.

ESOM can also be interpreted as a variation of PMM that substitutes the proximal augmented Lagrangian with its quadratic approximation. Implementation of ESOM requires computing the inverse of the Hessian of the proximal augmented Lagrangian. Since this inversion cannot be computed using local and neighboring information, ESOM-K approximates the Hessian inverse with the K-order truncation of the Taylor's series expansion of the Hessian inverse, as explained in the derivation of NN in Chapter 5. This expansion can be carried out using an inner loop of local operations.

A remarkable property of all ESOM-K methods is that they can be shown to pay a performance penalty relative to (centralized) PMM that vanishes with increasing iterations. To be more specific, in this chapter we establish linear convergence of (centralized) PMM and use its linear convergence factor as a benchmark for methods that can be implemented in a distributed manner. We then prove linear convergence of ESOM and to show that ESOM's linear convergence factor approaches the corresponding PMM factor as the iterates

approach the optimal solution. This indicates that the convergence paths of (distributed) ESOM-K and (centralized) PMM are very close.

Chapter 7 closes the second part of the thesis by presenting a novel stochastic distributed method for solving large-scale optimization problems. Decentralized optimization method lead to reducing the computational complexity required by processors to train massive ERM problems through distributing samples over multiple processors and diving the computational cost among them. However, the possibility of facing scenarios that operating on assigned samples to each processor is beyond its computational capacity is not rare in big-data problems. This is a valid claim since distributed optimization methods rely on the computation of local gradients $\nabla f_v(\mathbf{w}) = \sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{w})$ at nodes. As in centralized case, a natural solution would be splitting samples over time and using stochastic gradients in lieu of local gradients. The use of stochastic decentralized methods reduces the computational cost per iteration but results in sublinear convergences rates of order $O(1/t)$ even if the corresponding deterministic algorithm exhibits linear convergence. Our interest here is in solving (1.4) with a method that is decentralized, stochastic, and has a linear convergence rate.

In this chapter, we achieve this goal by presenting the decentralized double stochastic averaging gradient (DSA) method [71, 73]. The method exploits a new interpretation of EXTRA as a saddle point method and uses stochastic averaging gradients in lieu of gradients. DSA is *decentralized* because it is implementable in a network setting where nodes can communicate only with their neighbors. It is *double* because iterations utilize the information of two consecutive iterates. It is *stochastic* because the gradient of only one randomly selected function is evaluated at each iteration and it is an *averaging* method because it uses an average of stochastic gradients to approximate the local gradients. DSA is proven to converge linearly to the optimal argument $\tilde{\mathbf{x}}^*$ in expectation when the local instantaneous functions $f_{v,i}$ are strongly convex, with Lipschitz continuous gradients. This is in contrast to all other decentralized stochastic methods that converge at sublinear rates.

Chapter 8 launches the third part of the thesis which introduces the idea of adaptive sample size methods for solving large-scale ERM problems. In particular, in this chapter we explain that most (if not all) iterative stochastic and decentralized methods aim to solve general finite sum minimization (FSM) problems by either sampling over time or space. This approach leads to ignoring some critical properties of ERM as a particular case of FSM. The first property is that since the empirical risk and the statistical loss have different minimizers, there is no reason to solve ERM beyond the expected difference between the two objectives. This so-called *statistical accuracy* takes the place of ϵ in the complexity orders of the previous paragraph and is a constant of order $\mathcal{O}(1/N^{\alpha})$ where α is a constant from the interval $[0.5, 1]$ depending on the regularity of the loss function.

The second important property of ERM is that the component functions are drawn from a common distribution. This implies that if we consider subsets of the training set, the respective empirical risk functions are not that different from each other and, indeed, their differences are related to the statistical accuracy of the subset.

The relationship of ERM to statistical loss minimization suggests that ERM problems have more structure than FSM problems. This is not exploited by most existing methods which, albeit used for ERM, are in fact designed for FSM. The goal of this part of the thesis is to exploit the relationship between ERM and statistical loss minimization to achieve lower overall computational complexity for a broad class of first-order methods applied to ERM. The technique we propose uses subsamples of the training set containing $n \leq N$ component functions that we grow geometrically. In particular, we start by a small number of samples and minimize the corresponding empirical risk added by a regularization term of order V_n up to its statistical accuracy. Note that, based on the first property of ERM, the added adaptive regularization term does not modify the required accuracy while it makes the problem strongly convex and improves the problem condition number. After solving the subproblem, we double the size of the training set and use the solution of the problem with n samples as a warm start for the problem with $2n$ samples. This is a reasonable initialization since based on the second property of ERM the functions are drawn from a joint distribution, and, therefore, the optimal values of the ERM problems with n and $2n$ functions are not that different from each other. The proposed approach succeeds in exploiting the two properties of ERM problems to improve complexity bounds of first-order methods. In particular, we show that to reach the statistical accuracy of the full training set the adaptive sample size scheme reduces the overall computational complexity of a broad range of first-order methods by a factor of $\log(N^\alpha)$. For instance, the overall computational complexity of adaptive sample size AGD to reach the statistical accuracy of the full training set is of order $\mathcal{O}(N\sqrt{\kappa})$ which is lower than $\mathcal{O}((N\sqrt{\kappa})\log(N^\alpha))$ complexity of AGD.

Chapter 9 extends the idea of adaptive sample size methods into second-order Newton's method [59]. In this chapter we attempt to circumvent the challenges in the implementation of Newton's method for ERM with the Ada Newton algorithm that combines the use of Newton iterations with adaptive sample sizes. Say the total number of available samples is N, consider subsets of $n \leq N$ samples, and suppose the statistical accuracy of the ERM associated with n samples is V_n. In Ada Newton we add a quadratic regularization term of order V_n to the empirical risk – so that the regularized risk also has statistical accuracy V_n – and assume that for a certain initial sample size m_0, the problem has been solved to its statistical accuracy V_{m_0}. The sample size is then increased by a factor $\alpha > 1$ to $n = \alpha m_0$. We proceed to perform a single Newton iteration with unit stepsize and prove that the result of this update solves this extended ERM problem to its statistical accuracy. This

permits a second increase of the sample size by a factor α and a second Newton iteration that is likewise guaranteed to solve the problem to its statistical accuracy. Overall, this permits minimizing the empirical risk in $\alpha/(\alpha-1)$ passes over the dataset and inverting $\log_\alpha N$ Hessians. Our theoretical results provide a characterization of the values of α that are admissible with respect to different problem parameters. In particular, we show that asymptotically on the number of samples n and with proper parameter selection we can set $\alpha = 2$. In such case we can optimize to within statistical accuracy in about 2 passes over the dataset and after inversion of about $3.32 \log_{10} N$ Hessians. Our numerical experiments verify that $\alpha = 2$ is a valid factor for increasing the size of the training set at each iteration while performing a single Newton iteration for each value of the sample size.

Chapter 10 closes the thesis with concluding remarks.

Part I

Stochastic (Incremental) Quasi-Newton Methods

Chapter 2

Regularized stochastic BFGS algorithm

2.1 Context and background

Stochastic optimization algorithms are used to solve the problem of optimizing an objective function over a set of feasible values in situations where the objective function is defined as an expectation over a set of random functions. To be precise, consider an optimization variable $\mathbf{w} \in \mathbb{R}^p$ and a random variable $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$ that determines the choice of a function $f(\mathbf{w}, \boldsymbol{\theta}) : \mathbb{R}^{p \times d} \to \mathbb{R}$. The stochastic optimization problems considered in this paper entail determination of the argument $\mathbf{w}^*$ that minimizes the expected value $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$,

$$\mathbf{w}^* \ := \ \underset{\mathbf{w}}{\operatorname{argmin}} \, \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})] \ := \ \underset{\mathbf{w}}{\operatorname{argmin}} \, F(\mathbf{w}). \tag{2.1}$$

We refer to $f(\mathbf{w}, \boldsymbol{\theta})$ as the random or instantaneous functions and to $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ as the average function. We assume that the instantaneous functions $f(\mathbf{w}, \boldsymbol{\theta})$ are strongly convex for all $\boldsymbol{\theta}$ from where it follows that the average function $F(\mathbf{w})$ is also strongly convex. Problems having the form in (2.1) are common in machine learning [14, 15, 108] as well as in optimal resource allocation in wireless systems [67, 96, 97].

Note that the the empirical risk minimization (ERM) problem in (1.2) can be considered as a specific case of the problem formulation in (2.1). In order to clarify this connection assume that the distribution of the random variable $\boldsymbol{\theta}$ is uniform over the training set $\{\boldsymbol{\theta}_i\}_{i=1}^N$. Therefore, the optimization problem in (2.1) reduces to

$$\mathbf{w}^* \ := \ \underset{\mathbf{w}}{\operatorname{argmin}} \, \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{w}, \boldsymbol{\theta}_i), \tag{2.2}$$

which is equivalent to the ERM problem in (1.2). In this chapter we focus on the general setting in (2.1), and, of course, the results hold for the ERM problem in (2.2).

Since the objective function of (2.1) is strongly convex, descent algorithms can be used for its minimization. However, descent methods require exact determination of the objective function's gradient $\nabla_{\mathbf{w}} F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\nabla_{\mathbf{w}} f(\mathbf{w}, \boldsymbol{\theta})]$, which is intractable in general. Stochastic gradient descent (SGD) methods overcome this issue by using unbiased gradient estimates based on small data samples and are the workhorse methodology used to solve large-scale stochastic optimization problems [15, 49, 82, 104, 107, 130]. Practical appeal of SGD methods remains limited, however, because they require a large number of iterations to converge. Indeed, SGD inherits slow convergence from its use of gradients which is aggravated by their replacement with stochastic estimates.

Alternatives to reduce randomness in SGD have been proposed to render the convergence times of SGD closer to the convergence times of gradient descent. Some early methods make use of memory to either smooth iterates [88] or stochastic gradients [101]. More recent developments have focused on hybrid approaches that use both, gradients and stochastic gradients, or update descent directions so that they become progressively closer to gradients [47, 104, 129]. Inasmuch as they succeed in reducing randomness, these algorithms end up exhibiting the asymptotic convergence rate of gradient descent which is faster than the asymptotic convergence rate of SGD. Although they improve asymptotic convergence rates, the latter methods are still often slow in practice. This is not unexpected. Reducing randomness is of no use when the function $F(\mathbf{w})$ has a challenging curvature profile. In these ill conditioned functions SGD is limited by the slow convergence times of (deterministic) gradient descent.

To overcome problems with the objective function's curvature, one may think of developing stochastic versions of Newton's method. However, computing unbiased estimates of Newton steps is not easy except in problems with some specific structures [10, 128]. Recourse to quasi-Newton methods then arises as a natural alternative because they can achieve superlinear convergence rates in deterministic settings while relying on gradients to compute curvature estimates [24, 32, 87, 89]. Considering the fact that unbiased gradient estimates are computable at manageable cost, stochastic generalizations of quasi-Newton methods are not difficult to devise [12, 67, 105]. Numerical tests of these methods on simple quadratic objectives suggest that stochastic quasi-Newton methods retain the convergence rate advantages of their deterministic counterparts [105]. The success of these preliminary experiments notwithstanding, Hessian estimations based on random stochastic gradients may result in near singular curvature estimates. The possibility of having singular curvature estimates makes it impossible to provide convergence analyses for stochastic quasi-Newton methods [12, 105] and may result in erratic numerical behavior (see Section 2.5.2).

In this chapter we introduce a stochastic regularized version of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method to solve problems with the generic structure in (2.1). The proposed regularization avoids the near-singularity problems of more straightforward extensions and yields an algorithm with provable convergence guarantees when the functions $f(\mathbf{w}, \boldsymbol{\theta})$ are strongly convex. We begin the paper with a brief discussion of SGD (Section 2.2) and deterministic BFGS (Section 2.2.1). The fundamental idea of BFGS is to continuously satisfy a secant condition that captures information on the curvature of the function being minimized while staying close to previous curvature estimates. To regularize deterministic BFGS we retain the secant condition but modify the proximity condition so that eigenvalues of the Hessian approximation matrix stay above a given threshold (Section 2.2.1). This regularized version is leveraged to introduce the regularized stochastic BFGS algorithm (Section 2.2.2). Regularized stochastic BFGS differs from standard BFGS in the use of a regularization to make a bound on the largest eigenvalue of the Hessian inverse approximation matrix and in the use of stochastic gradients in lieu of deterministic gradients for both, the determination of descent directions and the approximation of the objective function's curvature. We abbreviate regularized stochastic BFGS as RES.

Convergence properties of RES are then analyzed (Section 2.3). We prove that lower and upper bounds on the Hessians of the sample functions $f(\mathbf{w}, \boldsymbol{\theta})$ are sufficient to guarantee convergence of a subsequence to the optimal argument $\mathbf{w}^*$ with probability 1 over realizations of the sample functions (Theorem 1). We complement this result with a characterization of the convergence rate which is shown to be at least of order $O(1/t)$ in expectation (Theorem 2). This expected convergence rate is typical of stochastic optimization algorithms and, in that sense, no better than SGD [82]. Advantages of RES relative to SGD are nevertheless significant, as we establish in numerical results for the minimization of a family of quadratic objective functions of varying dimensionality and condition number (Section 2.4). As we vary the condition number we observe that for well conditioned objectives RES and SGD exhibit comparable performance, whereas for ill conditioned functions RES outperforms SGD by an order of magnitude (Section 2.4.1). As we vary problem dimension we observe that SGD becomes unworkable for large dimensional problems. RES however, exhibits manageable degradation as the number of iterations required for convergence doubles when the problem dimension increases by a factor of ten (Section 2.4.4).

An important example of a class of problems having the form in (2.1) are support vector machines (SVMs) that reduce binary classification to the determination of a hyperplane that separates points in a given training set; see, e.g., [13, 15, 120]. We adapt RES for SVM problems (Section 2.5) and show the improvement relative to SGD in convergence time and stability through numerical analysis (Section 2.5.1). For this particular problem of finding optimal SVM classifiers, several accelerations of SGD have been proposed. These

include the Stochastic Average Gradient (SAG) method [104], the Semi-Stochastic Gradient Descent (S2GD) algorithm [47], and Stochastic Approximation by Averaging (SAA) [88]. The comparison of RES with these accelerated versions yields the expected conclusion. SAG and S2GD accelerate the convergence of SGD but still underperform RES for problems that are not well conditioned. As we commented above, RES solves a different problem than the one targeted by SAG, S2GD, and SAA. The latter attempt to reduce the randomness in SGD to make the convergence rate closer to that of gradient descent. RES attempts to adapt to the curvature of the objective function. We also compare RES to standard (non-regularized) stochastic BFGS. The regularization in RES is fundamental in guaranteeing convergence as standard (non-regularized) stochastic BFGS is observed to routinely fail in the computation of a separating hyperplane.

Notation Lowercase boldface $\mathbf{v}$ denotes a vector and uppercase boldface $\mathbf{A}$ a matrix. We use $\|\mathbf{v}\|$ to denote the Euclidean norm of vector $\mathbf{v}$ and $\|\mathbf{A}\|$ to denote the Euclidean norm of matrix $\mathbf{A}$. The trace of $\mathbf{A}$ is written as $\text{tr}(\mathbf{A})$ and the determinant as $\det(\mathbf{A})$. We use $\mathbf{I}$ for the identity matrix of appropriate dimension. The notation $\mathbf{A} \succeq \mathbf{B}$ implies that the matrix $\mathbf{A} - \mathbf{B}$ is positive semidefinite. The operator $\mathbb{E}_{\mathbf{x}}[\cdot]$ stands in for expectation over random variable $\mathbf{x}$ and $\mathbb{E}[\cdot]$ for expectation with respect to the distribution of a stochastic process.

2.2 Algorithm definition

Recall the definitions of the sample functions $f(\mathbf{w}, \boldsymbol{\theta})$ and the average function $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$. Since the function $F(\mathbf{w})$ is strongly convex, we can find the optimal argument $\mathbf{w}^*$ in (2.1) with a gradient descent algorithm. Considering that strongly convex functions are continuously differentiable and further assuming that the instantaneous functions $f(\mathbf{w}, \boldsymbol{\theta})$ have finite gradients it follows that the gradients of $F(\mathbf{w})$ are given by

$$\mathbf{s}(\mathbf{w}) := \nabla F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\nabla f(\mathbf{w}, \boldsymbol{\theta})]. \tag{2.3}$$

When the number of functions $f(\mathbf{w}, \boldsymbol{\theta})$ is large, as is the case in most problems of practical interest, exact evaluation of the gradient $\mathbf{s}(\mathbf{w})$ is impractical. This motivates the use of stochastic gradients in lieu of actual gradients. More precisely, consider a given set of L realizations $\tilde{\boldsymbol{\theta}} = [\boldsymbol{\theta}_1; ...; \boldsymbol{\theta}_L]$ and define the stochastic gradient of $F(\mathbf{w})$ at $\mathbf{w}$ given samples $\tilde{\boldsymbol{\theta}}$ as

$$\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) := \frac{1}{L} \sum_{l=1}^{L} \nabla f(\mathbf{w}, \boldsymbol{\theta}_l). \tag{2.4}$$

Introducing now a time index t, an initial iterate $\mathbf{w}_0$, and a step size sequence ϵ_t, a stochastic gradient descent algorithm is defined by the iteration

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t \, \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t). \tag{2.5}$$

To implement (2.5) we compute stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ using (2.4). In turn, this requires determination of the gradients of the random functions $f(\mathbf{w}, \boldsymbol{\theta}_{tl})$ for each $\boldsymbol{\theta}_{tl}$ component of $\tilde{\boldsymbol{\theta}}_t$ and their corresponding average. The computational cost is manageable for small values of L.

The stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ in (2.4) is an unbiased estimate of the (average) gradient $\mathbf{s}(\mathbf{w})$ in (2.3) in the sense that $\mathbb{E}_{\tilde{\boldsymbol{\theta}}}[\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})] = \mathbf{s}(\mathbf{w})$. Thus, the iteration in (2.5) is such that, on average, iterates descend along a negative gradient direction, see, e.g., [82]. This intuitive observation can be formalized into a proof of convergence when the step size sequence is selected as nonsummable but square summable, i.e.,

$$\sum_{t=0}^{\infty} \epsilon_t = \infty, \quad \text{and} \quad \sum_{t=0}^{\infty} \epsilon_t^2 < \infty. \tag{2.6}$$

A customary step size choice for which (2.6) holds is to make $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$, for given parameters ϵ_0 and T_0 that control the initial step size and its speed of decrease, respectively. Convergence notwithstanding, the number of iterations required to approximate $\mathbf{w}^*$ is very large in problems that don't have small condition numbers [49]. This motivates the alternative methods we discuss in subsequent sections.

2.2.1 Regularized BFGS

To speed up convergence of (2.5) resort to second order methods is of little use because evaluating Hessians of the objective function is computationally intensive. A better suited methodology is the use of quasi-Newton methods whereby gradient descent directions are premultiplied by a matrix $\mathbf{B}_t^{-1}$,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t \, \mathbf{B}_t^{-1} \mathbf{s}(\mathbf{w}_t). \tag{2.7}$$

The idea is to select positive definite matrices $\mathbf{B}_t \succ 0$ close to the Hessian of the objective function $\mathbf{H}(\mathbf{w}_t) := \nabla^2 F(\mathbf{w}_t)$. Various methods are known to select matrices $\mathbf{B}_t$, including those by Broyden e.g., [21]; Davidon, Fletcher, and Powell (DFP) [35]; and Broyden, Fletcher, Goldfarb, and Shanno (BFGS) e.g., [24, 87, 89]. We work here with the matrices $\mathbf{B}_t$ used in BFGS since they have been observed to work best in practice [24].

In BFGS – and all other quasi-Newton methods – the function's curvature is approximated by a finite difference. Specifically, define the variable and gradient variations at time

t as

$$\mathbf{v}_t := \mathbf{w}_{t+1} - \mathbf{w}_t, \quad \text{and} \quad \mathbf{r}_t := \mathbf{s}(\mathbf{w}_{t+1}) - \mathbf{s}(\mathbf{w}_t), \tag{2.8}$$

respectively, and select the matrix $\mathbf{B}_{t+1}$ to be used in the next time step so that it satisfies the secant condition $\mathbf{B}_{t+1}\mathbf{v}_t = \mathbf{r}_t$. The rationale for this selection is that the Hessian $\mathbf{H}(\mathbf{w}_t)$ satisfies this condition for $\mathbf{w}_{t+1}$ tending to $\mathbf{w}_t$. Notice however that the secant condition $\mathbf{B}_{t+1}\mathbf{v}_t = \mathbf{r}_t$ is not enough to completely specify $\mathbf{B}_{t+1}$. To resolve this indeterminacy, matrices $\mathbf{B}_{t+1}$ in BFGS are also required to be as close as possible to $\mathbf{B}_t$ in terms of the Gaussian differential entropy,

$$\begin{aligned} \mathbf{B}_{t+1} \;=\; & \underset{\mathbf{Z}}{\operatorname{argmin}} \;\; \operatorname{tr}\left[\mathbf{B}_t^{-1}\mathbf{Z}\right] - \log\det\left[\mathbf{B}_t^{-1}\mathbf{Z}\right] - p, \\ & \text{s.t.} \qquad \mathbf{Z}\mathbf{v}_t = \mathbf{r}_t, \quad \mathbf{Z} \succeq \mathbf{0}. \end{aligned} \tag{2.9}$$

The constraint $\mathbf{Z} \succeq \mathbf{0}$ in (2.9) restricts the feasible space to positive semidefinite matrices whereas the constraint $\mathbf{Z}\mathbf{v}_t = \mathbf{r}_t$ requires $\mathbf{Z}$ to satisfy the secant condition. The objective $\operatorname{tr}(\mathbf{B}_t^{-1}\mathbf{Z}) - \log\det(\mathbf{B}_t^{-1}\mathbf{Z}) - p$ represents the differential entropy between random variables with zero-mean Gaussian distributions $\mathcal{N}(\mathbf{0}, \mathbf{B}_t)$ and $\mathcal{N}(\mathbf{0}, \mathbf{Z})$ having covariance matrices $\mathbf{B}_t$ and $\mathbf{Z}$. The differential entropy is nonnegative and equal to zero if and only if $\mathbf{Z} = \mathbf{B}_t$. The solution $\mathbf{B}_{t+1}$ of the semidefinite program in (2.9) is therefore closest to $\mathbf{B}_t$ in the sense of minimizing the Gaussian differential entropy among all positive semidefinite matrices that satisfy the secant condition $\mathbf{Z}\mathbf{v}_t = \mathbf{r}_t$.

Strongly convex functions are such that the inner product of the gradient and variable variations is positive, i.e., $\mathbf{v}_t^T\mathbf{r}_t > 0$. In that case the matrix $\mathbf{B}_{t+1}$ in (2.9) is explicitly given by the update – see, e.g., [87] and the proof of Lemma 1 –,

$$\mathbf{B}_{t+1} = \mathbf{B}_t + \frac{\mathbf{r}_t\mathbf{r}_t^T}{\mathbf{v}_t^T\mathbf{r}_t} - \frac{\mathbf{B}_t\mathbf{v}_t\mathbf{v}_t^T\mathbf{B}_t}{\mathbf{v}_t^T\mathbf{B}_t\mathbf{v}_t}. \tag{2.10}$$

In principle, the solution to (2.9) could be positive semidefinite but not positive definite, i.e., we can have $\mathbf{B}_{t+1} \succeq \mathbf{0}$ but $\mathbf{B}_{t+1} \not\succ \mathbf{0}$. However, through direct operation in (2.10) it is not difficult to conclude that $\mathbf{B}_{t+1}$ stays positive definite if the matrix $\mathbf{B}_t$ is positive definite. Thus, initializing the curvature estimate with a positive definite matrix $\mathbf{B}_0 \succ \mathbf{0}$ guarantees $\mathbf{B}_t \succ \mathbf{0}$ for all subsequent times t. Still, it is possible for the smallest eigenvalue of $\mathbf{B}_t$ to become arbitrarily close to zero which means that the largest eigenvalue of $\mathbf{B}_t^{-1}$ can become arbitrarily large. This has been proven not to be an issue in BFGS implementations but is a significant challenge in the stochastic version proposed here.

To avoid this problem we introduce a regularization of (2.9) to enforce the eigenvalues of $\mathbf{B}_{t+1}$ to exceed a positive constant δ. Specifically, we redefine $\mathbf{B}_{t+1}$ as the solution of

problem,

$$\begin{aligned}\mathbf{B}_{t+1} = &\underset{\mathbf{Z}}{\text{argmin}} \;\; \text{tr}\left[\mathbf{B}_t^{-1}(\mathbf{Z}-\delta\mathbf{I})\right] - \log\det\left[\mathbf{B}_t^{-1}(\mathbf{Z}-\delta\mathbf{I})\right] - p, \\ &\text{s. t.} \qquad \mathbf{Z}\mathbf{v}_t = \mathbf{r}_t, \quad \mathbf{Z} \succeq \mathbf{0}. \end{aligned} \tag{2.11}$$

The curvature approximation matrix $\mathbf{B}_{t+1}$ defined in (2.11) still satisfies the secant condition $\mathbf{B}_{t+1}\mathbf{v}_t = \mathbf{r}_t$ but has a different proximity requirement since instead of comparing $\mathbf{B}_t$ and $\mathbf{Z}$ we compare $\mathbf{B}_t$ and $\mathbf{Z}-\delta\mathbf{I}$. While (2.11) does not ensure that all eigenvalues of $\mathbf{B}_{t+1}$ exceed δ we can show that this will be the case under two minimally restrictive assumptions. We do so in the following proposition where we also give an explicit solution for (2.11) analogous to the expression in (2.10) that solves the non regularized problem in (2.9).

Proposition 1 *Consider the semidefinite program in* (2.11) *where the matrix* $\mathbf{B}_t \succ \mathbf{0}$ *is positive definite and define the corrected gradient variation*

$$\tilde{\mathbf{r}}_t := \mathbf{r}_t - \delta\mathbf{v}_t, \tag{2.12}$$

where $\delta > 0$ *is a constant. If the inner product* $\tilde{\mathbf{r}}_t^T\mathbf{v}_t = (\mathbf{r}_t - \delta\mathbf{v}_t)^T\mathbf{v}_t$ *is positive, the solution* $\mathbf{B}_{t+1}$ *of* (2.11) *is such that all eigenvalues of* $\mathbf{B}_{t+1}$ *are larger than* δ*,*

$$\mathbf{B}_{t+1} \succeq \delta\mathbf{I}. \tag{2.13}$$

Furthermore, $\mathbf{B}_{t+1}$ *is explicitly given by the expression*

$$\mathbf{B}_{t+1} = \mathbf{B}_t + \frac{\tilde{\mathbf{r}}_t\tilde{\mathbf{r}}_t^T}{\mathbf{v}_t^T\tilde{\mathbf{r}}_t} - \frac{\mathbf{B}_t\mathbf{v}_t\mathbf{v}_t^T\mathbf{B}_t}{\mathbf{v}_t^T\mathbf{B}_t\mathbf{v}_t} + \delta\mathbf{I}. \tag{2.14}$$

Proof: We first show that (2.14) is true. Since the optimization problem in (2.11) is convex in $\mathbf{Z}$ we can determine the optimal variable $\mathbf{B}_{t+1} = \mathbf{Z}^*$ using Lagrangian duality. Introduce then the multiplier variable $\boldsymbol{\mu}$ associated with the secant constraint $\mathbf{Z}\mathbf{v}_t = \mathbf{r}_t$ in (2.11) and define the Lagrangian

$$\mathcal{L}(\mathbf{Z}, \boldsymbol{\mu}) \;= \text{tr}(\mathbf{B}_t^{-1}(\mathbf{Z}-\delta\mathbf{I})) - \log\det(\mathbf{B}_t^{-1}(\mathbf{Z}-\delta\mathbf{I})) - p + \boldsymbol{\mu}^T\left(\mathbf{Z}\mathbf{v}_t - \mathbf{r}_t\right). \tag{2.15}$$

The dual function is defined as $d(\boldsymbol{\mu}) := \min_{\mathbf{Z}\succeq\mathbf{0}} \mathcal{L}(\mathbf{Z}, \boldsymbol{\mu})$ and the optimal dual variable is $\boldsymbol{\mu}^* := \text{argmin}_{\boldsymbol{\mu}}\, d(\boldsymbol{\mu})$. We define the primal Lagrangian minimizer associated with dual variable $\boldsymbol{\mu}$ as

$$\mathbf{Z}(\boldsymbol{\mu}) := \underset{\mathbf{Z}\succeq\mathbf{0}}{\text{argmin}}\, \mathcal{L}(\mathbf{Z}, \boldsymbol{\mu}). \tag{2.16}$$

Observe that combining the definitions in (2.16) and (2.15) we can write the dual function

$d(\boldsymbol{\mu})$ as

$$\begin{aligned} d(\boldsymbol{\mu}) &= \mathcal{L}(\mathbf{Z}(\boldsymbol{\mu}), \boldsymbol{\mu}) \\ &= \operatorname{tr}(\mathbf{B}_t^{-1}(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})) - \log\det(\mathbf{B}_t^{-1}(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})) - p + \boldsymbol{\mu}^T\left(\mathbf{Z}(\boldsymbol{\mu})\mathbf{v}_t - \mathbf{r}_t\right). \end{aligned} \quad (2.17)$$

We will determine the optimal Hessian approximation $\mathbf{Z}^* = \mathbf{Z}(\boldsymbol{\mu}^*)$ as the Lagrangian minimizer associated with the optimal dual variable $\boldsymbol{\mu}^*$. To do so we first find the Lagrangian minimizer (2.16) by nulling the gradient of $\mathcal{L}(\mathbf{Z}, \boldsymbol{\mu})$ with respect to $\mathbf{Z}$ in order to show that $\mathbf{Z}(\boldsymbol{\mu})$ must satisfy

$$\mathbf{B}_t^{-1} - (\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})^{-1} + \frac{\boldsymbol{\mu}\mathbf{v}_t^T + \mathbf{v}_t\boldsymbol{\mu}^T}{2} = 0. \quad (2.18)$$

Multiplying the equality in (2.18) by $\mathbf{B}_t$ from the right and rearranging terms it follows that the inverse of the argument of the log-determinant function in (2.17) can be written as

$$(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})^{-1}\mathbf{B}_t = \mathbf{I} + \left(\frac{\boldsymbol{\mu}\mathbf{v}_t^T + \mathbf{v}_t\boldsymbol{\mu}^T}{2}\right)\mathbf{B}_t. \quad (2.19)$$

If, instead, we multiply (2.18) by $(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})$ from the right it follows after rearranging terms that

$$\mathbf{B}_t^{-1}(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I}) = \mathbf{I} - \frac{\boldsymbol{\mu}\mathbf{v}_t^T + \mathbf{v}_t\boldsymbol{\mu}^T}{2}(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I}). \quad (2.20)$$

Further considering the trace of both sides of (2.20) and noting that $\operatorname{tr}(\mathbf{I}) = n$ we can write the trace in (2.17) as

$$\operatorname{tr}(\mathbf{B}_t^{-1}(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})) = p - \operatorname{tr}\Big[\frac{\boldsymbol{\mu}\mathbf{v}_t^T + \mathbf{v}_t\boldsymbol{\mu}^T}{2}(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})\Big]. \quad (2.21)$$

Observe now that since the trace of a product is invariant under cyclic permutations of its arguments and the matrix $\mathbf{Z}$ is symmetric we have $\operatorname{tr}[\boldsymbol{\mu}\mathbf{v}_t^T(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})] = \operatorname{tr}[\mathbf{v}\boldsymbol{\mu}_t^T(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})] = \operatorname{tr}[\boldsymbol{\mu}^T(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})\mathbf{v}_t]$. Since the argument in the latter is a scalar the trace operation is inconsequential from where it follows that we can rewrite (2.21) as

$$\operatorname{tr}(\mathbf{B}_t^{-1}(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})) = n - \boldsymbol{\mu}^T(\mathbf{Z}(\boldsymbol{\mu}) - \delta\mathbf{I})\mathbf{v}_t. \quad (2.22)$$

Observing that the log-determinant of a matrix is the opposite of the log-determinant of its inverse we can substitute (2.19) for the argument of the log-determinant in (2.17). Further substituting (2.22) for the trace in (2.17) and rearranging terms yields the explicit expression for the dual function

$$d(\boldsymbol{\mu}) = \log\det\left[\mathbf{I} + \Big(\frac{\boldsymbol{\mu}\mathbf{v}_t^T + \mathbf{v}_t\boldsymbol{\mu}^T}{2}\Big)\mathbf{B}_t\right] - \boldsymbol{\mu}^T(\mathbf{r}_t - \delta\mathbf{v}_t). \quad (2.23)$$

In order to compute the optimal dual variable $\boldsymbol{\mu}^*$ we set the gradient of (2.23) to zero and manipulate terms to obtain

$$\boldsymbol{\mu}^* = \frac{1}{\tilde{\mathbf{r}}_t^T \mathbf{v}_t} \left(\mathbf{v}_t \left(1 + \frac{\tilde{\mathbf{r}}_t^T \mathbf{B}_t^{-1} \tilde{\mathbf{r}}_t}{\tilde{\mathbf{r}}_t^T \mathbf{v}_t} \right) - 2\mathbf{B}_t^{-1} \tilde{\mathbf{r}}_t \right), \tag{2.24}$$

where we have used the definition of the corrected gradient variation $\tilde{\mathbf{r}}_t := \mathbf{r}_t - \delta \mathbf{v}_t$. To complete the derivation plug the expression for the optimal multiplier $\boldsymbol{\mu}^*$ in (2.24) into the Lagrangian minimizer expression in (2.18) and regroup terms so as to write

$$(\mathbf{Z}(\boldsymbol{\mu}^*) - \delta \mathbf{I})^{-1} = \frac{\mathbf{v}_t \mathbf{v}_t^T}{\tilde{\mathbf{r}}_t^T \mathbf{v}_t} + \left(\mathbf{I} - \frac{\mathbf{v}_t \tilde{\mathbf{r}}_t^T}{\tilde{\mathbf{r}}_t^T \mathbf{v}_t} \right) \mathbf{B}_t^{-1} \left(\mathbf{I} - \frac{\tilde{\mathbf{r}}_t \mathbf{v}_t^T}{\tilde{\mathbf{r}}_t^T \mathbf{v}_t} \right). \tag{2.25}$$

Applying the Sherman-Morrison formula to compute the inverse of the right hand side of (2.25) leads to

$$\mathbf{Z}(\boldsymbol{\mu}^*) - \delta \mathbf{I} = \mathbf{B}_t + \frac{\tilde{\mathbf{r}}_t \tilde{\mathbf{r}}_t^T}{\mathbf{v}_t^T \tilde{\mathbf{r}}_t} - \frac{\mathbf{B}_t \mathbf{v}_t \mathbf{v}_t^T \mathbf{B}_t}{\mathbf{v}_t^T \mathbf{B}_t \mathbf{v}_t}, \tag{2.26}$$

which can be verified by direct multiplication. The result in (2.14) follows after solving (2.26) for $\mathbf{Z}(\boldsymbol{\mu}^*)$ and noting that for the convex optimization problem in (2.11) we must have $\mathbf{Z}(\boldsymbol{\mu}^*) = \mathbf{Z}^* = \mathbf{B}_{t+1}$ as we already argued.

To prove (2.13) we operate directly from (2.14). Consider first the term $\tilde{\mathbf{r}}_t \tilde{\mathbf{r}}_t^T / \mathbf{v}_t^T \tilde{\mathbf{r}}_t$ and observe that since the hypotheses include the condition $\mathbf{v}_t^T \tilde{\mathbf{r}}_t > 0$, we must have

$$\frac{\tilde{\mathbf{r}}_t \tilde{\mathbf{r}}_t^T}{\mathbf{v}_t^T \tilde{\mathbf{r}}_t} \succeq \mathbf{0}. \tag{2.27}$$

Consider now the term $\mathbf{B}_t - \mathbf{B}_t \mathbf{v}_t \mathbf{v}_t^T \mathbf{B}_t / \mathbf{v}_t^T \mathbf{B}_t \mathbf{v}_t$ and factorize $\mathbf{B}_t^{1/2}$ from the left and right side so as to write

$$\mathbf{B}_t - \frac{\mathbf{B}_t \mathbf{v}_t \mathbf{v}_t^T \mathbf{B}_t}{\mathbf{v}_t^T \mathbf{B}_t \mathbf{v}_t} = \mathbf{B}_t^{1/2} \left(\mathbf{I} - \frac{\mathbf{B}_t^{1/2} \mathbf{v}_t \mathbf{v}_t^T \mathbf{B}_t^{1/2}}{\mathbf{v}_t^T \mathbf{B}_t \mathbf{v}_t} \right) \mathbf{B}_t^{1/2} \tag{2.28}$$

Define the vector $\mathbf{u}_t := \mathbf{B}_t^{1/2} \mathbf{v}_t$ and write $\mathbf{v}_t^T \mathbf{B}_t \mathbf{v}_t = (\mathbf{B}_t^{1/2} \mathbf{v}_t)^T (\mathbf{B}_t^{1/2} \mathbf{v}_t) = \mathbf{u}_t^T \mathbf{u}_t$ as well as $\mathbf{B}_t^{1/2} \mathbf{v}_t \mathbf{v}_t^T \mathbf{B}_t^{1/2} = \mathbf{u}_t \mathbf{u}_t^T$. Substituting these observation into (2.28) we can conclude that

$$\mathbf{B}_t - \frac{\mathbf{B}_t \mathbf{v}_t \mathbf{v}_t^T \mathbf{B}_t}{\mathbf{v}_t^T \mathbf{B}_t \mathbf{v}_t} = \mathbf{B}_t^{1/2} \left(\mathbf{I} - \frac{\mathbf{u}_t \mathbf{u}_t^T}{\mathbf{u}_t^T \mathbf{u}_t} \right) \mathbf{B}_t^{1/2} \succeq \mathbf{0}, \tag{2.29}$$

because the eigenvalues of the matrix $\mathbf{u}_t \mathbf{u}_t^T / \mathbf{u}_t^T \mathbf{u}_t$ belong to the interval $[0, 1]$. The only term in (2.14) which has not been considered is $\delta \mathbf{I}$. Since the rest add up to a positive semidefinite matrix it then must be that (2.13) is true. ■

Comparing (2.10) and (2.14) follows that the differences between BFGS and regularized BFGS are the replacement of the gradient variation $\mathbf{r}_t$ in (2.8) by the corrected variation $\tilde{\mathbf{r}}_t := \mathbf{r}_t - \delta \mathbf{v}_t$ and the addition of the regularization term $\delta \mathbf{I}$. We use (2.14) in the construction of the stochastic BFGS in the following section.

2.2.2 RES: Regularized stochastic BFGS

As can be seen from (2.14) the regularized BFGS curvature estimate $\mathbf{B}_{t+1}$ is obtained as a function of previous estimates $\mathbf{B}_t$, iterates $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$, and corresponding gradients $\mathbf{s}(\mathbf{w}_t)$ and $\mathbf{s}(\mathbf{w}_{t+1})$. We can then think of a method in which gradients $\mathbf{s}(\mathbf{w}_t)$ are replaced by stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ in both, the curvature approximation update in (2.14) and the descent iteration in (2.7). Specifically, start at time t with current iterate $\mathbf{w}_t$ and let $\hat{\mathbf{B}}_t$ stand for the Hessian approximation computed by stochastic BFGS in the previous iteration. Obtain a batch of samples $\tilde{\boldsymbol{\theta}}_t = [\boldsymbol{\theta}_{t1}; ...; \boldsymbol{\theta}_{tL}]$, determine the value of the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ as per (2.4), and update the iterate $\mathbf{w}_t$ as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t \left(\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I} \right) \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t), \tag{2.30}$$

where we added the identity bias term $\Gamma \mathbf{I}$ for a given positive constant $\Gamma > 0$. Relative to SGD as defined by (2.5), RES as defined by (2.30) differs in the use of the matrix $\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I}$ to account for the curvature of $F(\mathbf{w})$. Relative to (regularized or non regularized) BFGS as defined in (2.7) RES differs in the use of stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ instead of actual gradients and in the use of the curvature approximation $\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I}$ in lieu of $\mathbf{B}_t^{-1}$. Observe that in (2.30) we add a bias $\Gamma \mathbf{I}$ to the curvature approximation $\hat{\mathbf{B}}_t^{-1}$. This is necessary to ensure convergence by hedging against random variations in $\hat{\mathbf{B}}_t^{-1}$ as we discuss in Section 2.3.

To update the Hessian approximation $\hat{\mathbf{B}}_t$ compute the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ associated with the *same* set of samples $\tilde{\boldsymbol{\theta}}_t$ used to compute the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. Define then the stochastic gradient variation at time t as

$$\hat{\mathbf{r}}_t := \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t), \tag{2.31}$$

and redefine $\tilde{\mathbf{r}}_t$ so that it stands for the modified stochastic gradient variation

$$\tilde{\mathbf{r}}_t := \hat{\mathbf{r}}_t - \delta \mathbf{v}_t, \tag{2.32}$$

by using $\hat{\mathbf{r}}_t$ instead of $\mathbf{r}_t$. The Hessian approximation $\hat{\mathbf{B}}_{t+1}$ for the next iteration is defined as the matrix that satisfies the stochastic secant condition $\mathbf{Z}\mathbf{v}_t = \hat{\mathbf{r}}_t$ and is closest to $\hat{\mathbf{B}}_t$ in

Algorithm 1 RES: Regularized Stochastic BFGS

Require: Variable $\mathbf{w}_0$. Hessian approximation $\hat{\mathbf{B}}_0 \succ \delta \mathbf{I}$.

1: **for** $t = 0, 1, 2, \ldots$ **do**

2: Acquire L independent samples $\tilde{\boldsymbol{\theta}}_t = [\boldsymbol{\theta}_{t1}, \ldots, \boldsymbol{\theta}_{tL}]$

3: Compute $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ [cf. (2.4)] $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) = \dfrac{1}{L} \displaystyle\sum_{l=1}^{L} \nabla_{\mathbf{w}} f(\mathbf{w}_t, \boldsymbol{\theta}_{tl})$.

4: Descend along direction $(\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I})\, \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ [cf. (2.30)]

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t \, (\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I})\, \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t).$$

5: Compute $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ [cf. (2.4)] $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) = \dfrac{1}{L} \displaystyle\sum_{l=1}^{L} \nabla_{\mathbf{w}} f(\mathbf{w}_{t+1}, \boldsymbol{\theta}_{tl})$.

6: Compute variable variation [cf. (2.8)] $\mathbf{v}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$.

7: Compute modified stochastic gradient variation [cf. (2.32)]

$$\tilde{\mathbf{r}}_t = \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) - \delta \mathbf{v}_t.$$

8: Update Hessian approximation matrix [cf. (2.33)]

$$\hat{\mathbf{B}}_{t+1} = \hat{\mathbf{B}}_t + \frac{\tilde{\mathbf{r}}_t \tilde{\mathbf{r}}_t^T}{\mathbf{v}_t^T \tilde{\mathbf{r}}_t} - \frac{\hat{\mathbf{B}}_t \mathbf{v}_t \mathbf{v}_t^T \hat{\mathbf{B}}_t}{\mathbf{v}_t^T \hat{\mathbf{B}}_t \mathbf{v}_t} + \delta \mathbf{I}.$$

9: **end for**

the sense of (2.11). As per Proposition 1 we can compute $\hat{\mathbf{B}}_{t+1}$ explicitly as

$$\hat{\mathbf{B}}_{t+1} = \hat{\mathbf{B}}_t + \frac{\tilde{\mathbf{r}}_t \tilde{\mathbf{r}}_t^T}{\mathbf{v}_t^T \tilde{\mathbf{r}}_t} - \frac{\hat{\mathbf{B}}_t \mathbf{v}_t \mathbf{v}_t^T \hat{\mathbf{B}}_t}{\mathbf{v}_t^T \hat{\mathbf{B}}_t \mathbf{v}_t} + \delta \mathbf{I}. \tag{2.33}$$

as long as $(\hat{\mathbf{r}}_t - \delta \mathbf{v}_t)^T \mathbf{v}_t = \tilde{\mathbf{r}}^T \mathbf{v}_t > 0$. Conditions to guarantee that $\tilde{\mathbf{r}}_t^T \mathbf{v}_t > 0$ are introduced in Section 2.3.

The resulting RES algorithm is summarized in Algorithm 3. The two core steps in each iteration are the descent in Step 4 and the update of the Hessian approximation $\hat{\mathbf{B}}_t$ in Step 8. Step 2 comprises the observation of L samples that are required to compute the stochastic gradients in Steps 3 and 5. The stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ in Step 3 is used in the descent iteration in Step 4. The stochastic gradient of Step 3 along with the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ of Step 5 are used to compute the variations in steps 6 and 7 that permit carrying out the update of the Hessian approximation $\hat{\mathbf{B}}_t$ in Step 8. Iterations are initialized at arbitrary variable $\mathbf{w}_0$ and positive definite matrix $\hat{\mathbf{B}}_0$ with the smallest eigenvalue larger than δ.

Remark 1 One may think that the natural substitution of the gradient variation $\mathbf{r}_t = \mathbf{s}(\mathbf{w}_{t+1}) - \mathbf{s}(\mathbf{w}_t)$ is the stochastic gradient variation $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_{t+1}) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ instead of the variation $\hat{\mathbf{r}}_t = \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ in (2.31). This would have the advantage that

$\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_{t+1})$ is the stochastic gradient used to descend in iteration $t+1$ whereas $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ is not and is just computed for the purposes of updating $\mathbf{B}_t$. Therefore, using the variation $\hat{\mathbf{r}}_t = \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ requires twice as many stochastic gradient evaluations as using the variation $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_{t+1}) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. However, the use of the variation $\hat{\mathbf{r}}_t = \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is necessary to ensure that $(\hat{\mathbf{r}}_t - \delta \mathbf{v}_t)^T \mathbf{v}_t = \tilde{\mathbf{r}}_t^T \mathbf{v}_t > 0$, which in turn is required for (2.33) to be true. This cannot be guaranteed if we use the variation $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_{t+1}) - \hat{\mathbf{s}}(\mathbf{w}_t \tilde{\boldsymbol{\theta}}_t)$ – see Lemma 1 for details. The same observation holds true for the non-regularized version of stochastic BFGS introduced in [105].

2.3 Convergence analysis of RES

For the subsequent analysis we define the instantaneous objective function associated with samples $\tilde{\boldsymbol{\theta}} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L]$ as

$$\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) := \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{w}, \boldsymbol{\theta}_l). \tag{2.34}$$

The definition of the instantaneous objective function $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ in association with the fact that $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ implies

$$F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})]. \tag{2.35}$$

Our goal here is to show that as time progresses the sequence of variable iterates $\mathbf{w}_t$ approaches the optimal argument $\mathbf{w}^*$. In proving this result we make the following assumptions.

Assumption 1 The instantaneous functions $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ are twice differentiable and the eigenvalues of the instantaneous Hessian $\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) = \nabla^2_{\mathbf{w}} \hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ are bounded between constants $0 < \tilde{m}$ and $\tilde{M} < \infty$ for all random variables $\tilde{\boldsymbol{\theta}}$,

$$\tilde{m}\mathbf{I} \;\preceq\; \hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) \;\preceq\; \tilde{M}\mathbf{I}. \tag{2.36}$$

Assumption 2 The second moment of the norm of the stochastic gradient is bounded for all $\mathbf{w}$. i.e., there exists a constant S^2 such that for all variables $\mathbf{w}$ it holds

$$\mathbb{E}_{\boldsymbol{\theta}}\big[\|\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\|^2 \,\big|\, \mathbf{w}_t\big] \leq S^2. \tag{2.37}$$

Assumption 3 The regularization constant δ is smaller than the smallest Hessian eigenvalue $\tilde{m}$, i.e., $\delta < \tilde{m}$.

As a consequence of Assumption 1 similar eigenvalue bounds hold for the (average) function $F(\mathbf{w})$. Indeed, it follows from the linearity of the expectation operator and the expression in (2.35) that the Hessian is $\nabla^2_{\mathbf{w}} F(\mathbf{w}) = \mathbf{H}(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})]$. Combining this observation with the bounds in (2.36) it follows that there are constants $m \geq \tilde{m}$ and $M \leq \tilde{M}$ such that

$$\tilde{m}\mathbf{I} \preceq m\mathbf{I} \preceq \mathbf{H}(\mathbf{w}) \preceq M\mathbf{I} \preceq \tilde{M}\mathbf{I}. \tag{2.38}$$

The bounds in (2.38) are customary in convergence proofs of descent methods. For the results here the stronger condition spelled in Assumption 1 is needed. The lower bound implies strong convexity of instantaneous functions and the upper bound is equivalent to them having Lipschitz Continuous gradients. The restriction imposed by Assumption 2 is typical of stochastic descent algorithms, its intent being to limit the random variation of stochastic gradients [82]. Assumption 3 is necessary to guarantee that the inner product $\tilde{\mathbf{r}}_t^T \mathbf{v}_t = (\mathbf{r}_t - \delta \mathbf{v}_t)^T \mathbf{v}_t > 0$ [cf. Proposition 1] is positive as we show in the following lemma.

Lemma 1 *Consider the modified stochastic gradient variation* $\tilde{\mathbf{r}}_t$ *defined in* (2.32) *and the variable variation* $\mathbf{v}_t$ *defined in* (2.8)*. Let Assumption 1 hold and recall the lower bound* $\tilde{m}$ *on the smallest eigenvalue of the instantaneous Hessians. Then, for all constants* $0 < \delta < \tilde{m}$ *it holds*

$$\tilde{\mathbf{r}}_t^T \mathbf{v}_t \;=\; (\hat{\mathbf{r}}_t - \delta \mathbf{v}_t)^T \mathbf{v}_t \;\geq (\tilde{m} - \delta)\|\mathbf{v}_t\|^2 \;>\; 0. \tag{2.39}$$

Proof: As per (2.36) in Assumption 1 the eigenvalues of the instantaneous Hessian $\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ are bounded below by $\tilde{m} > 0$ which is equivalent to say that instantaneous objective functions $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ associated with samples $\tilde{\boldsymbol{\theta}}$ are m-strongly convex with respect to $\mathbf{w}$. Considering the strong monotonicity of gradients for the m-strongly convex functions $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}}_t)$, we can write

$$\Big[\nabla \hat{f}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \nabla \hat{f}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\Big]^T (\mathbf{w}_{t+1} - \mathbf{w}_t) \geq \tilde{m}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2. \tag{2.40}$$

Observing the definitions of stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ in (2.4) and instantaneous objective functions $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ in (2.34) it follows that $\nabla \hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) = \hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$. Hence, we can rewrite (3.35) as

$$\Big(\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}})\Big)^T (\mathbf{w}_{t+1} - \mathbf{w}_t) \geq \tilde{m}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2. \tag{2.41}$$

Using the definitions of stochastic gradient variation $\hat{\mathbf{r}}_t$ and variable variation $\mathbf{v}_t$ in (2.31) and (2.8) we further simplify (3.37) to

$$\hat{\mathbf{r}}_t^T \mathbf{v}_t \;\geq\; \tilde{m}\|\mathbf{v}_t\|^2. \tag{2.42}$$

Consider now the inner product $\tilde{\mathbf{r}}_t^T \mathbf{v}_t \;=\; (\hat{\mathbf{r}}_t - \delta \mathbf{v}_t)^T \mathbf{v}_t$ in (3.33) and use the bound in (2.42) to write

$$\tilde{\mathbf{r}}_t^T \mathbf{v}_t \;=\; \hat{\mathbf{r}}_t^T \mathbf{v}_t - \delta \mathbf{v}_t^T \mathbf{v}_t \;\geq\; (\tilde{m} - \delta) \|\mathbf{v}_t\|^2.$$

Since we are selecting $\delta < \tilde{m}$ by hypothesis it follows that (3.33) is true for all times t. ■

Initializing the curvature approximation matrix $\hat{\mathbf{B}}_0 \succ \delta \mathbf{I}$, which implies $\hat{\mathbf{B}}_0^{-1} \succ \mathbf{0}$, and setting $\delta < \tilde{m}$ it follows from Lemma 1 that the hypotheses of Proposition 1 are satisfied for $t = 0$. Hence, the matrix $\hat{\mathbf{B}}_1$ computed from (2.33) is the solution of the semidefinite program in (2.11) and, more to the point, satisfies $\hat{\mathbf{B}}_1 \succ \delta \mathbf{I}$, which in turn implies $\hat{\mathbf{B}}_1^{-1} \succ \mathbf{0}$. Proceeding recursively we can conclude that $\hat{\mathbf{B}}_t \succ \delta \mathbf{I} \succ \mathbf{0}$ for all times $t \geq 0$. Equivalently, this implies that all the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$ are between 0 and $1/\delta$ and that, as a consequence, the matrix $\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I}$ is such that

$$\Gamma \mathbf{I} \;\preceq\; \hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I} \;\preceq\; (\Gamma + (1/\delta))\, \mathbf{I}. \tag{2.43}$$

Having matrices $\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I}$ that are strictly positive definite with eigenvalues uniformly upper bounded by $\Gamma + (1/\delta)$ leads to the conclusion that if $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is a descent direction, the same holds true of $(\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I})\, \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. The stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is not a descent direction in general, but we know that this is true for its conditional expectation $\mathbb{E}[\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) \,\big|\, \mathbf{w}_t] = \nabla_{\mathbf{w}} F(\mathbf{w}_t)$. Therefore, we conclude that $(\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I})\, \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is an average descent direction because $\mathbb{E}[(\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I})\, \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) \,\big|\, \mathbf{w}_t] = (\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I})\, \nabla_{\mathbf{w}} F(\mathbf{w}_t)$. Stochastic optimization algorithms whose displacements $\mathbf{w}_{t+1} - \mathbf{w}_t$ are descent directions on average are expected to approach optimal arguments in a sense that we specify formally in the following lemma.

Lemma 2 *Consider the RES algorithm as defined by* (2.30)-(2.33)*. If assumptions 1, 2 and 3 hold true, the sequence of average function $F(\mathbf{w}_t)$ satisfies*

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\big|\, \mathbf{w}_t\right] \leq F(\mathbf{w}_t) - \epsilon_t \Gamma \|\nabla F(\mathbf{w}_t)\|^2 + K \epsilon_t^2 \tag{2.44}$$

where the constant $K := MS^2(1/\delta + \Gamma)^2/2$.

Proof: As it follows from Assumption 1 the eigenvalues of the Hessian $\mathbf{H}(\mathbf{w}_t) = \nabla_{\mathbf{w}}^2 F(\mathbf{w}_t)$ are bounded between $0 < m$ and $M < \infty$ as stated in (2.38). Taking a Taylor's expansion of the dual function $F(\mathbf{w})$ around $\mathbf{w} = \mathbf{w}_t$ and using the upper bound in the Hessian eigenvalues we can write

$$F(\mathbf{w}_{t+1}) \leq F(\mathbf{w}_t) + \nabla F(\mathbf{w}_t)^T (\mathbf{w}_{t+1} - \mathbf{w}_t) + \frac{M}{2} \|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \tag{2.45}$$

From the definition of the RES update in (2.30) we can write the difference of two consecutive variables $\mathbf{w}_{t+1} - \mathbf{w}_t$ as $-\epsilon_t(\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I})\ \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. Making this substitution in (2.45), taking expectation with $\mathbf{w}_t$ given in both sides of the resulting inequality, and observing the fact that when $\mathbf{w}_t$ is given the Hessian approximation $\hat{\mathbf{B}}_t^{-1}$ is deterministic we can write

$$\begin{aligned}\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] \leq\ & F(\mathbf{w}_t) - \epsilon_t \nabla F(\mathbf{w}_t)^T (\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I}) \mathbb{E}\left[\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) \,\middle|\, \mathbf{w}_t\right] \\ & + \frac{\epsilon^2 M}{2} \mathbb{E}\left[\left\|(\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I})\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\right\|^2 \,\middle|\, \mathbf{w}_t\right]. \end{aligned} \tag{2.46}$$

We proceed to bound the third term in the right hand side of (2.46). Start by observing that the 2-norm of a product is not larger than the product of the 2-norms and that, as noted above, with $\mathbf{w}_t$ given the matrix $\hat{\mathbf{B}}_t^{-1}$ is also given to write

$$\mathbb{E}\left[\left\|\left(\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I}\right)\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\right\|^2 \,\middle|\, \mathbf{w}_t\right] \leq \left\|\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I}\right\|^2 \mathbb{E}\left[\left\|\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\right\|^2 \,\middle|\, \mathbf{w}_t\right]. \tag{2.47}$$

Notice that, as stated in (2.43), $\Gamma + 1/\delta$ is an upper bound for the eigenvalues of $\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I}$. Further observe that the second moment of the norm of the stochastic gradient is bounded by $\mathbb{E}\left[\|\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\|^2 \,\middle|\, \mathbf{w}_t\right] \leq S^2$, as stated in Assumption 2. These two upper bounds substituted in (2.47) yield

$$\mathbb{E}\left[\left\|\left(\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I}\right)\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\right\|^2 \,\middle|\, \mathbf{w}_t\right] \leq S^2(1/\delta + \Gamma)^2. \tag{2.48}$$

Substituting the upper bound in (2.48) for the third term of (2.46) and further using the fact that $\mathbb{E}\left[\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) \,\middle|\, \mathbf{w}_t\right] = \nabla F(\mathbf{w}_t)$ in the second term leads to

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] \leq F(\mathbf{w}_t) - \epsilon_t \nabla F(\mathbf{w}_t)^T \left[\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I}\right] \nabla F(\mathbf{w}_t) + \frac{\epsilon_t^2 M S^2}{2}(1/\delta + \Gamma)^2. \tag{2.49}$$

We now find a lower bound for the second term in the right hand side of (2.49). Since the Hessian approximation matrices $\hat{\mathbf{B}}_t$ are positive definite their inverses $\hat{\mathbf{B}}_t^{-1}$ are positive semidefinite. In turn, this implies that all the eigenvalues of $\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I}$ are not smaller than Γ since $\Gamma\mathbf{I}$ increases all the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$ by Γ. This lower bound for the eigenvalues of $\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I}$ implies that

$$\nabla F(\mathbf{w}_t)^T \left(\hat{\mathbf{B}}_t^{-1} + \Gamma\mathbf{I}\right) \nabla F(\mathbf{w}_t) \geq \Gamma \|\nabla F(\mathbf{w}_t)\|^2. \tag{2.50}$$

Substituting the lower bound in (2.50) for the corresponding summand in (2.49) and noting the definition of $K := MS^2(1/\delta + \Gamma)^2/2$ in the statement of the lemma, the result in (2.45) follows. ■

Setting aside the term $K\epsilon_t^2$ for the sake of argument (2.44) defines a supermartingale relationship for the sequence of average functions $F(\mathbf{w}_t)$. This implies that the sequence $\epsilon_t \Gamma \ \|\nabla F(\mathbf{w}_t)\|^2$ is almost surely summable which, given that the step sizes ϵ_t are non-summable as per (2.6), further implies that the limit infimum $\liminf_{t\to\infty} \|\nabla F(\mathbf{w}_t)\|$ of the gradient norm $\|\nabla F(\mathbf{w}_t)\|$ is almost surely null. This latter observation is equivalent to having $\liminf_{t\to\infty} \|\mathbf{w}_t - \mathbf{w}^*\|^2 = 0$ with probability 1 over realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$. The term $K\epsilon_t^2$ is a relatively minor nuisance that can be taken care with a technical argument that we present in the proof of the following theorem.

Theorem 1 *Consider the RES algorithm as defined by* (2.30)-(2.33)*. If assumptions 1, 2 and 3 hold true and the sequence of stepsizes satisfies* (2.6)*, the limit of the squared Euclidean distance to optimality* $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ *satisfies*

$$\lim_{t\to\infty} \|\mathbf{w}_t - \mathbf{w}^*\|^2 = 0 \qquad a.s. \tag{2.51}$$

over realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$*.*

Proof: The proof uses the relationship in the statement (2.44) of Lemma 2 to build a supermartingale sequence. For that purpose define the stochastic process γ_t with values

$$\gamma_t := F(\mathbf{w}_t) - F(\mathbf{w}^*) + K \sum_{u=t}^{\infty} \epsilon_u^2. \tag{2.52}$$

Note that γ_t is well defined because the $\sum_{u=t}^{\infty} \epsilon_u^2 < \sum_{u=0}^{\infty} \epsilon_u^2 < \infty$ is summable. Further define the sequence β_t with values

$$\beta_t := \ \epsilon_t \ \Gamma \ \|\nabla F(\mathbf{w}_t)\|^2. \tag{2.53}$$

Let now $\mathcal{F}_t$ be a sigma-algebra measuring γ_t, β_t, and $\mathbf{w}_t$. The conditional expectation of γ_{t+1} given $\mathcal{F}_t$ can be written as

$$\mathbb{E}\left[\gamma_{t+1} \,\middle|\, \mathcal{F}_t\right] = \mathbb{E}\left[F(\mathbf{w}_{t+1}) - F(\mathbf{w}^*) \,\middle|\, \mathcal{F}_t\right] + K \sum_{u=t+1}^{\infty} \epsilon_u^2, \tag{2.54}$$

because the term $K \sum_{u=t}^{\infty} \epsilon_u^2$ is just a deterministic constant. Substituting (2.44) of Lemma 2 into (2.54) and using the definitions of γ_t in (2.52) and β_t in (2.53) yields

$$\mathbb{E}\left[\gamma_{t+1} \,\middle|\, \mathcal{F}_t\right] \ \leq \ \gamma_t - \beta_t \tag{2.55}$$

Since the sequences γ_t and β_t are nonnegative it follows from (2.55) that they satisfy the conditions of the supermartingale convergence theorem – see e.g. theorem E7.4 [114] .

Therefore, we conclude that: (i) The sequence γ_t converges almost surely. (ii) The sum $\sum_{t=0}^{\infty} \beta_t < \infty$ is almost surely finite. Using the explicit form of β_t in (2.53) we have that $\sum_{t=0}^{\infty} \beta_t < \infty$ is equivalent to

$$\sum_{t=0}^{\infty} \epsilon_t \Gamma \|\nabla F(\mathbf{w}_t)\|^2 < \infty, \qquad \text{a.s.} \tag{2.56}$$

Since the sequence of stepsizes is nonsummable for (2.56) to be true we need to have a vanishing subsequence embedded in $\|\nabla F(\mathbf{w}_t)\|^2$. By definition, this miles that the limit infimum of the sequence $\|\nabla F(\mathbf{w}_t)\|^2$ is null,

$$\liminf_{t\to\infty} \|\nabla F(\mathbf{w}_t)\|^2 = 0, \qquad \text{a.s.} \tag{2.57}$$

By using the strong convexity condition we can show that $\|\nabla F(\mathbf{w}_t)\|^2 \geq 2m(F(\mathbf{w}_t) - F(\mathbf{w}^*))$ and therefore,

$$\liminf_{t\to\infty} F(\mathbf{w}_t) - F(\mathbf{w}^*) = 0, \qquad \text{a.s.} \tag{2.58}$$

Further, since the sequence γ_t converges almost surely implies that the limit $\lim_{t\to\infty} F(\mathbf{w}_t) - F(\mathbf{w}^*)$ of the nonnegative objective function errors $F(\mathbf{w}_t) - F(\mathbf{w}^*)$ almost surely exists. This observation in conjunction with the result in (2.58) implies that the whole sequence of $F(\mathbf{w}_t) - F(\mathbf{w}^*)$ converges almost surely to zero,

$$\lim_{t\to\infty} F(\mathbf{w}_t) - F(\mathbf{w}^*) = 0. \qquad \text{a.s.} \tag{2.59}$$

To transform the suboptimality bound in (2.59) into a bound pertaining to the squared distance to optimality $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ simply observe that the lower bound m on the eigenvalues of $\mathbf{H}(\mathbf{w}_t)$ applied to a Taylor's expansion around the optimal argument $\mathbf{w}^*$ implies that

$$F(\mathbf{w}_t) - F(\mathbf{w}^*) \geq \frac{m}{2} \|\mathbf{w}_t - \mathbf{w}^*\|^2. \tag{2.60}$$

Since the limit of $\|F(\mathbf{w}_t) - F(\mathbf{w}^*)\|$ is null the result in (2.51) follows from considering the bound in (2.60). ■

Theorem 1 establishes convergence of the sequence of the RES algorithm summarized in Algorithm 3. In the proof of the prerequisite Lemma 2 the lower bound in the eigenvalues of $\hat{\mathbf{B}}_t$ enforced by the regularization in (2.33) plays a fundamental role. Roughly speaking, the lower bound in the eigenvalues of $\hat{\mathbf{B}}_t$ results in an upper bound on the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$ which limits the effect of random variations on the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. If this regularization is not implemented, i.e., if we keep $\delta = 0$, we may observe catastrophic amplification of random variations of the stochastic gradient. This effect is indeed observed

in the numerical experiments in Section 2.4. The addition of the identity matrix bias $\Gamma\mathbf{I}$ in (2.30) is instrumental in the proof of Theorem 1 proper. This bias limits the effects of randomness in the curvature estimate $\hat{\mathbf{B}}_t$. If random variations in the curvature estimate $\hat{\mathbf{B}}_t$ result in a matrix $\hat{\mathbf{B}}_t^{-1}$ with small eigenvalues the term $\Gamma\mathbf{I}$ dominates and (2.30) reduces to (regular) SGD. This ensures continued progress towards the optimal argument $\mathbf{w}^*$.

2.3.1 Rate of convergence

We complement the convergence result in Theorem 1 with a characterization of the expected convergence rate that we introduce in the following theorem.

Theorem 2 *Consider the RES algorithm as defined by* (2.30)-(2.33) *and let the sequence of step sizes be given by* $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$ *with the parameter* ϵ_0 *sufficiently small and the parameter* T_0 *sufficiently large so as to satisfy the inequality*

$$2\ \epsilon_0 T_0 \Gamma > 1\ . \tag{2.61}$$

If Assumptions 1-3 hold true the difference between the expected objective value $\mathbb{E}\left[F(\mathbf{w}_t)\right]$ *at time* t *and the optimal objective* $F(\mathbf{w}^*)$ *satisfies*

$$\mathbb{E}\left[F(\mathbf{w}_t)\right] - F(\mathbf{w}^*)\ \leq\ \frac{C_0}{T_0 + t}\ , \tag{2.62}$$

where the constant C_0 *satisfies*

$$C_0\ =\ \max\ \left\{\frac{\epsilon_0^2\ T_0^2 K}{2\epsilon_0 T_0 \Gamma - 1}\ ,\ T_0\left(F(\mathbf{w}_0) -\ F(\mathbf{w}^*)\right)\right\}. \tag{2.63}$$

Proof: Theorem 2 claims that the sequence of expected objective values $\mathbb{E}\left[F(\mathbf{w}_t)\right]$ approaches the optimal objective $F(\mathbf{w}^*)$ at a linear rate $O(1/t)$. Before proceeding to the proof of Theorem 2 we introduce a technical lemma that provides a sufficient condition for a sequence u_t to exhibit a linear convergence rate.

Lemma 3 *Let* $c > 1$, $b > 0$ *and* $t_0 > 0$ *be given constants and* $u_t \geq 0$ *be a nonnegative sequence that satisfies the inequality*

$$u_{t+1} \leq \left(1 - \frac{c}{t + t_0}\right) u_t + \frac{b}{(t + t_0)^2}\ , \tag{2.64}$$

for all times $t \geq 0$. *The sequence* u_t *is then bounded as*

$$u_t \leq\ \frac{Q}{t + t_0}, \tag{2.65}$$

for all times $t \geq 0$, where the constant Q is defined as

$$Q := \max\left[\frac{b}{c-1},\ t_0 u_0\right]. \tag{2.66}$$

The proof of Lemma 3 can be found in Appendix A.1. Lemma 3 shows that satisfying (2.64) is sufficient for a sequence to have the linear rate of convergence specified in (2.65). In the following proof of Theorem 2 we show that if the stepsize sequence parameters ϵ_0 and T_0 satisfy (2.61) the sequence $\mathbb{E}\left[F(\mathbf{w}_t)\right] - F(\mathbf{w}^*)$ of expected optimality gaps satisfies (2.64) with $c = 2\epsilon_0 T_0 \Gamma$, $b = \epsilon_0^2 T_0^2 K$ and $t_0 = T_0$. The result in (2.62) then follows as a direct consequence of Lemma 3.

Consider the result in (2.44) of Lemma 2 and subtract the average function optimal value $F(\mathbf{w}^*)$ from both sides of the inequality to conclude that the sequence of optimality gaps in the RES algorithm satisfies

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] - \ F(\mathbf{w}^*) \leq F(\mathbf{w}_t) - \ F(\mathbf{w}^*) - \epsilon_t \Gamma \|\nabla F(\mathbf{w}_t)\|^2 + \epsilon_t^2 K, \tag{2.67}$$

where, we recall, $K := MS^2((1/\delta) + \Gamma)^2/2$ by definition.

We proceed to find a lower bound for the gradient norm $\|\nabla F(\mathbf{w}_t)\|$ in terms of the error of the objective value $F(\mathbf{w}_t) - \ F(\mathbf{w}^*)$ – this is a standard derivation which we include for completeness, see, e.g., [20]. As it follows from Assumption 1 the eigenvalues of the Hessian $\mathbf{H}(\mathbf{w}_t)$ are bounded between $0 < m$ and $M < \infty$ as stated in (2.38). Taking a Taylor's expansion of the objective function $F(\mathbf{y})$ around $\mathbf{w}$ and using the lower bound in the Hessian eigenvalues we can write

$$F(\mathbf{y}) \geq \ F(\mathbf{w}) + \nabla F(\mathbf{w})^T(\mathbf{y} - \mathbf{w}) + \frac{m}{2}\|\mathbf{y} - \mathbf{w}\|^2. \tag{2.68}$$

For fixed $\mathbf{w}$, the right hand side of (2.68) is a quadratic function of $\mathbf{y}$ whose minimum argument we can find by setting its gradient to zero. Doing this yields the minimizing argument $\hat{\mathbf{y}} = \mathbf{w} - (1/m)\nabla F(\mathbf{w})$ implying that for all $\mathbf{y}$ we must have

$$\begin{aligned} F(\mathbf{y}) \geq \ & F(\mathbf{w}) + \nabla F(\mathbf{w})^T(\hat{\mathbf{y}} - \mathbf{w}) + \frac{m}{2}\|\hat{\mathbf{y}} - \mathbf{w}\|^2 \\ = \ & F(\mathbf{w}) - \frac{1}{2m}\|\nabla F(\mathbf{w})\|^2. \end{aligned} \tag{2.69}$$

The bound in (2.69) is true for all $\mathbf{w}$ and $\mathbf{y}$. In particular, for $\mathbf{y} = \mathbf{w}^*$ and $\mathbf{w} = \mathbf{w}_t$ (2.69) yields

$$F(\mathbf{w}^*) \geq F(\mathbf{w}_t) - \frac{1}{2m}\|\nabla F(\mathbf{w}_t)\|^2. \tag{2.70}$$

Rearrange terms in (2.70) to obtain a bound on the gradient norm squared $\|\nabla F(\mathbf{w}_t)\|^2$. Further substitute the result in (2.67) and regroup terms to obtain the bound

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] - \ F(\mathbf{w}^*) \leq \ (1 - 2m\epsilon_t\Gamma)\big(F(\mathbf{w}_t) - \ F(\mathbf{w}^*)\big) + \epsilon_t^2 K. \tag{2.71}$$

Take now expected values on both sides of (2.71). The resulting double expectation in the left hand side simplifies to $\mathbb{E}\left[\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right]\right] = \mathbb{E}\left[F(\mathbf{w}_{t+1})\right]$, which allow us to conclude that (2.71) implies that

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\right] - \ F(\mathbf{w}^*) \leq \ (1 - 2m\epsilon_t\Gamma)\big(\mathbb{E}\left[F(\mathbf{w}_t)\right] - \ F(\mathbf{w}^*)\big) + \epsilon_t^2 K. \tag{2.72}$$

Further substituting $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$, which is the assumed form of the step size sequence by hypothesis, we can rewrite (2.72) as

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\right] - \ F(\mathbf{w}^*) \leq \left(1 - \frac{2\ \epsilon_0 T_0 \Gamma}{(T_0 + t)}\right)\Big(\mathbb{E}\left[F(\mathbf{w}_t)\right] - F(\mathbf{w}^*)\Big) + \left(\frac{\epsilon_0 T_0}{T_0 + t}\right)^2 K. \tag{2.73}$$

Given that the product $2\epsilon_0 T_0 \Gamma > 1$ as per the hypothesis in (2.61) the sequence $\mathbb{E}\left[F(\mathbf{w}_{t+1})\right] - F(\mathbf{w}^*)$ satisfies the hypotheses of Lemma 3 with $c = 2\epsilon_0 T_0 \Gamma$, $b = \epsilon_0^2 T_0^2 K$ and $t_0 = T_0$. It then follows from (2.65) and (2.66) that (2.62) is true for the C_0 constant defined in (2.63) upon identifying u_t with $\mathbb{E}\left[F(\mathbf{x}_{t+1})\right] - \ F(\mathbf{x}^*)$, C_0 with Q, and substituting $c = 2\epsilon_0 T_0 \Gamma$, $b = \epsilon_0^2 T_0^2 K$ and $t_0 = T_0$ for their explicit values. ■

Theorem 2 shows that under specified assumptions, the expected error in terms of the objective value after t RES iterations is at least of order $O(1/t)$. An expected convergence rate of order $O(1/t)$ is typical of stochastic optimization algorithms and, in that sense, no better than conventional SGD. While the convergence rate doesn't change, improvements in convergence time are marked as we illustrate with the numerical experiments of sections 2.4 and 2.5.1.

2.4 Numerical analysis

We compare convergence times of RES and SGD in problems with small and large condition numbers. We use a stochastic quadratic objective function as a test case. In particular, consider a positive definite diagonal matrix $\mathbf{A} \in \mathbb{S}_p^{++}$, a vector $\mathbf{b} \in \mathbb{R}^p$, a random vector $\boldsymbol{\theta} \in \mathbb{R}^p$, and diagonal matrix $\mathrm{diag}(\boldsymbol{\theta})$ defined by $\boldsymbol{\theta}$. The function $F(\mathbf{w})$ in (2.1) is defined as

$$F(\mathbf{w}) := \mathbb{E}_\theta\left[f(\mathbf{w}, \theta)\right] := \mathbb{E}_\theta\left[\frac{1}{2}\mathbf{w}^T\Big(\mathbf{A} + \mathbf{A}\mathrm{diag}(\boldsymbol{\theta})\Big)\mathbf{w} + \mathbf{b}^T\mathbf{w}\right]. \tag{2.74}$$

In (2.74), the random vector $\boldsymbol{\theta}$ is chosen uniformly at random from the p dimensional

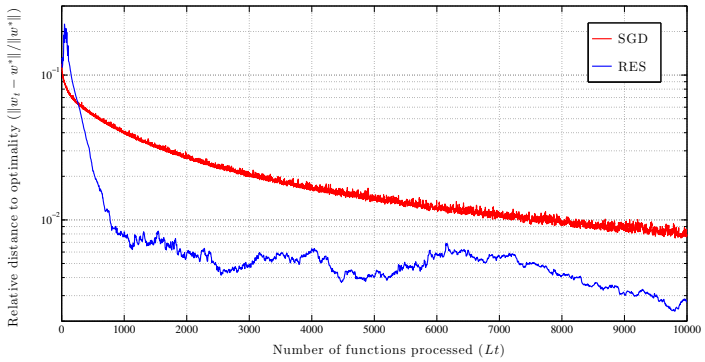


Figure 2.1: Convergence of SGD and RES for the function in (2.74). Relative distance to optimality $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\|$ shown with respect to the number Lt of stochastic functions processed. For RES the number of iterations required to achieve a certain accuracy is smaller than the corresponding number for SGD. See text for parameters' values.

box $\Theta = [-\theta_0, \theta_0]^p$ for some given constant $\theta_0 < 1$. The linear term $\mathbf{b}^T\mathbf{w}$ is added so that the instantaneous functions $f(\mathbf{w}, \theta)$ have different minima which are (almost surely) different from the minimum of the average function $F(\mathbf{w})$. The quadratic term is chosen so that the condition number of $F(\mathbf{w})$ is the condition number of $\mathbf{A}$. Indeed, just observe that since $\mathbb{E}_\theta[\boldsymbol{\theta}] = \mathbf{0}$, the average function in (2.74) can be written as $F(\mathbf{w}) = (1/2)\mathbf{w}^T\mathbf{A}\mathbf{w} + \mathbf{b}^T\mathbf{w}$. The parameter θ_0 controls the variability of the instantaneous functions $f(\mathbf{w}, \theta)$. For small $\theta_0 \approx 0$ instantaneous functions are close to each other and to the average function. For large $\theta_0 \approx 1$ instantaneous functions vary over a large range. We emphasize that the restriction of $F(\mathbf{w})$ to diagonal positive definite quadratic forms is not significant. What is important is the ability to control the condition number of $F(\mathbf{w})$ and the variability of the instantaneous functions $f(\mathbf{w}, \theta)$. Analogous results can be obtained if we pre and post multiply the quadratic form with a random orthogonal matrix.

Further note that we can write the optimum argument as $\mathbf{w}^* = \mathbf{A}^{-1}\mathbf{b}$ for comparison against iterates $\mathbf{w}_t$. This allows us to consider a given ρ and study the convergence metric

$$\tau := L \min_t \left\{ t : \frac{\|\mathbf{w}_t - \mathbf{w}^*\|}{\|\mathbf{w}^*\|} \leq \rho \right\}, \tag{2.75}$$

which represents the time needed to achieve a given relative distance to optimality $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq \rho$ as measured in terms of the number Lt of stochastic functions that are processed to achieve such accuracy.

2.4.1 Effect of problem's condition number

To study the effect of the problem's condition number we generate instances of (2.74) by choosing $\mathbf{b}$ uniformly at random from the box $[0, 1]^p$ and the matrix $\mathbf{A}$ as diagonal with

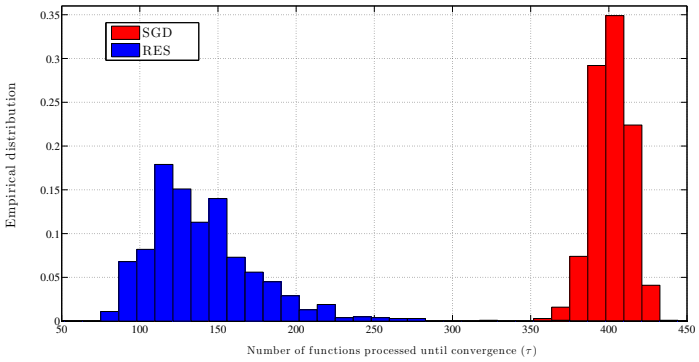


Figure 2.2: Convergence of SGD and RES for well conditioned problems. Empirical distributions of the number $\tau = Lt$ of stochastic functions that are processed to achieve relative precision $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq 10^{-2}$ are shown. Histogram is across $J = 1,000$ realizations of functions as in (2.74) with condition number $10^{\xi} = 10$. See text for parameters' values.

elements a_{ii} uniformly drawn from the discrete set $\{1, 10^{-1}, \ldots, 10^{-\xi}\}$. This choice of $\mathbf{A}$ yields problems with condition number 10^{ξ}.

Representative runs of RES and SGD for $p = 50$, $\theta_0 = 0.5$, and $\xi = 3$ are shown in Fig. 2.1. For the RES run the stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ in (2.4) are computed as an average of $L = 5$ realizations, the regularization parameter in (2.11) is set to $\delta = 10^{-3}$, and the minimum progress parameter in (2.30) to $\Gamma = 10^{-4}$. For SGD we use $L = 1$ in (2.4). In both cases the step size sequence is of the form $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$. Separate rough searches are performed to find step size parameters ϵ_0 and T_0 for RES and SGD that minimize the objective function after 10^4 iterations. For the runs in Fig. 2.1 the best parameters for SGD are $\epsilon_0 = 10^{-1}$ and $T_0 = 10^3$, while for RES the best choices are $\epsilon_0 = 2 \times 10^{-2}$ and $T_0 = 10^3$. Since we are using different values of L for SGD and RES we plot the relative distance to optimality $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\|$ against the number Lt of functions processed up until iteration t.

As expected for a problem with large condition number – since we are using $\xi = 3$, the condition number of $F(\mathbf{w})$ is 10^3 – RES is much faster than SGD. After $t = 10^4$ the distance to optimality for the SGD iterate is $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| = 7.9 \times 10^{-3}$. Comparable accuracy $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| = 7.9 \times 10^{-3}$ for RES is achieved after $t = 179$ iterations. Since we are using $L = 5$ for RES this corresponds to $Lt = 895$ random function evaluations. Conversely, upon processing $Lt = 10^4$ random functions – which corresponds to $t = 2 \times 10^3$ iterations – RES achieves accuracy $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| = 2.7 \times 10^{-3}$. This relative performance difference can be made arbitrarily large by modifying the condition number of $\mathbf{A}$.

A more comprehensive analysis of the relative advantages of RES appears in Figs. 2.2 and 2.3. We keep the same parameters used to generate Fig. 2.1 except that we use $\xi = 1$ for Fig. 2.2 and $\xi = 3$ for Fig. 2.3. This yields a family of well-conditioned functions

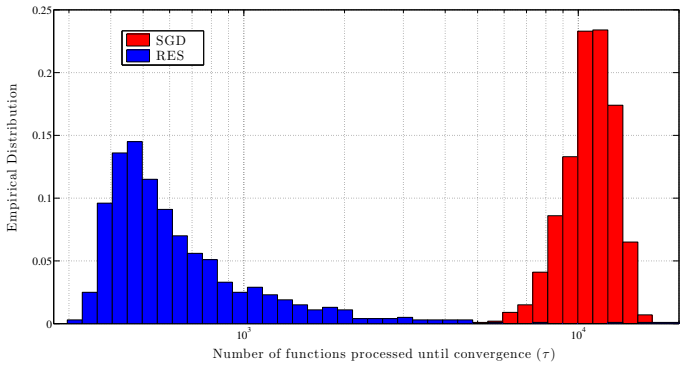


Figure 2.3: Convergence of SGD and RES for ill conditioned problems. Empirical distributions of the number $\tau = Lt$ of stochastic functions that are processed to achieve relative precision $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq 10^{-2}$ are shown. Histogram is across $J = 1,000$ realizations of functions as in (2.74) with condition number $10^{\xi} = 10^3$. See text for parameters' values.

with condition number $10^{\xi} = 10$ and a family of ill-conditioned functions with condition number $10^{\xi} = 10^3$. In Fig. 2.3 we use the same step size parameters of Fig. 2.1 because the function's parameters are the same. In Fig 2.2, where the condition number is smaller, the best step size parameters for SGD are $\epsilon_0 = 6 \times 10^{-1}$ and $T_0 = 10^3$, and for RES the optimal choices are $\epsilon_0 = 10^{-1}$ and $T_0 = 10^3$. In both figures we consider $\rho = 10^{-2}$ and study the convergence times τ and τ' of RES and SGD, respectively [cf. (2.75)]. Resulting empirical distributions of τ and τ' across $J = 1,000$ instances of the functions $F(\mathbf{w})$ in (2.74) are reported in Figs. 2.2 and 2.3 for the well conditioned and ill conditioned families, respectively. For the well conditioned family RES reduces the number of functions processed from an average of $\bar{\tau}' = 401$ in the case of SGD to an average of $\bar{\tau} = 139$. This nondramatic improvement becomes more significant for the ill conditioned family where the reduction is from an average of $\bar{\tau}' = 1.1 \times 10^4$ for SGD to an average of $\bar{\tau} = 7.8 \times 10^2$ for RES.

2.4.2 Central processing unit runtime comparisons

Since the complexity of each RES iteration is larger than the corresponding complexity of SGD we also compare the performances of SGD and RES in terms of the central processing unit (CPU) runtime required to achieve relative accuracy $\rho = 10^{-2}$. The empirical distributions of runtimes across $J = 1,000$ realizations are reported in Figs. 2.4 and 2.5 for the well conditioned and ill conditioned families, respectively. In the well conditioned family, RES reduces runtime from an average of 4.4×10^{-2} seconds in the case of SGD to an average of 3.2×10^{-2} seconds. A more significant improvement can be seen for the ill conditioned family where the reduction is from an average of 5.7×10^{-1} seconds for SGD to an average of 1.5×10^{-1} seconds for RES.

It is important to emphasize that the advantage of RES in terms of CPU runtime

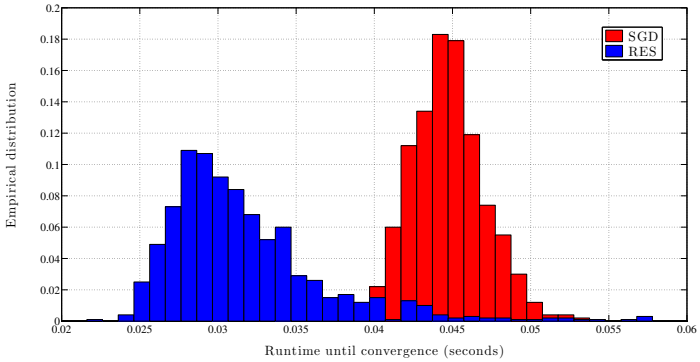


Figure 2.4: CPU runtimes of SGD and RES for well conditioned problems. Empirical distributions of CPU runtimes to achieve relative precision $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq 10^{-2}$ are shown. Histogram is across $J = 1,000$ realizations of functions as in (2.74) with condition number $10^{\xi} = 10^1$.

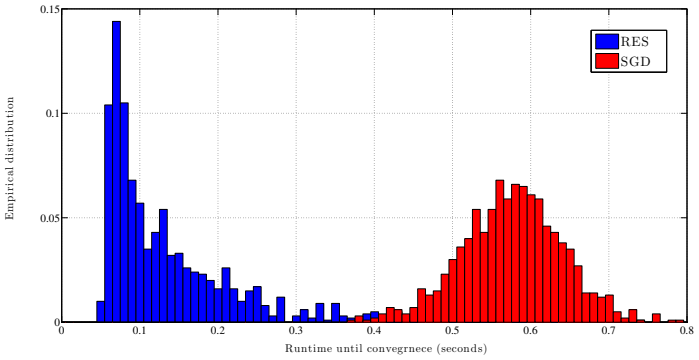


Figure 2.5: CPU runtimes of SGD and RES for ill conditioned problems. Empirical distributions of CPU runtimes to achieve relative precision $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq 10^{-2}$ are shown. Histogram is across $J = 1,000$ realizations of functions as in (2.74) with condition number $10^{\xi} = 10^3$.

depends on specific problem parameters. In particular, if the condition number of $F(\mathbf{w})$ is small, we expect that as we increase the variable dimension p the RES reduction on the number of iterations is overcome by the added computational complexity of each iteration. To illustrate this drawback we repeat the numerical experiments in Figs. 2.2 and 2.4 where the condition number is $10^{\xi} = 10$, but change the number of variables to $p = 5 \times 10^2$. Convergence times needed to achieve relative accuracy $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq 10^{-2}$ as measured by the number of stochastic functions processed and the CPU runtime are shown in Figs. 2.6 and 2.7, respectively. In both cases we show empirical distributions across $J = 1,000$ realizations of the functions $F(\mathbf{w})$. As evidenced by Fig. 2.6, RES is faster in terms of the number of random functions required. But, as evidenced by Fig. 2.7, the opposite is true when we consider CPU runtimes. Indeed, the average number of function evaluations are $\bar{\tau}' = 322$ for RES and $\bar{\tau} = 1.24 \times 10^3$ and SGD but the average runtimes are 1.6 seconds

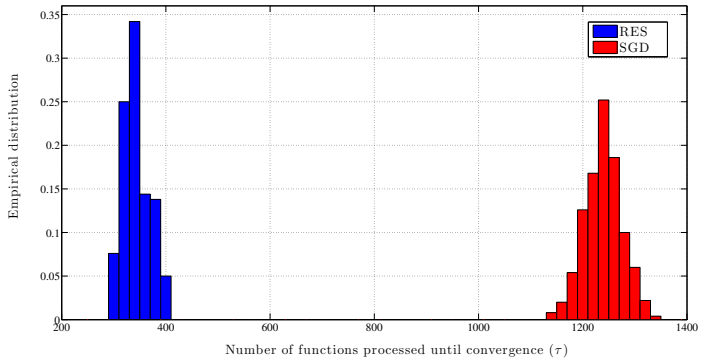


Figure 2.6: Convergence of SGD and RES for a very large dimensional problem with small condition number. Empirical distributions of the number Lt of stochastic functions that are processed to achieve relative precision $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq 10^{-2}$ are shown. Histogram is across $J = 1,000$ realizations of functions as in (2.74) with condition number $10^{\xi} = 10^1$.

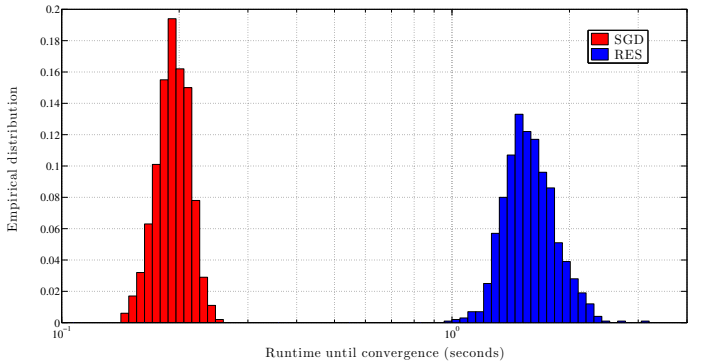


Figure 2.7: Convergence of SGD and RES for a very large dimensional problem with small condition number. Empirical distributions of CPU runtimes to achieve relative precision $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq 10^{-2}$ are shown. Histogram is across $J = 1,000$ realizations of functions as in (2.74) with condition number $10^{\xi} = 10^1$.

for RES and 1.9×10^{-1} seconds for SGD. It follows as a conclusion that SGD outperforms RES for problems that are well conditioned and large dimensional. We summarize this conclusion in the following remark.

Remark 2 In all of our numerical experiments RES reduces the number of stochastic functions that have to be processed to achieve a target accuracy. The reduction is moderate for well conditioned problems but becomes arbitrarily large as the condition number of the objective function increases. However, the computational cost of each RES iteration becomes progressively larger as the dimension of the variable increases. It follows that RES is best suited to problems where the cost of computing stochastic gradients is large, problems where the dimension is not too large, problems where the Hessian approximation

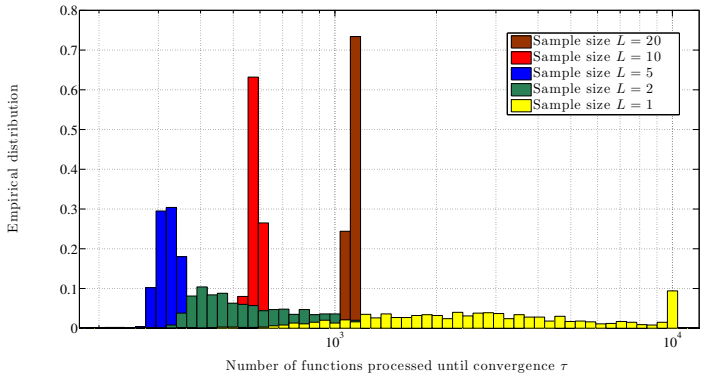


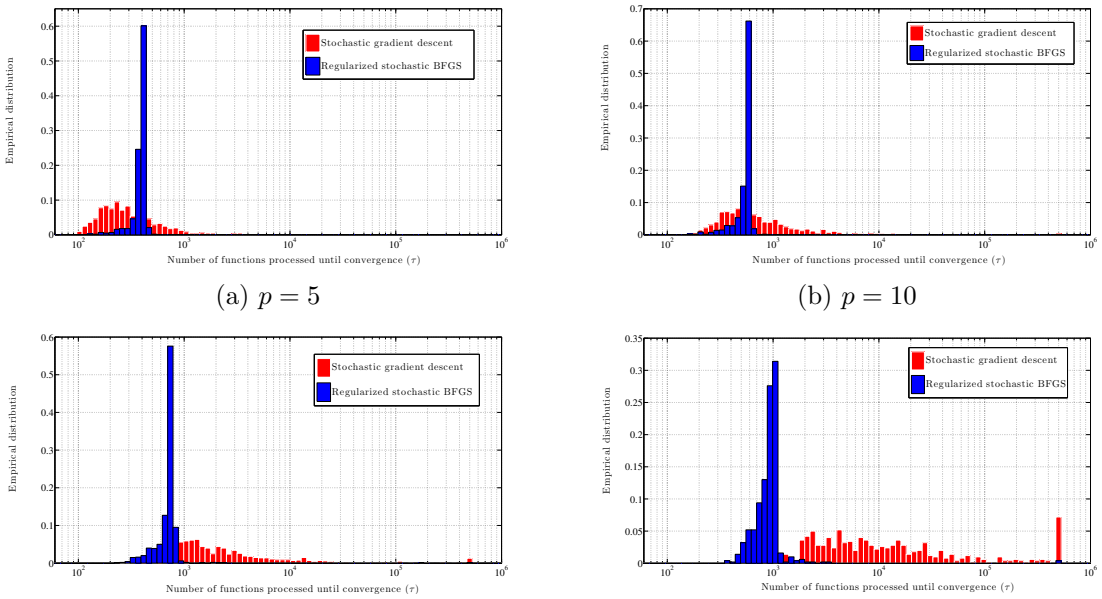
Figure 2.8: Convergence of RES for different sample sizes in the computation of stochastic gradients. Empirical distributions of the number $\tau = Lt$ of processed stochastic functions to achieve relative precision $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq 10^{-2}$ are shown when we use $L = 1$, $L = 2$, $L = 5$, $L = 10$, and $L = 20$ in the evaluation of the stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ in (2.4). The average convergence time decreases as we go from small to moderate values of L and starts increasing as we go from moderate to large values of L. The variance of convergence times decreases monotonically with increasing L.

matrices are sparse, or problems where the condition number makes SGD impracticable. For problems where the cost of computing stochastic gradients is reasonable, have condition numbers close to one, and whose Hessians lack any amenable structure, SGD and variants of SGD are preferable; see also Section 2.5.1.

2.4.3 Choice of stochastic gradient average

The stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ in (2.4) are computed as an average of L sample gradients $\nabla f(\mathbf{w}, \boldsymbol{\theta}_l)$. To study the effect of the choice of L on RES we consider problems as in (2.74) with matrices $\mathbf{A}$ and vectors $\mathbf{b}$ generated as in Section 2.4.1. We consider problems with $p = 50$, $\theta_0 = 0.5$, and $\xi = 2$; set the RES parameters to $\delta = 10^{-3}$ and $\Gamma = 10^{-4}$; and the step size sequence to $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$ with $\epsilon_0 = 10^{-1}$ and $T_0 = 10^3$. We then consider different choices of L and for each specific value generate $J = 1,000$ problem instances. For each run we record the total number τ_L of sample functions that need to be processed to achieve relative distance to optimality $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}^*\| \leq 10^{-2}$ [cf. (2.75)]. If $\tau > 10^4$ we report $\tau = 10^4$ and interpret this outcome as a convergence failure. The resulting estimates of the probability distributions of the times τ_L are reported in Fig. 2.8 for $L = 1$, $L = 2$, $L = 5$, $L = 10$, and $L = 20$.

The trends in convergence times τ apparent in Fig. 2.8 are: (i) As we increase L the variance of convergence times decreases. (ii) The average convergence time decreases as we go from small to moderate values of L and starts increasing as we go from moderate to large values of L. Indeed, the empirical standard deviations of convergence times decrease monotonically from $\sigma_{\tau_1} = 2.8 \times 10^3$ to $\sigma_{\tau_2} = 2.6 \times 10^2$, $\sigma_{\tau_5} = 31.7$, $\sigma_{\tau_{10}} = 28.8$, and



(a) $p = 5$ (b) $p = 10$

(c) $p = 20$ (d) $p = 50$

Figure 2.9: Histogram of the number of data points that SGD and RES needs to converge. Convergence time for RES increases smoothly by increasing the dimension of problem, while convergence time of SGD increases faster.

$\sigma_{\tau_{20}} = 22.7$, when L increases from $L = 1$ to $L = 2$, $L = 5$, $L = 10$, and $L = 20$. The empirical mean decreases from $\bar{\tau}_1 = 3.5 \times 10^3$ to $\bar{\tau}_2 = 6.3 \times 10^2$ as we move from $L = 1$ to $L = 2$, stays at about the same value $\bar{\tau}_5 = 3.3 \times 10^2$ for $L = 5$ and then increases to $\bar{\tau}_{10} = 5.8 \times 10^2$ and $\bar{\tau}_{20} = 1.2 \times 10^3$ for $L = 10$ and $L = 20$. This behavior is expected since increasing L results in curvature estimates $\hat{\mathbf{B}}_t$ closer to the Hessian $\mathbf{H}(\mathbf{w}_t)$ thereby yielding better convergence times. As we keep increasing L, there is no payoff in terms of better curvature estimates and we just pay a penalty in terms of more function evaluations for an equally good $\hat{\mathbf{B}}_t$ matrix. This can be corroborated by observing that the convergence times τ_5 are about half those of τ_{10} which in turn are about half those of τ_{20}. This means that the *actual* convergence times τ/L have similar distributions for $L = 5$, $L = 10$, and $L = 20$. The empirical distributions in Fig. 2.8 show that moderate values of L suffice to provide workable curvature approximations. This justifies the use $L = 5$ in sections 2.4.1 and 2.4.4

2.4.4 Effect of problem's dimension

To evaluate performance for problems of different dimensions we consider functions of the form in (2.74) with $\mathbf{b}$ uniformly chosen from the box $[0, 1]^p$ and diagonal matrix $\mathbf{A}$ as in Section 2.4.1. However, we select the elements a_{ii} as uniformly drawn from the interval $[0, 1]$. This results in problems with more moderate condition numbers and allows for a

comparative study of performance degradations of RES and SGD as the problem dimension p grows.

The variability parameter for the random vector $\boldsymbol{\theta}$ is set to $\theta_0 = 0.5$. The RES parameters are $L = 5$, $\delta = 10^{-3}$, and $\Gamma = 10^{-4}$. For SGD we use $L = 1$. In both methods the step size sequence is $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$ with $\epsilon_0 = 10^{-1}$ and $T_0 = 10^3$. For a problem of dimension p we study convergence times τ_p and τ_p' of RES and SGD as defined in (2.75) with $\rho = 1$. For each value of p considered we determine empirical distributions of τ_p and τ_p' across $J = 1,000$ problem instances. If $\tau > 5 \times 10^5$ we report $\tau = 5 \times 10^5$ and interpret this outcome as a convergence failure. The resulting histograms are shown in Fig. 2.9 for $p = 5$, $p = 10$, $p = 20$, and $p = 50$.

For problems of small dimension having $p = 5$ the average performances of RES and SGD are comparable, with SGD performing slightly better. E.g., the medians of these times are median$(\tau_5) = 400$ and median$(\tau_5') = 265$, respectively. A more significant difference is that times τ_5 of RES are more concentrated than times τ_5' of SGD. The latter exhibits large convergence times $\tau_5' > 10^3$ with probability 0.06 and fails to converge altogether in a few rare instances – we have $\tau_5' = 5 \times 10^5$ in 1 out of 1,000 realizations. In the case of RES all realizations of τ_5 are in the interval $70 \leq \tau_5 \leq 1095$.

As we increase p we see that RES retains the smaller spread advantage while eventually exhibiting better average performance as well. Medians for $p = 10$ are still comparable at median$(\tau_{10}) = 575$ and median$(\tau_{10}') = 582$, as well as for $p = 20$ at median$(\tau_{20}) = 745$ and median$(\tau_{20}') = 1427$. For $p = 50$ the RES median is decidedly better since median$(\tau_{50}) = 950$ and median$(\tau_{50}') = 7942$.

For large dimensional problems having $p = 50$ SGD becomes unworkable. It fails to achieve convergence in 5×10^5 iterations with probability 0.07 and exceeds 10^4 iterations with probability 0.45. For RES we fail to achieve convergence in 5×10^5 iterations with probability 3×10^{-3} and achieve convergence in less than 10^4 iterations in all other cases. Further observe that RES degrades smoothly as p increases. The median number of gradient evaluations needed to achieve convergence increases by a factor of median(τ_{50}')/median$(\tau_5') = 29.9$ as we increase p by a factor of 10. The spread in convergence times remains stable as p grows.

2.5 Support vector machines

A particular case of (2.1) is the implementation of a support vector machine (SVM). Given a training set with points whose class is known the goal of a SVM is to find a hyperplane that best separates the training set. To be specific let $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be a training set containing N pairs of the form $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^p$ is a feature vector and $y_i \in \{-1, 1\}$ is the corresponding vector's class. The goal is to find a hyperplane supported by a vector

$\mathbf{w} \in \mathbb{R}^p$ which separates the training set so that $\mathbf{w}^T\mathbf{x}_i > 0$ for all points with $y_i = 1$ and $\mathbf{w}^T\mathbf{x}_i < 0$ for all points with $y_i = -1$. This vector may not exist if the data is not perfectly separable, or, if the data is separable there may be more than one separating vector. We can deal with both situations with the introduction of a loss function $l((\mathbf{x}, y); \mathbf{w})$ defining some measure of distance between the point $\mathbf{x}_i$ and the hyperplane supported by $\mathbf{w}$. We then select the hyperplane supporting vector as

$$\mathbf{w}^* := \underset{\mathbf{w}}{\operatorname{argmin}} \ \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} l((\mathbf{x}_i, y_i); \mathbf{w}), \tag{2.76}$$

where we also added the regularization term $\lambda\|\mathbf{w}\|^2/2$ for some constant $\lambda > 0$. The vector $\mathbf{w}^*$ in (2.76) balances the minimization of the sum of distances to the separating hyperplane, as measured by the loss function $l((\mathbf{x}, y); \mathbf{w})$, with the minimization of the L_2 norm $\|\mathbf{w}\|_2$ to enforce desirable properties in $\mathbf{w}^*$. Common selections for the loss function are the hinge loss $l((\mathbf{x}, y); \mathbf{w}) = \max(0, 1 - y(\mathbf{w}^T\mathbf{x}))$, the squared hinge loss $l((\mathbf{x}, y); \mathbf{w}) = \max(0, 1 - y(\mathbf{w}^T\mathbf{x}))^2$ and the log loss $l((\mathbf{x}, y); \mathbf{w}) = \log(1 + \exp(-y(\mathbf{w}^T\mathbf{x})))$. See, e.g., [15, 120].

In order to model (2.76) as a stochastic optimization problem in the form of problem (2.1), we define $\boldsymbol{\theta}_i = (\mathbf{x}_i, y_i)$ as a given training point and $m_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ as a uniform probability distribution on the training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N = \{\boldsymbol{\theta}_i\}_{i=1}^N$. Upon defining the sample functions

$$f(\mathbf{w}, \boldsymbol{\theta}) = \ f(\mathbf{w}, (\mathbf{x}, y)) \ := \ \frac{\lambda}{2}\|\mathbf{w}\|^2 + l((\mathbf{x}, y); \mathbf{w}), \tag{2.77}$$

it follows that we can rewrite the objective function in (2.76) as

$$\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} l((\mathbf{x}_i, y_i); \mathbf{w}) \ = \ \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})] \tag{2.78}$$

since each of the functions $f(\mathbf{w}, \boldsymbol{\theta})$ is drawn with probability $1/N$ according to the definition of $m_{\boldsymbol{\theta}}(\boldsymbol{\theta})$. Substituting (2.78) into (2.76) yields a problem with the general form of (2.1) with random functions $f(\mathbf{w}, \boldsymbol{\theta})$ explicitly given by (2.77).

We can then use Algorithm (3) to attempt solution of (2.76). For that purpose we particularize Step 2 to the drawing of L feature vectors $\tilde{\mathbf{x}}_t = [\mathbf{x}_{t1}; \ldots; \mathbf{x}_{tL}]$ and their corresponding class values $\tilde{\mathbf{y}}_t = [y_{t1}; \ldots; y_{tL}]$ to construct the vector of pairs $\tilde{\boldsymbol{\theta}}_t = [(\mathbf{x}_{t1}, y_{t1}); \ldots; (\mathbf{x}_{tL}, y_{tL})]$. These training points are selected uniformly at random from the training set $\mathcal{S}$. We also need to particularize steps 3 and 5 to evaluate the stochastic gradient of the instantaneous function in (2.77). E.g., Step 3 takes the form

$$\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) \ = \ \hat{\mathbf{s}}(\mathbf{w}_t, (\tilde{\mathbf{x}}_t, \tilde{\mathbf{y}}_t)) \ = \ \lambda\mathbf{w}_t + \frac{1}{L}\sum_{i=1}^{L} \nabla_{\mathbf{w}} \ l((\mathbf{x}_{ti}, y_{ti}); \mathbf{w}_t). \tag{2.79}$$

The specific form of Step 5 is obtained by replacing $\mathbf{w}_{t+1}$ for $\mathbf{w}_t$ in (2.79). We analyze the behavior of Algorithm (2.1) in the implementation of a SVM in the following section.

2.5.1 RES vs stochastic gradient descent for suport vector machines

We test Algorithm 1 when using the squared hinge loss $l((\mathbf{x}, y); \mathbf{w}) = \max(0, 1 - y(\mathbf{x}^T\mathbf{w}))^2$ in (2.76). The training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ contains N feature vectors half of which belong to the class $y_i = -1$ with the other half belonging to the class $y_i = 1$. For the class $y_i = -1$ each of the p components of each of the feature vectors $\mathbf{x}_i \in \mathbb{R}^p$ is chosen uniformly at random from the interval $[-0.8, 0.2]$. Likewise, each of the p components of each of the feature vectors $\mathbf{x}_i \in \mathbb{R}^p$ is chosen uniformly at random from the interval $[-0.2, 0.8]$ for the class $y_i = 1$. Observe that the overlap in the range of the feature vectors is such that the classification accuracy expected from a clairvoyant classifier that knows the statistic model of the data set is less than 100%.

In all of our numerical experiments the parameter λ in (2.76) is set to $\lambda = 10^{-4}$. Recall that since the Hessian eigenvalues of $f(\mathbf{w}, \boldsymbol{\theta}) := \lambda\|\mathbf{w}\|^2/2 + l((\mathbf{x}_i, y_i); \mathbf{w})$ are, at least, equal to λ this implies that the eigenvalue lower bound $\tilde{m}$ is such that $\tilde{m} \geq \lambda = 10^{-4}$. We therefore set the RES regularization parameter to $\delta = \lambda = 10^{-4}$. Further set the minimum progress parameter in (2.4) to $\Gamma = 10^{-4}$.

Accelerated versions of SGD can be used for the implementation of SVMs. We provide a comparison of RES with respect to regular SGD and three accelerated versions: Stochastic Average Gradient (SAG) [104], Semi-Stochastic Gradient Descent (S2GD) [47], and Stochastic Approximation by Averaging (SAA) [88]. The SAG algorithm incorporates memory of previous stochastic gradients and uses an average of stochastic gradients as descent direction. The S2GD algorithm is a hybrid method which runs through several epochs. Each epoch is characterized by the computation of a single full gradient and a random number of stochastic gradients, with the number of stochastic gradients selected according to a geometric distribution. In SAA a time average of iterates is computed and reported.

An illustration of the relative performances of SAA, SGD, SAG, S2GD and RES for $p = 40$ and $N = 10^3$ is presented in Fig. 2.10. For RES, we set $L = 5$ and choose the decreasing stepsize sequence $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$ with $\epsilon_0 = 4 \times 10^{-1}$ and $T_0 = 10^6$. These parameters yield best performance after processing 10^4 feature vectors. For SGD, SAA, SAG, and S2GD we tune the various parameters and report results for the combination that yields best performance after processing 10^5 feature vectors. In Fig. 2.10 the value of the objective function $F(\mathbf{w}_t)$ is represented with respect to the number of feature vectors processed, which is given by the product Lt between the iteration index and the sample size used to compute stochastic gradients. To achieve the objective function value $F(\mathbf{w}_t) =$

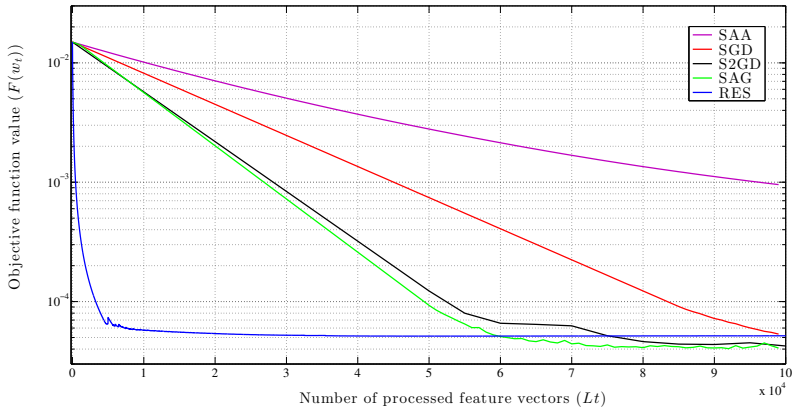


Figure 2.10: Comparison of RES, SGD, the SGD accelerations SAA, SAG, and S2GD to find an optimal linear classifier with respect to the cost in (2.76) for a problem of dimension $p = 40$ and training set with $N = 10^3$ feature vectors. RES processes a much smaller number of feature vectors to achieve comparable objective values. See text for parameters values.

Table 2.1: CPU runtimes of RES, SGD, the SGD accelerations SAA, SAG, and S2GD to find an optimal linear classifier with respect to the cost in (2.76) for different problem dimension p and cardinality of training set N. Times reported are to achieve objective values $F(\mathbf{w}_t) = 10^{-4}$.

p	N	RES	SGD	SAG	S2GD	SAA
40	10^3	45 ms	520 ms	350 ms	280 ms	$>$ 690 ms
400	10^4	0.8 s	$>$ 1.5 s	1.3 s	1.2 s	$>$ 1.7 s

10^{-4}, RES processes $Lt = 3.3 \times 10^3$ training points which is a little more than 3 passes over the complete data set. The required time for processing these number of feature vectors is 45 milliseconds (ms). Reaching the same objective function value $F(\mathbf{w}_t) = 10^{-4}$ requires processing $Lt = 8.3 \times 10^4$ training points for SGD which is more than 83 passes over the whole data set. It takes 520 ms for SGD to achieve this value for the objective function. The number of processed training points to achieve the same objective function value $F(\mathbf{w}_t) = 10^{-4}$ for SAG and S2GD are $Lt = 4.9 \times 10^4$ and $Lt = 5.2 \times 10^4$, respectively. In terms of CPU runtime SAG and S2GD requires 350 ms and 280 ms to achieve objective function value 10^{-4}. The performance of SAA is worse than the performance of regular SGD.

To compare the performances of RES, SGD, SAA, SAG, and S2GD in a larger SVM problem we set the size of the training set to $N = 10^4$ and the dimension of feature vectors to $p = 400$. For RES we make $\epsilon_0 = 1 \times 10^{-1}$, $T_0 = 10^1$ and $L = 20$. For SGD and its accelerations we select the parameters that achieve optimal performance after processing 10^4 feature vectors. The results with respect to number of feature vectors processed are shown in Fig. 2.11 and the CPU times are shown in Table 2.1. The advantage of RES

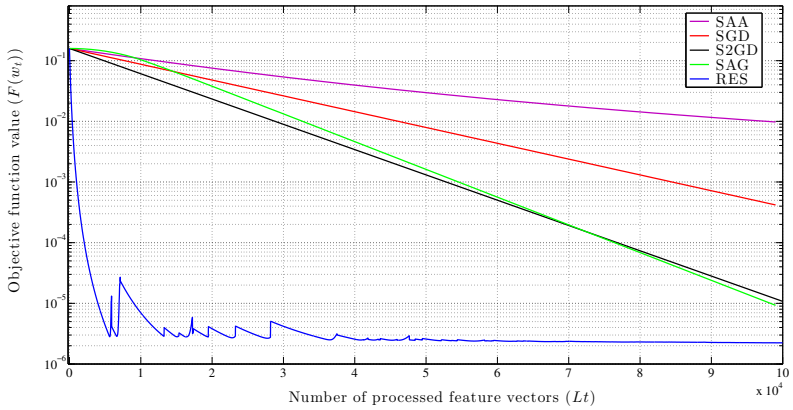


Figure 2.11: Comparison of Fig. 2.10 for problem dimension $p = 400$ and training set cardinality $N = 10^4$. The reduction in the number of feature vectors processed is more pronounced than in Fig. 2.10, but less pronounced in terms of the CPU runtimes shown in Table 2.1. See text for parameters values.

in terms of the number of feature vectors processed is more marked than in the previous experiment. The advantage in terms of CPU processing times is smaller. For reference, the RES achieves the objective value $F(\mathbf{w}_t) = 10^{-4}$ after processing 2.1×10^3 feature vectors in 0.8 seconds. Correspondingly, the numbers of feature vectors processed to attain $F(\mathbf{w}_t) = 10^{-4}$ are 7.6×10^4 and 7.7×10^4 for SAG and S2GD. The CPU runtimes are 1.3 and 1.2, respectively. SGD can not achieve objective function value $F(\mathbf{w}_t) = 10^{-4}$ after processing 10^5 feature vectors in 1.5 seconds. The performance of SAA is still worse than the performance of regular SGD.

2.5.2 RES and stochastic BFGS

We also investigate the difference between regularized and non-regularized versions of stochastic BFGS for feature vectors of dimension $p = 40$. Observe that non-regularized stochastic BFGS corresponds to making $\delta = 0$ and $\Gamma = 0$ in Algorithm 1. To illustrate the advantage of the regularization induced by the proximity requirement in (2.11), as opposed to the non regularized proximity requirement in (2.9), we keep a constant stepsize $\epsilon_t = 10^{-1}$. The corresponding evolutions of the objective function values $F(\mathbf{w}_t)$ with respect to the number of feature vectors processed Lt are shown in Fig. 2.12 along with the values associated with stochastic gradient descent. As we reach convergence the likelihood of having small eigenvalues appearing in $\hat{\mathbf{B}}_t$ becomes significant. In regularized stochastic BFGS (RES) this results in recurrent jumps away from the optimal classifier $\mathbf{w}^*$. However, the regularization term limits the size of the jumps and further permits the algorithm to consistently recover a reasonable curvature estimate. In Fig. 2.12 we process 10^4 feature vectors and observe

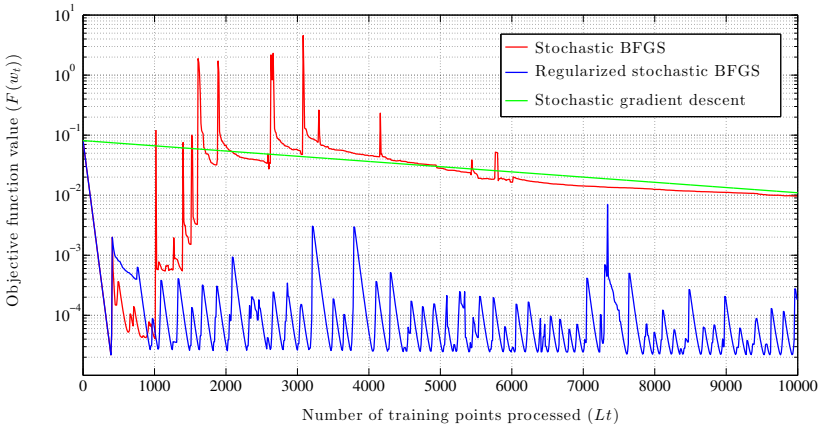


Figure 2.12: Comparison of SGD, regularized stochastic BFGS (RES), and (non regularized) stochastic BFGS. The regularization is fundamental to control the erratic behavior of stochastic BFGS; See text for parameters values.

many occurrences of small eigenvalues. However, the algorithm always recovers and heads back to a good approximation of $\mathbf{w}^*$. In the absence of regularization small eigenvalues in $\hat{\mathbf{B}}_t$ result in larger jumps away from $\mathbf{w}^*$. This not only sets back the algorithm by a much larger amount than in the regularized case but also results in a catastrophic deterioration of the curvature approximation matrix $\hat{\mathbf{B}}_t$. In Fig. 2.12 we observe recovery after the first two occurrences of small eigenvalues but eventually there is a catastrophic deviation after which non-regularized stochastic BFGS behaves not better than SGD.

Chapter 3

Online limited memory BFGS method

3.1 Context and background

Many problems in Machine Learning can be reduced to the minimization of a stochastic objective defined as an expectation over a set of random functions [14,15,69,108]. Specifically, consider an optimization variable $\mathbf{w} \in \mathbb{R}^p$ and a random variable $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$ that determines the choice of a function $f(\mathbf{w}, \boldsymbol{\theta}) : \mathbb{R}^{p \times d} \to \mathbb{R}$. Stochastic optimization problems entail determination of the argument $\mathbf{w}^*$ that minimizes the expected value $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$,

$$\mathbf{w}^* \;:=\; \operatorname*{argmin}_{\mathbf{w}} \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})] \;:=\; \operatorname*{argmin}_{\mathbf{w}} F(\mathbf{w}). \tag{3.1}$$

As we described in Chapter 2, the problem formulation in (3.1) is a general case for the ERM problem in (1.2). To keep the results as general as possible, in this chapter, we focus on the problem in (3.1), but, indeed, the results can be extended to the ERM problem.

In this chapter, we refer to $f(\mathbf{w}, \boldsymbol{\theta})$ as the random or instantaneous functions and to $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ as the average function. A canonical class of problems having this form are support vector machines (SVMs) that reduce binary classification to the determination of a hyperplane that separates points in a given training set; see, e.g., [13, 15, 120]. In that case $\boldsymbol{\theta}$ denotes individual training samples, $f(\mathbf{w}, \boldsymbol{\theta})$ the loss of choosing the hyperplane defined by $\mathbf{w}$, and $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ the mean loss across all elements of the training set. The optimal argument $\mathbf{w}^*$ is the optimal linear classifier.

Numerical evaluation of objective function gradients $\nabla_{\mathbf{w}} F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\nabla_{\mathbf{w}} f(\mathbf{w}, \boldsymbol{\theta})]$ is intractable when the cardinality of Θ is large, as is the case, e.g., when SVMs are trained on large sets. This motivates the use of algorithms relying on stochastic gradients that provide gradient estimates based on small data subsamples. For the purpose of this paper

stochastic optimization algorithms can be divided into three categories: Stochastic gradient descent (SGD) and related first order methods, stochastic Newton methods, and stochastic quasi-Newton methods.

SGD is the most popular method used to solve stochastic optimization problems [15, 49, 107, 130]. However, as we consider problems of ever larger dimension their slow converge times have limited their practical appeal and fostered the search for alternatives. In this regard, it has to be noted that SGD is slow because of both, the use of gradients as descent directions and their replacement by random estimates. Several alternatives have been proposed to deal with randomness in an effort to render the convergence times of SGD closer to the faster convergence times of gradient descent [47, 49, 62, 101, 129]. These SGD variants succeed in reducing randomness and end up exhibiting the asymptotic convergence rate of gradient descent. Although they improve asymptotic convergence rates, the latter methods are still often slow in practice. This is not unexpected. Reducing randomness is of no use when the function $F(\mathbf{w})$ has a challenging curvature profile. In these ill conditioned functions SGD is limited by the already slow convergence times of deterministic gradient descent. The golden standard to deal with ill conditioned functions in deterministic setting is Newton's method. However, unbiased stochastic estimates of Newton steps can't be computed in general. This fact limits the application of stochastic Newton methods to problems with specific structure [10, 128].

If SGD is slow to converge and stochastic Newton can't be used in general, the remaining alternative is to modify deterministic quasi-Newton methods that speed up convergence times relative to gradient descent without using Hessian evaluations [24, 32, 87, 89]. This has resulted in the development of the stochastic quasi-Newton methods known as online (o)Broyden-Fletcher-Goldfarb-Shanno (BFGS) [12, 105], regularized stochastic BFGS (RES) [70], and online limited memory (oL)BFGS [105] which occupy the middle ground of broad applicability irrespective of problem structure and conditioning. All of these three algorithms extend BFGS by using stochastic gradients as both, descent directions and constituents of Hessian estimates. The oBFGS algorithm is a direct generalization of BFGS that uses stochastic gradients in lieu of deterministic gradients. RES, presented in Chapter 2, differs in that it further modifies BFGS to yield an algorithm that retains its convergence advantages while improving theoretical convergence guarantees and numerical behavior. The oLBFGS method uses a modification of BFGS to reduce the computational cost of each iteration.

An important observation here is that in trying to adapt to the changing curvature of the objective, stochastic quasi-Newton methods may end up exacerbating the problem. Indeed, since Hessian estimates are stochastic, it is possible to end up with almost singular Hessian estimates. The corresponding small eigenvalues then result in a catastrophic amplification

of the noise which nullifies progress made towards convergence. This is not a minor problem. In oBFGS this possibility precludes convergence analyses [12,105] and may result in erratic numerical behavior; see e.g., Figure 2.12. As a matter of fact, the main motivation for the introduction of RES is to avoid this catastrophic noise amplification so as to retain smaller convergence times while ensuring that optimal arguments are found with probability 1 [70]. However valuable, the convergence guarantees of RES and the convergence time advantages of oBFGS and RES are tainted by an iteration cost of order $O(p^2)$ and $O(p^3)$, respectively, which precludes their use in problems where p is very large. In deterministic settings this problem is addressed by limited memory (L)BFGS [55] which can be easily generalized to develop the oLBFGS algorithm [105]. Numerical tests of oLBGS are promising but theoretical convergence characterizations are still lacking. The main contribution of this paper is to show that oLBFGS converges with probability 1 to optimal arguments across realizations of the random variables $\boldsymbol{\theta}$. This is the same convergence guarantee provided for RES and is in marked contrast with oBFGS that fails to converge if not properly regularized. Convergence guarantees for oLBFGS do not require such measures.

We begin this chapter with brief discussions of deterministic BFGS (Section 3.2 and LBFGS (Section 3.2.1) and the introduction of oLBFGS (Section 3.2.2). The fundamental idea in BFGS and oLBFGS is to continuously satisfy a secant condition while staying close to previous curvature estimates. They differ in that BFGS uses all past gradients to estimate curvature while oLBFGS uses a fixed moving window of past gradients. The use of this window reduces memory and computational cost. The difference between LBGS and oLBFGS is the use of stochastic gradients in lieu of their deterministic counterparts.

Convergence properties of oLBFGS are then analyzed (Section 3.3). Under the assumption that the sample functions $f(\mathbf{w}, \boldsymbol{\theta})$ are strongly convex and Lipschitz continuous we show that the trace and determinant of the Hessian approximations computed by oLBFGS are upper and lower bounded, respectively (Lemma 5). These bounds are then used to limit the range of variation of the ratio between the Hessian approximations' largest and smallest eigenvalues (Lemma 6). In turn, this condition number limit is shown to be sufficient to prove convergence to the optimal argument $\mathbf{w}^*$ with probability 1 over realizations of the sample functions (Theorem 3). This is an important result because it ensures that oLBFGS doesn't suffer from the numerical problems that hinder oBFGS. We complement this almost sure convergence result with a characterization of the convergence rate which is shown to be at least linear in expectation (Theorem 4). It is fair to emphasize that, different from the deterministic case, the convergence rate of oLBFGS is not better than the convergence rate of SGD. This is not a limitation of our analysis. The difference between stochastic and regular gradients introduces a noise term that dominates convergence once we are close to the optimum, which is where superlinear convergence rates manifest. In fact, the same

convergence rate would be observed if exact Hessians were available. The best that can be proven of oLBFGS is that the convergence rate is not worse than that of SGD. Given that theoretical guarantees only state that the curvature correction does not exacerbate the problem's condition it is perhaps fairer to describe oLBFGS as an adaptive reconditioning strategy instead of a stochastic quasi-Newton method. The latter description refers to the genesis of the algorithm. The former is a more accurate description of its actual behavior.

To show the advantage of using oLBFGS as an adaptive reconditioning strategy we develop its application to the training of SVMs (Section 3.4) and perform a comparative numerical analysis with synthetic data. The conclusions of this numerical analysis are that oLBFGS performs as well as oBFGS and RES while outperforming SGD when convergence is measured with respect to the number of feature vectors processed. In terms of computation time, oLBFGS outperforms all three methods, SGD, oBFGS, and RES. The advantages of oLBFGS grow with the dimension of the feature vector and can be made arbitrarily large (Section 3.4.1). To further substantiate numerical claims we use oLBFGS to train a logistic regressor to predict the click through rate in a search engine advertising problem (Section 3.5). The logistic regression uses a heterogeneous feature vector with 174,026 binary entries that describe the user, the search, and the advertisement (Section 3.5.1). Being a large scale problem with heterogeneous data, the condition number of the logistic log likelihood objective is large and we expect to see significant advantages of oLBFGS relative to SGD. This expectation is fulfilled. The oLBFGS algorithm trains the regressor using less than 1% of the data required by SGD to obtain similar classification accuracy (Section 3.5.3).

Notation Lowercase boldface $\mathbf{v}$ denotes a vector and uppercase boldface $\mathbf{A}$ a matrix. We use $\|\mathbf{v}\|$ to denote the Euclidean norm of vector $\mathbf{v}$ and $\|\mathbf{A}\|$ to denote the Euclidean norm of matrix $\mathbf{A}$. The trace of $\mathbf{A}$ is written as $\text{tr}(\mathbf{A})$ and the determinant as $\det(\mathbf{A})$. We use $\mathbf{I}$ for the identity matrix of appropriate dimension. The notation $\mathbf{A} \succeq \mathbf{B}$ implies that the matrix $\mathbf{A} - \mathbf{B}$ is positive semidefinite. The operator $\mathbb{E}_{\mathbf{x}}[\cdot]$ stands in for expectation over random variable $\mathbf{x}$ and $\mathbb{E}[\cdot]$ for expectation with respect to the distribution of a stochastic process.

3.2 Algorithm definition

Recall the definitions of the sample functions $f(\mathbf{w}, \boldsymbol{\theta})$ and the average function $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$. We assume the sample functions $f(\mathbf{w}, \boldsymbol{\theta})$ are strongly convex for all $\boldsymbol{\theta}$. This implies the objective function $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$, being an average of the strongly convex sample functions, is also strongly convex. We define the gradient $\mathbf{s}(\mathbf{w}) := \nabla F(\mathbf{w})$ of the

average function $F(\mathbf{w})$ and assume that it can be computed as

$$\mathbf{s}(\mathbf{w}) := \nabla F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\nabla f(\mathbf{w}, \boldsymbol{\theta})]. \tag{3.2}$$

Since the function $F(\mathbf{w})$ is strongly convex, gradients $\mathbf{s}(\mathbf{w})$ are descent directions that can be used to find the optimal argument $\mathbf{w}^*$ in (3.1). Introduce then a time index t, a step size ϵ_t, and a positive definite matrix $\mathbf{B}_t^{-1} \succ 0$ to define a generic descent algorithm through the iteration

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \epsilon_t \, \mathbf{B}_t^{-1} \mathbf{s}(\mathbf{w}_t) \;=\; \mathbf{w}_t - \epsilon_t \, \mathbf{d}_t. \tag{3.3}$$

where we have also defined the descent step $\mathbf{d}_t = \mathbf{B}_t^{-1}\mathbf{s}(\mathbf{w}_t)$. When $\mathbf{B}_t^{-1} = \mathbf{I}$ is the identity matrix, (3.3) reduces to gradient descent. When $\mathbf{B}_t = \mathbf{H}(\mathbf{w}_t) := \nabla^2 F(\mathbf{w}_t)$ is the Hessian of the objective function, (3.3) defines Newton's algorithm. In this paper we focus on quasi-Newton methods whereby we attempt to select matrices $\mathbf{B}_t$ close to the Hessian $\mathbf{H}(\mathbf{w}_t)$. Various methods are known to select matrices $\mathbf{B}_t$, including those by Broyden e.g., [21]; Davidon, Fletcher, and Powell (DFP) e.g., [35]; and Broyden, Fletcher, Goldfarb, and Shanno (BFGS) e.g., [24, 89]. We work with the matrices $\mathbf{B}_t$ used in BFGS since they have been observed to work best in practice (see [24]).

In BFGS, the function's curvature $\mathbf{B}_t$ is approximated by a finite difference. Let $\mathbf{v}_t$ denote the variable variation at time t and $\mathbf{r}_t$ the gradient variation at time t which are respectively defined as

$$\mathbf{v}_t := \mathbf{w}_{t+1} - \mathbf{w}_t, \qquad \mathbf{r}_t := \mathbf{s}(\mathbf{w}_{t+1}) - \mathbf{s}(\mathbf{w}_t). \tag{3.4}$$

We select the matrix $\mathbf{B}_{t+1}$ to be used in the next time step so that it satisfies the secant condition $\mathbf{B}_{t+1}\mathbf{v}_t = \mathbf{r}_t$. The rationale for this selection is that the Hessian $\mathbf{H}(\mathbf{w}_t)$ satisfies this condition for $\mathbf{w}_{t+1}$ tending to $\mathbf{w}_t$. Notice however that the secant condition $\mathbf{B}_{t+1}\mathbf{v}_t = \mathbf{r}_t$ is not enough to completely specify $\mathbf{B}_{t+1}$. To resolve this indeterminacy, matrices $\mathbf{B}_{t+1}$ in BFGS are also required to be as close as possible to the previous Hessian approximation $\mathbf{B}_t$ in terms of a weighted Frobenius norm (see [87]). These conditions can be resolved in closed form leading to the explicit expression,

$$\mathbf{B}_{t+1} = \mathbf{B}_t + \frac{\mathbf{r}_t \mathbf{r}_t^T}{\mathbf{v}_t^T \mathbf{r}_t} - \frac{\mathbf{B}_t \mathbf{v}_t \mathbf{v}_t^T \mathbf{B}_t}{\mathbf{v}_t^T \mathbf{B}_t \mathbf{v}_t}. \tag{3.5}$$

While the expression in (3.5) permits updating the Hessian approximations $\mathbf{B}_{t+1}$, implementation of the descent step in (3.3) requires its inversion. This can be avoided by using the Sherman-Morrison formula in (3.5) to write

$$\mathbf{B}_{t+1}^{-1} \;=\; \mathbf{Z}_t^T \, \mathbf{B}_t^{-1} \, \mathbf{Z}_t \;+\; \rho_t \, \mathbf{v}_t \mathbf{v}_t^T, \tag{3.6}$$

where we defined the scalar ρ_t and the matrix $\mathbf{Z}_t$ as

$$\rho_t := \frac{1}{\mathbf{v}_t^T \mathbf{r}_t}, \qquad \mathbf{Z}_t := \mathbf{I} - \rho_t \mathbf{r}_t \mathbf{v}_t^T. \tag{3.7}$$

The updates in (3.5) and (3.6) require the inner product of the gradient and variable variations to be positive, i.e., $\mathbf{v}_t^T \mathbf{r}_t > 0$. This is always true if the objective $F(\mathbf{w})$ is strongly convex and further implies that $\mathbf{B}_{t+1}^{-1}$ stays positive definite if $\mathbf{B}_t^{-1} \succ \mathbf{0}$, ([87]).

Each BFGS iteration has a cost of $O(p^2)$ arithmetic operations. This is less than the $O(p^3)$ of each step in Newton's method but more than the $O(p)$ cost of each gradient descent iteration. In general, the relative convergence rates are such that the total computational cost of BFGS to achieve a target accuracy is smaller than the corresponding cost of gradient descent. Still, alternatives to reduce the computational cost of each iteration are of interest for large scale problems. Likewise, BFGS requires storage and propagation of the $O(p^2)$ elements of $\mathbf{B}_t^{-1}$, whereas gradient descent requires storage of $O(p)$ gradient elements only. This motivates alternatives that have smaller memory footprints. Both of these objectives are accomplished by the limited memory (L)BFGS algorithm that we describe in the following section.

3.2.1 LBFGS: Limited memory BFGS

As it follows from (3.6), the updated Hessian inverse approximation $\mathbf{B}_t^{-1}$ depends on $\mathbf{B}_{t-1}^{-1}$ and the curvature information pairs $\{\mathbf{v}_{t-1}, \mathbf{r}_{t-1}\}$. In turn, to compute $\mathbf{B}_{t-1}^{-1}$, the estimate $\mathbf{B}_{t-2}^{-1}$ and the curvature pair $\{\mathbf{v}_{t-2}, \mathbf{r}_{t-2}\}$ are used. Proceeding recursively, it follows that $\mathbf{B}_t^{-1}$ is a function of the initial approximation $\mathbf{B}_0^{-1}$ and all previous t curvature information pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=0}^{t-1}$. The idea in LBFGS is to restrict the use of past curvature information to the last τ pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=t-\tau}^{t-1}$. Since earlier iterates $\{\mathbf{v}_u, \mathbf{r}_u\}$ with $u < t - \tau$ are likely to carry little information about the curvature at the current iterate $\mathbf{w}_t$, this restriction is expected to result in a minimal performance penalty.

For a precise definition, pick a positive definite matrix $\mathbf{B}_{t,0}^{-1}$ as the initial Hessian inverse approximation at step t. Proceed then to perform τ updates of the form in (3.6) using the last τ curvature information pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=t-\tau}^{t-1}$. Denoting as $\mathbf{B}_{t,u}^{-1}$ the curvature approximation after u updates are performed we have that the refined matrix approximation $\mathbf{B}_{t,u+1}^{-1}$ is given by [cf. (3.6)]

$$\mathbf{B}_{t,u+1}^{-1} = \mathbf{Z}_{t-\tau+u}^T \, \mathbf{B}_{t,u}^{-1} \, \mathbf{Z}_{t-\tau+u} + \rho_{t-\tau+u} \, \mathbf{v}_{t-\tau+u} \, \mathbf{v}_{t-\tau+u}^T, \tag{3.8}$$

where $u = 0, \ldots, \tau - 1$ and the constants $\rho_{t-\tau+u}$ and rank-one plus identity matrices $\mathbf{Z}_{t-\tau+u}$ are as given in (3.7). The inverse Hessian approximation $\mathbf{B}_t^{-1}$ to be used in (3.3) is the one yielded after completing the τ updates in (3.13), i.e., $\mathbf{B}_t^{-1} = \mathbf{B}_{t,\tau}^{-1}$. Observe that when $t < \tau$

there are not enough pairs $\{\mathbf{v}_u, \mathbf{r}_u\}$ to perform τ updates. In such case we just redefine $\tau = t$ and proceed to use the $t = \tau$ available pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=0}^{t-1}$.

Implementation of the product $\mathbf{B}_t^{-1}\mathbf{s}(\mathbf{w}_t)$ in (3.3) for matrices $\mathbf{B}_t^{-1} = \mathbf{B}_{t,\tau}^{-1}$ obtained from the recursion in (3.13) does not need explicit computation of the matrix $\mathbf{B}_{t,\tau}^{-1}$. Although the details are not straightforward, observe that each iteration in (3.13) is similar to a rank-one update and that as such it is not unreasonable to expect that the product $\mathbf{B}_t^{-1}\mathbf{s}(\mathbf{w}_t) = \mathbf{B}_{t,\tau}^{-1}\mathbf{s}(\mathbf{w}_t)$ can be computed using τ recursive inner products. Assuming that this is possible, the implementation of the recursion in (3.13) doesn't need computation and storage of prior matrices $\mathbf{B}_{t-1}^{-1}$. Rather, it suffices to keep the τ most recent curvature information pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=t-\tau}^{t-1}$, thus reducing storage requirements from $O(p^2)$ to $O(\tau p)$. Furthermore, each of these inner products can be computed at a cost of p operations yielding a total computational cost of $O(\tau p)$ per LBFGS iteration. Hence, LBFGS decreases both the memory requirements and the computational cost of each iteration from the $O(p^2)$ required by regular BFGS to $O(\tau p)$. We present the details of this iteration in the context of the online (stochastic) LBFGS that we introduce in the following section.

3.2.2 Online (Stochastic) limited memory BFGS

To implement (3.3) and (3.13) we need to compute gradients $\mathbf{s}(\mathbf{w}_t)$. This is impractical when the number of functions $f(\mathbf{w}, \boldsymbol{\theta})$ is large, as is the case in most stochastic problems of practical interest and motivates the use of stochastic gradients in lieu of actual gradients. Consider a given set of L realizations $\tilde{\boldsymbol{\theta}} = [\boldsymbol{\theta}_1; ...; \boldsymbol{\theta}_L]$ and define the stochastic gradient of $F(\mathbf{w})$ at $\mathbf{w}$ given samples $\tilde{\boldsymbol{\theta}}$ as

$$\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) := \frac{1}{L}\sum_{l=1}^{L} \nabla f(\mathbf{w}, \boldsymbol{\theta}_l). \tag{3.9}$$

In oLBFGS we use stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ for descent directions and curvature estimators. In particular, the descent iteration in (3.3) is replaced by the descent iteration

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \epsilon_t\, \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) \;=\; \mathbf{w}_t - \epsilon_t \hat{\mathbf{d}}_t, \tag{3.10}$$

where $\tilde{\boldsymbol{\theta}}_t = [\boldsymbol{\theta}_{t1}; ...; \boldsymbol{\theta}_{tL}]$ is the set of samples used at step t to compute the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ as per (3.9) and the matrix $\hat{\mathbf{B}}_t^{-1}$ is a function of past stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}_u, \tilde{\boldsymbol{\theta}}_u)$ with $u \leq t$ instead of a function of past gradients $\mathbf{s}(\mathbf{w}_u)$ as in (3.3). As we also did in (3.3) we have defined the stochastic step $\hat{\mathbf{d}}_t := \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ to simplify upcoming discussions.

To properly specify $\hat{\mathbf{B}}_t^{-1}$ we define the stochastic gradient variation $\hat{\mathbf{r}}_t$ at time t as the difference between the stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ and $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ associated with

subsequent iterates $\mathbf{w}_{t+1}$ and $\mathbf{w}_t$ and the *common* set of samples $\tilde{\boldsymbol{\theta}}_t$ [cf. (3.4)],

$$\hat{\mathbf{r}}_t := \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t). \tag{3.11}$$

Observe that $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is the stochastic gradient used at time t in (3.10) but that $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ is computed solely for the purpose of determining the stochastic gradient variation. The perhaps more natural definition $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_{t+1}) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ for the stochastic gradient variation, which relies on the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_{t+1})$ used at time $t+1$ in (3.10) is not sufficient to guarantee convergence; see e.g., [70].

To define the oLBFGS algorithm we just need to provide stochastic versions of the definitions in (3.7) and (3.13). The scalar constants and identity plus rank-one matrices in (3.7) are redefined to the corresponding stochastic quantities

$$\hat{\rho}_{t-\tau+u} \;=\; \frac{1}{\mathbf{v}_{t-\tau+u}^T \hat{\mathbf{r}}_{t-\tau+u}} \quad \text{and} \quad \hat{\mathbf{Z}}_{t-\tau+u} \;=\; \mathbf{I} - \hat{\rho}_{t-\tau+u} \hat{\mathbf{r}}_{t-\tau+u} \mathbf{v}_{t-\tau+u}^T, \tag{3.12}$$

whereas the LBFGS matrix $\mathbf{B}_t^{-1} = \mathbf{B}_{t,\tau}^{-1}$ in (3.13) is replaced by the oLBFGS Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ which we define as the outcome of τ recursive applications of the update,

$$\hat{\mathbf{B}}_{t,u+1}^{-1} \;=\; \hat{\mathbf{Z}}_{t-\tau+u}^T \, \hat{\mathbf{B}}_{t,u}^{-1} \, \hat{\mathbf{Z}}_{t-\tau+u} + \hat{\rho}_{t-\tau+u} \, \mathbf{v}_{t-\tau+u} \, \mathbf{v}_{t-\tau+u}^T, \tag{3.13}$$

where the initial matrix $\hat{\mathbf{B}}_{t,0}^{-1}$ is given and the time index is $u = 0, \ldots, \tau - 1$. The oLBFGS algorithm is defined by the stochastic descent iteration in (3.10) with matrices $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ computed by τ recursive applications of (3.13). Except for the fact that they use stochastic variables, (3.10) and (3.13) are identical to (3.3) and (3.13). Thus, as is the case in (3.3), the Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1}$ in (3.13) is a function of the initial Hessian inverse approximation $\mathbf{B}_{t,0}^{-1}$ and the τ most recent curvature information pairs $\{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=t-\tau}^{t-1}$. Likewise, when $t < \tau$ there are not enough pairs $\{\mathbf{v}_u, \hat{\mathbf{r}}_u\}$ to perform τ updates. In such case we just redefine $\tau = t$ and proceed to use the $t = \tau$ available pairs $\{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=0}^{t-1}$. We also point out that the update in (3.13) necessitates $\hat{\mathbf{r}}_u^T \mathbf{v}_u > 0$ for all time indexes u. This is true as long as the instantaneous functions $f(\mathbf{w}, \boldsymbol{\theta})$ are strongly convex with respect to $\mathbf{w}$ as we show in Lemma 4.

The equations in (3.10) and (3.13) are used conceptually but not in practical implementations. For the latter we exploit the structure of (3.13) to rearrange the terms in the computation of the product $\hat{\mathbf{B}}_t^{-1} \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. To see how this is done consider the recursive update for the Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1}$ in (3.13) and make $u = \tau - 1$ to write

$$\hat{\mathbf{B}}_t^{-1} \;=\; \hat{\mathbf{B}}_{t,\tau}^{-1} \;=\; \left(\hat{\mathbf{Z}}_{t-1}^T\right) \hat{\mathbf{B}}_{t,\tau-1}^{-1} \left(\hat{\mathbf{Z}}_{t-1}\right) + \hat{\rho}_{t-1} \, \mathbf{v}_{t-1} \, \mathbf{v}_{t-1}^T. \tag{3.14}$$

Equation (3.14) shows the relation between the Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1}$ and the $(\tau-1)$st updated version of the initial Hessian inverse approximation $\hat{\mathbf{B}}_{t,\tau-1}^{-1}$ at step t. Set now $u=\tau-2$ in (3.13) to express $\hat{\mathbf{B}}_{t,\tau-1}^{-1}$ in terms of $\hat{\mathbf{B}}_{t,\tau-2}^{-1}$ and substitute the result in (3.14) to rewrite $\hat{\mathbf{B}}_t^{-1}$ as

$$\hat{\mathbf{B}}_t^{-1} = \left(\hat{\mathbf{Z}}_{t-1}^T\hat{\mathbf{Z}}_{t-2}^T\right)\hat{\mathbf{B}}_{t,\tau-2}^{-1}\left(\hat{\mathbf{Z}}_{t-2}\hat{\mathbf{Z}}_{t-1}\right) + \hat{\rho}_{t-2}\left(\hat{\mathbf{Z}}_{t-1}^T\right)\mathbf{v}_{t-2}\,\mathbf{v}_{t-2}^T\left(\hat{\mathbf{Z}}_{t-1}\right) + \hat{\rho}_{t-1}\,\mathbf{v}_{t-1}\,\mathbf{v}_{t-1}^T. \tag{3.15}$$

We can proceed recursively by substituting $\hat{\mathbf{B}}_{t,\tau-2}^{-1}$ for its expression in terms of $\hat{\mathbf{B}}_{t,\tau-3}^{-1}$ and in the result substitute $\hat{\mathbf{B}}_{t,\tau-3}^{-1}$ for its expression in terms of $\hat{\mathbf{B}}_{t,\tau-3}^{-1}$ and so on. Observe that a new summand is added in each of these substitutions from which it follows that repeating this process τ times yields

$$\begin{aligned}\hat{\mathbf{B}}_t^{-1} = {} & \left(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau}^T\right)\hat{\mathbf{B}}_{t,0}^{-1}\left(\hat{\mathbf{Z}}_{t-\tau}\ldots\hat{\mathbf{Z}}_{t-1}\right)\\ & + \hat{\rho}_{t-\tau}\left(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau+1}^T\right)\mathbf{v}_{t-\tau}\mathbf{v}_{t-\tau}^T\left(\hat{\mathbf{Z}}_{t-\tau+1}\ldots\hat{\mathbf{Z}}_{t-1}\right)\\ & + \cdots + \hat{\rho}_{t-2}\left(\hat{\mathbf{Z}}_{t-1}^T\right)\mathbf{v}_{t-2}\mathbf{v}_{t-2}^T\left(\hat{\mathbf{Z}}_{t-1}\right) + \hat{\rho}_{t-1}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^T.\end{aligned} \tag{3.16}$$

The important observation in (3.16) is that the matrix $\hat{\mathbf{Z}}_{t-1}$ and its transpose $\hat{\mathbf{Z}}_{t-1}^T$ are the first and last product terms of all summands except the last, that the matrices $\hat{\mathbf{Z}}_{t-2}$ and its transpose $\hat{\mathbf{Z}}_{t-2}^T$ are second and penultimate in all terms but the last two, and so on. Thus, when computing the oLBFGS step $\hat{\mathbf{d}}_t := \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)$ the operations needed to compute the product with the next to last summand of (3.16) can be reused to compute the product with the second to last summand which in turn can be reused in determining the product with the third to last summand and so on. This observation compounded with the fact that multiplications with the identity plus rank one matrices $\hat{\mathbf{Z}}_{t-1}$ requires $O(p)$ operations yields an algorithm that can compute the oLBFGS step $\hat{\mathbf{d}}_t := \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)$ in $O(\tau p)$ operations.

We summarize the specifics of such computation in the following proposition where we consider the computation of the product $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ with a given arbitrary vector $\mathbf{p}$.

Proposition 2 *Consider the oLBFGS Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ obtained after τ recursive applications of the update in* (3.13) *with the scalar sequence $\hat{\rho}_{t-\tau+u}$ and identity plus rank-one matrix sequence $\hat{\mathbf{Z}}_{t-\tau+u}$ as defined in* (3.12) *for given variable and stochastic gradient variation pairs $\{\mathbf{v}_u,\mathbf{r}_u\}_{u=t-\tau}^{t-1}$. For a given vector $\mathbf{p}=\mathbf{p}_0$ define the sequence of vectors $\mathbf{p}_k$ through the recursion*

$$\mathbf{p}_{u+1} \;=\; \mathbf{p}_u - \alpha_u\hat{\mathbf{r}}_{t-u-1} \qquad \textit{for } u=0,\ldots,\tau-1, \tag{3.17}$$

where we also define the constants $\alpha_u := \hat{\rho}_{t-u-1}\mathbf{v}_{t-u-1}^T\mathbf{p}_u$. *Further define the sequence of vectors* $\mathbf{q}_k$ *with initial value* $\mathbf{q}_0 = \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$ *and subsequent elements*

$$\mathbf{q}_{u+1} \;=\; \mathbf{q}_u + (\alpha_{\tau-u-1} - \beta_u)\mathbf{v}_{t-\tau+u} \quad \textit{for } u = 0, \ldots, \tau-1, \tag{3.18}$$

where we define constants $\beta_u := \hat{\rho}_{t-\tau+u}\hat{\mathbf{r}}_{t-\tau+u}^T\mathbf{q}_u$. *The product* $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ *equals* $\mathbf{q}_\tau$, *i.e.,* $\hat{\mathbf{B}}_t^{-1}\mathbf{p} \;=\; \mathbf{q}_\tau$.

Proof: We begin by observing that the $\mathbf{p}_u$ sequence in (3.17) is defined so that we can write $\mathbf{p}_{u+1} = \hat{\mathbf{Z}}_{t-u-1}\mathbf{p}_u$ with $\mathbf{p}_0 = \mathbf{p}$. Indeed, use the explicit expression for $\hat{\mathbf{Z}}_{t-u-1}$ in (3.12) to write the product $\hat{\mathbf{Z}}_{t-u-1}\mathbf{p}_u$ as

$$\hat{\mathbf{Z}}_{t-u-1}\mathbf{p}_u \;=\; \Big(\mathbf{I} - \hat{\rho}_{t-u-1}\hat{\mathbf{r}}_{t-u-1}\mathbf{v}_{t-u-1}^T\Big)\mathbf{p}_u \;=\; \mathbf{p}_u - \alpha_u\hat{\mathbf{r}}_{t-u-1} \;=\; \mathbf{p}_{u+1}, \tag{3.19}$$

where the second equality follows from the definition $\alpha_u := \hat{\rho}_{t-u-1}\mathbf{v}_{t-u-1}^T\mathbf{p}_u$ and the third equality from the definition of the $\mathbf{p}_u$ sequence in (3.17).

Recall now the oLBFGS Hessian inverse approximation expression in (3.16). It follows that for computing the product $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ we can multiply each of the $\tau+1$ summands in the right hand side of (3.16) by $\mathbf{p} = \mathbf{p}_0$. Implementing this procedure yields

$$\begin{aligned}\hat{\mathbf{B}}_t^{-1}\mathbf{p} = &\Big(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau}^T\Big)\,\hat{\mathbf{B}}_{t,0}^{-1}\,\Big(\hat{\mathbf{Z}}_{t-\tau}\ldots\hat{\mathbf{Z}}_{t-1}\Big)\,\mathbf{p}_0\\ &+\cdots+\hat{\rho}_{t-\tau}\Big(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau+1}^T\Big)\,\mathbf{v}_{t-\tau}\mathbf{v}_{t-\tau}^T\,\Big(\hat{\mathbf{Z}}_{t-\tau+1}\ldots\hat{\mathbf{Z}}_{t-1}\Big)\,\mathbf{p}_0\\ &+\cdots+\hat{\rho}_{t-2}\Big(\hat{\mathbf{Z}}_{t-1}^T\Big)\,\mathbf{v}_{t-2}\mathbf{v}_{t-2}^T\,\Big(\hat{\mathbf{Z}}_{t-1}\Big)\,\mathbf{p}_0 + \hat{\rho}_{t-1}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^T\mathbf{p}_0.\end{aligned} \tag{3.20}$$

The fundamental observation in (3.20) is that all summands except the last contain the product $\hat{\mathbf{Z}}_{t-1}\mathbf{p}_0$. This product cannot only be computed efficiently but, as shown in (3.19), is given by $\mathbf{p}_1 = \hat{\mathbf{Z}}_{t-1}\mathbf{p}_0$. A not so fundamental, yet still important observation, is that the last term can be simplified to $\hat{\rho}_{t-1}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^T\mathbf{p}_0 = \alpha_0\mathbf{v}_{t-1}$ given the definition of $\alpha_0 := \hat{\rho}_{t-1}\mathbf{v}_{t-1}^T\mathbf{p}_0$. Implementing both of these substitutions in (3.20) yields

$$\begin{aligned}\hat{\mathbf{B}}_t^{-1}\mathbf{p} = &\Big(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau}^T\Big)\,\hat{\mathbf{B}}_{t,0}^{-1}\,\Big(\hat{\mathbf{Z}}_{t-\tau}\ldots\hat{\mathbf{Z}}_{t-2}\Big)\,\mathbf{p}_1\\ &+\hat{\rho}_{t-\tau}\Big(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau+1}^T\Big)\,\mathbf{v}_{t-\tau}\mathbf{v}_{t-\tau}^T\,\Big(\hat{\mathbf{Z}}_{t-\tau+1}\ldots\hat{\mathbf{Z}}_{t-2}\Big)\,\mathbf{p}_1\\ &+\cdots+\hat{\rho}_{t-2}\Big(\hat{\mathbf{Z}}_{t-1}^T\Big)\,\mathbf{v}_{t-2}\mathbf{v}_{t-2}^T\mathbf{p}_1 + \alpha_0\mathbf{v}_{t-1}.\end{aligned} \tag{3.21}$$

The structure of (3.21) is analogous to the structure of (3.20). In all terms except the last two we require determination of the product $\hat{\mathbf{Z}}_{t-2}\mathbf{p}_1$, which, as per (3.19) can be computed with $2n$ multiplications and is given by $\mathbf{p}_2 = \hat{\mathbf{Z}}_{t-2}\mathbf{p}_1$. Likewise, in the second to last term we can simplify the product $\hat{\rho}_{t-2}\mathbf{v}_{t-2}\mathbf{v}_{t-2}^T\mathbf{p}_1 = \alpha_1\mathbf{v}_{t-2}$ using the definition $\alpha_1 = \hat{\rho}_{t-2}\mathbf{v}_{t-2}^T\mathbf{p}_1$.

Implementing these substitutions in (3.21) yields an expression that is, again, analogous. In all of the resulting summands except the last three we need to compute the product $\hat{\mathbf{Z}}_{t-3}\mathbf{p}_2$, which is given by $\mathbf{p}_3 = \hat{\mathbf{Z}}_{t-3}\mathbf{p}_2$ and in the third to last term we can simplify the product $\hat{\rho}_{t-3}\mathbf{v}_{t-3}\mathbf{v}_{t-3}^T\mathbf{p}_2 = \alpha_2\mathbf{v}_{t-3}$. Repeating this process keeps yielding terms with analogous structure and, after $\tau - 1$ repetitions we simplify (3.21) to

$$\begin{aligned}\hat{\mathbf{B}}_t^{-1}\mathbf{p} = &\left(\hat{\mathbf{Z}}_{t-1}^T \ldots \hat{\mathbf{Z}}_{t-\tau+1}^T \hat{\mathbf{Z}}_{t-\tau}^T\right)\hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau \\ &+ \left(\hat{\mathbf{Z}}_{t-1}^T \ldots \hat{\mathbf{Z}}_{t-\tau+1}^T\right)\alpha_{\tau-1}\mathbf{v}_{t-\tau} + \ldots + \hat{\mathbf{Z}}_{t-1}^T\alpha_1\mathbf{v}_{t-2} + \alpha_0\mathbf{v}_{t-1}. \end{aligned} \tag{3.22}$$

In the first summand in (3.22) we can substitute the definition of the first element of the $\mathbf{q}_u$ sequence $\mathbf{q}_0 := \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$. More important, observe that the matrix $\hat{\mathbf{Z}}_{t-1}^T$ is the first factor in all but the last summand. Likewise, the matrix $\hat{\mathbf{Z}}_{t-2}^T$ is the second factor in all but the last two summands and, in general, the matrix $\hat{\mathbf{Z}}_{t-u}^T$ is the uth factor in all but the last u summands. Pulling these common factors recursively through (3.22) it follows that $\hat{\mathbf{B}}_t^{-1}\mathbf{p}_t$ can be equivalently written as

$$\begin{aligned}\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \hat{\mathbf{Z}}_{t-1}^T\Bigg[\alpha_1\mathbf{v}_{t-2} + \hat{\mathbf{Z}}_{t-2}^T\Bigg[\ldots\Big[\alpha_{\tau-2}\mathbf{v}_{t-\tau+1} + \hat{\mathbf{Z}}_{t-\tau+1}^T\big[\alpha_{\tau-1}\mathbf{v}_{t-\tau} + \hat{\mathbf{Z}}_{t-\tau}^T\mathbf{q}_0\big]\Big]\ldots\Bigg]\Bigg] \\ + \alpha_0\mathbf{v}_{t-1}\end{aligned} \tag{3.23}$$

To conclude the proof we just need to note that the recursive definition of $\mathbf{q}_u$ in (3.18) is a computation of the nested elements of (3.23). To see this consider the innermost element of (3.23) and use the definition of $\beta_0 := \hat{\rho}_{t-\tau}\hat{\mathbf{r}}_{t-\tau}^T\mathbf{q}_0$ to conclude that $\alpha_{\tau-1}\mathbf{v}_{t-\tau} + \hat{\mathbf{Z}}_{t-\tau}^T\mathbf{q}_0$ is given by

$$\alpha_{\tau-1}\mathbf{v}_{t-\tau} + \hat{\mathbf{Z}}_{t-\tau}^T\mathbf{q}_0 \;=\; \alpha_{\tau-1}\mathbf{v}_{t-\tau} + \mathbf{q}_0 - \hat{\rho}_{t-\tau}\mathbf{v}_{t-\tau}\hat{\mathbf{r}}_{t-\tau}^T\mathbf{q}_0 \;=\; \mathbf{q}_0 + (\alpha_{\tau-1} - \beta_0)\mathbf{v}_{t-\tau} \;=\; \mathbf{q}_1 \tag{3.24}$$

where in the last equality we use the definition of $\mathbf{q}_1$ [cf. (3.18). Substituting this simplification into (3.23) eliminates the innermost nested term and leads to

$$\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \alpha_0\mathbf{v}_{t-1} + \hat{\mathbf{Z}}_{t-1}^T\Bigg[\alpha_1\mathbf{v}_{t-2} + \hat{\mathbf{Z}}_{t-2}^T\Bigg[\ldots\Big[\alpha_{\tau-2}\mathbf{v}_{t-\tau+1} + \hat{\mathbf{Z}}_{t-\tau+1}^T\mathbf{q}_1\Big]\ldots\Bigg]\Bigg]. \tag{3.25}$$

Mimicking the computations in (3.24) we can see that the innermost term in (3.25) is $\alpha_{\tau-2}\mathbf{v}_{t-\tau+1} + \hat{\mathbf{Z}}_{t-\tau+1}^T\mathbf{q}_1 = \mathbf{q}_2$ and obtain an analogous expression that we can substitute for $\mathbf{q}_3$ and so on. Repeating this process $\tau - 2$ times leads to the last term being $\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \alpha_0\mathbf{v}_{t-1} + \hat{\mathbf{Z}}_{t-1}^T\mathbf{q}_{\tau-1}$ which we can write as $\alpha_0\mathbf{v}_{t-1} + \hat{\mathbf{Z}}_{t-1}^T\mathbf{q}_{\tau-1} = \mathbf{q}_\tau$ by repeating the operations in (3.24). This final observation yields $\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \mathbf{q}_\tau$. ■

Algorithm 2 Computation of oLBFGS step $\mathbf{q} = \hat{\mathbf{B}}_t^{-1}\mathbf{p}$ when called with $\mathbf{p} = \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$.

function $\mathbf{q} = \mathbf{q}_\tau = \text{oLBFGS Step}\Big(\hat{\mathbf{B}}_{t,0}^{-1},\ \mathbf{p} = \mathbf{p}_0,\ \{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=t-\tau}^{t-1}\Big)$
for $u = 0, 1, \ldots, \tau - 1$ **do** [Loop to compute constants α_u and sequence $\mathbf{p}_u$]
    Compute and store scalar $\alpha_u = \hat{\rho}_{t-u-1}\mathbf{v}_{t-u-1}^T\mathbf{p}_u$
    Update sequence vector $\mathbf{p}_{u+1} = \mathbf{p}_u - \alpha_u\hat{\mathbf{r}}_{t-u-1}$. [cf. (3.17)]
end for
Multiply $\mathbf{p}_\tau$ by initial matrix: $\mathbf{q}_0 = \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$
for $u = 0, 1, \ldots, \tau - 1$ **do** [Loop to compute constants β_u and sequence $\mathbf{q}_u$]
    Compute scalar $\beta_u = \hat{\rho}_{t-\tau+u}\hat{\mathbf{r}}_{t-\tau+u}^T\mathbf{q}_u$
    Update sequence vector $\mathbf{q}_{u+1} = \mathbf{q}_u + (\alpha_{\tau-u-1} - \beta_u)\mathbf{v}_{t-\tau+u}$ [cf. (3.18)]
end for {return $\mathbf{q} = \mathbf{q}_\tau$}

The reorganization of computations described in Proposition 2 has been done for the deterministic LBFGS method in, e.g., [87]. We have used the same technique here for computing the descent direction of oLBFGS and have shown the result and derivations for completeness. In any event, Proposition 2 asserts that it is possible to reduce the computation of the product $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ between the oLBFGS Hessian approximation matrix and arbitrary vector $\mathbf{p}$ to the computation of two vector sequences $\{\mathbf{p}_u\}_{u=0}^{\tau-1}$ and $\{\mathbf{q}_u\}_{u=0}^{\tau-1}$. The product $\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \mathbf{q}_\tau$ is given by the last element of the latter sequence. Since determination of each of the elements of each sequence requires $O(p)$ operations and the total number of elements in each sequence is τ the total operation cost to compute both sequences is of order $O(\tau p)$. In computing $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ we also need to add the cost of the product $\mathbf{q}_0 = \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$ that links both sequences. To maintain overall computation cost of order $O(\tau p)$ this matrix has to have a sparse or low rank structure. A common choice in LBFGS, that we adopt for oLBFGS, is to make $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t\mathbf{I}$. The scalar constant $\hat{\gamma}_t$ is a function of the variable and stochastic gradient variations $\mathbf{v}_{t-1}$ and $\hat{\mathbf{r}}_{t-1}$, explicitly given by

$$\hat{\gamma}_t \;=\; \frac{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}}{\hat{\mathbf{r}}_{t-1}^T\hat{\mathbf{r}}_{t-1}} \;=\; \frac{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}}{\|\hat{\mathbf{r}}_{t-1}\|^2}\,. \tag{3.26}$$

with the value at the first iteration being $\hat{\gamma}_0 = 1$. The scaling factor $\hat{\gamma}_t$ attempts to estimate one of the eigenvalues of the Hessian matrix at step t and has been observed to work well in practice; see e.g., [55,87]. Further observe that the cost of computing $\hat{\gamma}_t$ is of order $O(p)$ and that since $\hat{\mathbf{B}}_{t,0}^{-1}$ is diagonal cost of computing the product $\mathbf{q}_0 = \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$ is also of order $O(p)$. We adopt the initialization in (3.26) in our subsequent analysis and numerical experiments.

The computation of the product $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ using the result in Proposition 2 is summarized in algorithmic form in the function in Algorithm 2. The function receives as arguments the initial matrix $\hat{\mathbf{B}}_{t,0}^{-1}$, the sequence of variable and stochastic gradient variations $\{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=t-\tau}^{t-1}$ and the vector $\mathbf{p}$ to produce the outcome $\mathbf{q} = \mathbf{q}_\tau = \hat{\mathbf{B}}_t^{-1}\mathbf{p}$. When called with the stochastic

Algorithm 3 oLBFGS

Require: Initial value $\mathbf{w}_0$. Initial Hessian approximation parameter $\hat{\gamma}_0 = 1$.

1: **for** $t = 0, 1, 2, \ldots$ **do**

2: Acquire L independent samples $\tilde{\boldsymbol{\theta}}_t = [\boldsymbol{\theta}_{t1}, \ldots, \boldsymbol{\theta}_{tL}]$

3: Compute stochastic gradient: $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) = \dfrac{1}{L}\displaystyle\sum_{l=1}^{L} \nabla_{\mathbf{w}} f(\mathbf{w}_t, \boldsymbol{\theta}_{tl})$ [cf. (3.9)]

4: Initialize Hessian inverse estimate as $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t \mathbf{I}$ with $\hat{\gamma}_t = \dfrac{\mathbf{v}_{t-1}^T \hat{\mathbf{r}}_{t-1}}{\hat{\mathbf{r}}_{t-1}^T \hat{\mathbf{r}}_{t-1}}$ for $t > 0$ [cf. (3.26)]

5: Compute $\hat{\mathbf{d}}_t$ with Algorithm 2: $\hat{\mathbf{d}}_t = \text{oLBFGS Step}\left(\hat{\mathbf{B}}_{t,0}^{-1},\ \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t),\ \{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=t-\tau}^{t-1}\right)$

6: Descend along direction $\hat{\mathbf{d}}_t$: $\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t \hat{\mathbf{d}}_t$ [cf. (3.10)]

7: Compute stochastic gradient: $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) = \dfrac{1}{L}\displaystyle\sum_{l=1}^{L} \nabla_{\mathbf{w}} f(\mathbf{w}_{t+1}, \boldsymbol{\theta}_{tl})$ [cf. (3.9)]

8: Variations $\mathbf{v}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$ [cf. (3.4)] $\hat{\mathbf{r}}_t = \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ [cf.(3.11)]

9: **end for**

gradient $\mathbf{p} = \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$, the function outputs the oLBFGS step $\hat{\mathbf{d}}_t := \hat{\mathbf{B}}_t^{-1} \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ needed to implement the oLBFGS descent step in (3.10). The core of Algorithm 2 is given by the loop in steps 2-5 that computes the constants α_u and sequence elements $\mathbf{p}_u$ as well as the loop in steps 7-10 that computes the constants β_u and sequence elements $\mathbf{q}_u$. The two loops are linked by the initialization of the second sequence with the outcome of the first which is performed in Step 6. To implement the first loop we require τ inner products in Step 4 and τ vector summations in Step 5 which yield a total of $2\tau p$ multiplications. Likewise, the second loop requires τ inner products and τ vector summations in steps 9 and 10, respectively, which yields a total cost of also $2\tau p$ multiplications. Since the initial Hessian inverse approximation matrix $\hat{\mathbf{B}}_{t,0}^{-1}$ is diagonal the cost of computation $\hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$ in Step 6 is p multiplications. Thus, Algorithm 2 requires a total of $(4\tau + 1)p$ multiplications which affirms the complexity cost of order $O(\tau p)$ for oLBFGS.

For reference, oLBFGS is also summarized in algorithmic form in Algorithm 3. As with any stochastic descent algorithm the descent iteration is implemented in three steps: the acquisition of L samples in Step 2, the computation of the stochastic gradient in Step 3, and the implementation of the descent update on the variable $\mathbf{w}_t$ in Step 6. Steps 4 and 5 are devoted to the computation of the oLBFGS descent direction $\hat{\mathbf{d}}_t$. In Step 4 we initialize the estimate $\hat{\mathbf{B}}_{t,0} = \hat{\gamma}_t \mathbf{I}$ as a scaled identity matrix using the expression for $\hat{\gamma}_t$ in (3.26) for $t > 0$. The value of $\gamma_t = \gamma_0$ for $t = 0$ is left as an input for the algorithm. We use $\hat{\gamma}_0 = 1$ in our numerical tests. In Step 5 we use Algorithm 2 for efficient computation of the descent direction $\hat{\mathbf{d}}_t = \hat{\mathbf{B}}_t^{-1} \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. Step 7 determines the value of the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ so that the variable variations $\mathbf{v}_t$ and stochastic gradient variations $\hat{\mathbf{r}}_t$ become available for the computation of the curvature approximation matrix $\hat{\mathbf{B}}_t^{-1}$. In Step 8 the variable variation $\mathbf{v}_t$ and stochastic gradient variation $\hat{\mathbf{r}}_t$ are computed to be used

in the next iteration. We analyze convergence properties of this algorithm in Section 3.3, study its application to SVMs in Section 3.4, and develop an application to search engine advertisement in Section 3.5.

3.3 Convergence analysis

For the subsequent analysis it is convenient to define the instantaneous objective function associated with samples $\tilde{\boldsymbol{\theta}} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L]$ as

$$\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) := \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{w}, \boldsymbol{\theta}_l). \tag{3.27}$$

The definition of the instantaneous objective function $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ in association with the fact that $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ implies that

$$F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})]. \tag{3.28}$$

Our goal here is to show that as time progresses the sequence of variable iterates $\mathbf{w}_t$ approaches the optimal argument $\mathbf{w}^*$. In proving this result we make the following assumptions.

Assumption 4 The instantaneous functions $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ are twice differentiable and the eigenvalues of the instantaneous Hessian $\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) = \nabla^2_{\mathbf{w}} \hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ are bounded between constants $0 < \tilde{m}$ and $\tilde{M} < \infty$ for all random variables $\tilde{\boldsymbol{\theta}}$,

$$\tilde{m}\mathbf{I} \;\preceq\; \hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) \;\preceq\; \tilde{M}\mathbf{I}. \tag{3.29}$$

Assumption 5 The second moment of the norm of the stochastic gradient is bounded for all $\mathbf{w}$. i.e., there exists a constant S^2 such that for all variables $\mathbf{w}$ it holds

$$\mathbb{E}_{\boldsymbol{\theta}}\left[\|\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\|^2 \,\middle|\, \mathbf{w}_t\right] \leq S^2. \tag{3.30}$$

Assumption 6 The step size sequence is selected as nonsummable but square summable, i.e.,

$$\sum_{t=0}^{\infty} \epsilon_t = \infty, \quad \text{and} \quad \sum_{t=0}^{\infty} \epsilon_t^2 < \infty. \tag{3.31}$$

Assumptions 5 and 6 are customary in stochastic optimization. The restriction imposed by Assumption 5 is intended to limit the random variation of stochastic gradients. If the variance of their norm is unbounded it is possible to have rare events that derail progress

towards convergence. The condition in Assumption 6 balances descent towards optimal arguments – which requires a slowly decreasing stepsize – with the eventual elimination of random variations – which requires rapidly decreasing stepsizes. An effective step size choice for which Assumption 6 holds is to make $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$, for given parameters ϵ_0 and T_0 that control the initial step size and its speed of decrease, respectively. Assumption 4 is stronger than usual and specific to oLBFGS. Observe that considering the linearity of the expectation operator and the expression in (3.28) it follows that the Hessian of the average function can be written as $\nabla^2_{\mathbf{w}} F(\mathbf{w}) = \mathbf{H}(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})]$. Combining this observation with the bounds in (3.29) we conclude that there are constants $m \geq \tilde{m}$ and $M \leq \tilde{M}$ such that

$$\tilde{m}\mathbf{I} \preceq m\mathbf{I} \preceq \mathbf{H}(\mathbf{w}) \preceq M\mathbf{I} \preceq \tilde{M}\mathbf{I}. \tag{3.32}$$

The bounds in (3.32) are customary in convergence proofs of descent methods. For the results here the stronger condition spelled in Assumption 4 is needed. This assumption in necessary to guarantee that the inner product $\hat{\mathbf{r}}_t^T \mathbf{v}_t > 0$ is positive as we show in the following lemma.

Lemma 4 *Consider the stochastic gradient variation $\hat{\mathbf{r}}_t$ defined in* (3.11) *and the variable variation $\mathbf{v}_t$ defined in* (3.4)*. Let Assumption 4 hold so that we have lower and upper bounds $\tilde{m}$ and $\tilde{M}$ on the eigenvalues of the instantaneous Hessians. Then, for all steps t the inner product of variable and stochastic gradient variations $\hat{\mathbf{r}}_t^T \mathbf{v}_t$ is bounded below as*

$$\tilde{m}\|\mathbf{v}_t\|^2 \leq \hat{\mathbf{r}}_t^T \mathbf{v}_t \ . \tag{3.33}$$

Furthermore, the ratio of stochastic gradient variation squared norm $\|\hat{\mathbf{r}}_t\|^2 = \hat{\mathbf{r}}_t^T \hat{\mathbf{r}}_t$ to inner product of variable and stochastic gradient variations is bounded as

$$\tilde{m} \ \leq \ \frac{\hat{\mathbf{r}}_t^T \hat{\mathbf{r}}_t}{\hat{\mathbf{r}}_t^T \mathbf{v}_t} \ = \ \frac{\|\hat{\mathbf{r}}_t\|^2}{\hat{\mathbf{r}}_t^T \mathbf{v}_t} \ \leq \ \tilde{M}. \tag{3.34}$$

Proof: As per (3.29) in Assumption 1 the eigenvalues of the instantaneous Hessian $\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ are bounded by $\tilde{m}$ and $\tilde{M}$. Thus, for any given vector $\mathbf{z}$ it holds

$$\tilde{m}\|\mathbf{z}\|^2 \leq \mathbf{z}^T \hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})\mathbf{z} \leq \tilde{M}\|\mathbf{z}\|^2. \tag{3.35}$$

For given $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$ define the mean instantaneous Hessian $\hat{\mathbf{G}}_t$ as the average Hessian value along the segment $[\mathbf{w}_t, \mathbf{w}_{t+1}]$

$$\hat{\mathbf{G}}_t = \int_0^1 \hat{\mathbf{H}} \left(\mathbf{w}_t + \tau(\mathbf{w}_{t+1} - \mathbf{w}_t), \tilde{\boldsymbol{\theta}}_t \right) d\tau. \tag{3.36}$$

Consider now the instantaneous gradient $\hat{\mathbf{s}}(\mathbf{w}_t+\tau(\mathbf{w}_{t+1}-\mathbf{w}_t),\tilde{\boldsymbol{\theta}}_t)$ evaluated at $\mathbf{w}_t+\tau(\mathbf{w}_{t+1}-\mathbf{w}_t)$ and observe that its derivative with respect to τ is $\partial\hat{\mathbf{s}}\big(\mathbf{w}_t+\tau(\mathbf{w}_{t+1}-\mathbf{w}_t),\tilde{\boldsymbol{\theta}}_t\big)/\partial\tau = \hat{\mathbf{H}}(\mathbf{w}_t+\tau(\mathbf{w}_{t+1}-\mathbf{w}_t),\tilde{\boldsymbol{\theta}}_t)(\mathbf{w}_{t+1}-\mathbf{w}_t)$. Then according to the fundamental theorem of calculus

$$\int_0^1 \hat{\mathbf{H}}\left(\mathbf{w}_t+\tau(\mathbf{w}_{t+1}-\mathbf{w}_t)\,,\,\tilde{\boldsymbol{\theta}}_t\right)(\mathbf{w}_{t+1}-\mathbf{w}_t)\;d\tau \;=\; \hat{\mathbf{s}}(\mathbf{w}_{t+1},\tilde{\boldsymbol{\theta}}_t)-\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t). \tag{3.37}$$

Using the definitions of the mean instantaneous Hessian $\hat{\mathbf{G}}_t$ in (3.36) as well as the definitions of the stochastic gradient variations $\hat{\mathbf{r}}_t$ and variable variations $\mathbf{v}_t$ in (3.11) and (3.4) we can rewrite (3.37) as

$$\hat{\mathbf{G}}_t\mathbf{v}_t=\hat{\mathbf{r}}_t. \tag{3.38}$$

Invoking (3.35) for the integrand in (3.36), i.e., for $\hat{\mathbf{H}}(\mathbf{w},\tilde{\boldsymbol{\theta}})=\hat{\mathbf{H}}\big(\mathbf{w}_t+\tau(\mathbf{w}_{t+1}-\mathbf{w}_t),\tilde{\boldsymbol{\theta}}\big)$, it follows that for all vectors $\mathbf{z}$ the mean instantaneous Hessian $\hat{\mathbf{G}}_t$ satisfies

$$\tilde{m}\|\mathbf{z}\|^2\leq\mathbf{z}^T\hat{\mathbf{G}}_t\mathbf{z}\leq\tilde{M}\|\mathbf{z}\|^2. \tag{3.39}$$

The claim in (3.33) follows from (3.38) and (3.39). Indeed, consider the ratio of inner products $\hat{\mathbf{r}}_t^T\mathbf{v}_t/\mathbf{v}_t^T\mathbf{v}_t$ and use (3.38) and the first inequality in (3.39) to write

$$\frac{\hat{\mathbf{r}}_t^T\mathbf{v}_t}{\mathbf{v}_t^T\mathbf{v}_t}=\frac{\mathbf{v}_t^T\hat{\mathbf{G}}_t\mathbf{v}_t}{\mathbf{v}_t^T\mathbf{v}_t}\geq\tilde{m}. \tag{3.40}$$

It follows that (3.33) is true for all times t.

To prove (3.34) we operate (3.38) and (3.39). Considering the ratio of inner products $\hat{\mathbf{r}}_t^T\hat{\mathbf{r}}_t/\hat{\mathbf{r}}_t^T\mathbf{v}_t$ and observing that (3.38) states $\hat{\mathbf{G}}_t\mathbf{v}_t=\hat{\mathbf{r}}_t$, we can write

$$\frac{\hat{\mathbf{r}}_t^T\hat{\mathbf{r}}_t}{\hat{\mathbf{r}}_t^T\mathbf{v}_t}=\frac{\mathbf{v}_t^T\hat{\mathbf{G}}_t^2\mathbf{v}_t}{\mathbf{v}_t^T\hat{\mathbf{G}}_t\mathbf{v}_t}. \tag{3.41}$$

Since the mean instantaneous Hessian $\hat{\mathbf{G}}_t$ is positive definite according to (3.39), we can define $\mathbf{z}_t=\hat{\mathbf{G}}_t^{1/2}\mathbf{v}_t$. Substituting this observation into (3.41) we can conclude

$$\frac{\hat{\mathbf{r}}_t^T\hat{\mathbf{r}}_t}{\hat{\mathbf{r}}_t^T\mathbf{v}_t}=\frac{\mathbf{z}_t^T\hat{\mathbf{G}}_t\mathbf{z}_t}{\mathbf{z}_t^T\mathbf{z}_t}. \tag{3.42}$$

Observing (3.42) and the inequalities in (3.39), it follows that (3.34) is true. ■

According to Lemma 4, strong convexity of instantaneous functions $\hat{f}(\mathbf{w},\tilde{\boldsymbol{\theta}})$ guaranties positiveness of the inner product $\mathbf{v}_t^T\hat{\mathbf{r}}_t$ as long as the variable variation is not identically null. In turn, this implies that the constant $\hat{\gamma}_t$ in (3.26) is nonnegative and that, as a

consequence, the initial Hessian inverse approximation $\hat{\mathbf{B}}_{t,0}^{-1}$ is positive definite for all steps t. The positive definiteness of $\hat{\mathbf{B}}_{t,0}^{-1}$ in association with the positiveness of the inner product of variable and stochastic gradient variations $\mathbf{v}_t^T \hat{\mathbf{r}}_t > 0$ further guarantees that all the matrices $\hat{\mathbf{B}}_{t,u+1}^{-1}$, including the matrix $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ in particular, that follow the update rule in (3.13) stay positive definite – see [70] for details. This proves that (3.10) is a proper stochastic descent iteration because the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is moderated by a positive definite matrix. However, this fact alone is not enough to guarantee convergence because the minimum and maximum eigenvalues of $\hat{\mathbf{B}}_t^{-1}$ could become arbitrarily small and arbitrarily large, respectively. To prove convergence we show this is not possible by deriving explicit lower and upper bounds on these eigenvalues.

The analysis is easier if we consider the matrix $\hat{\mathbf{B}}_t$ – as opposed to $\hat{\mathbf{B}}_t^{-1}$. Consider then the update in (3.13), and use the Sherman-Morrison formula to rewrite as an update that relates $\hat{\mathbf{B}}_{t,u+1}$ to $\hat{\mathbf{B}}_{t,u}$,

$$\hat{\mathbf{B}}_{t,u+1} \;=\; \hat{\mathbf{B}}_{t,u} - \frac{\hat{\mathbf{B}}_{t,u}\mathbf{v}_{t-\tau+u}\mathbf{v}_{t-\tau+u}^T\hat{\mathbf{B}}_{t,u}}{\mathbf{v}_{t-\tau+u}^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_{t-\tau+u}} + \frac{\hat{\mathbf{r}}_{t-\tau+u}\hat{\mathbf{r}}_{t-\tau+u}^T}{\mathbf{v}_{t-\tau+u}^T\hat{\mathbf{r}}_{t-\tau+u}}, \tag{3.43}$$

for $u = 0, \ldots, \tau - 1$ and $\hat{\mathbf{B}}_{t,0} = 1/\hat{\gamma}_t \mathbf{I}$ as per (3.26). As in (3.13), the Hessian approximation at step t is $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. In the following lemma we use the update formula in (3.43) to find bounds on the trace and determinant of the Hessian approximation $\hat{\mathbf{B}}_t$.

Lemma 5 *Consider the Hessian approximation $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$ defined by the recursion in* (3.43) *with $\hat{\mathbf{B}}_{t,0} = \hat{\gamma}_t^{-1}\mathbf{I}$ and $\hat{\gamma}_t$ as given by* (3.26)*. If Assumption 4 holds true, the trace* $\mathrm{tr}(\hat{\mathbf{B}}_t)$ *of the Hessian approximation $\hat{\mathbf{B}}_t$ is uniformly upper bounded for all times $t \geq 1$,*

$$\mathrm{tr}\left(\hat{\mathbf{B}}_t\right) \;\leq\; (p+\tau)\tilde{M}. \tag{3.44}$$

Likewise, if Assumption 4 holds true, the determinant $\det(\hat{\mathbf{B}}_t)$ *of the Hessian approximation $\hat{\mathbf{B}}_t$ is uniformly lower bounded for all times t*

$$\det\left(\hat{\mathbf{B}}_t\right) \;\geq\; \frac{\tilde{m}^{p+\tau}}{[(p+\tau)\tilde{M}]^{\tau}}\;. \tag{3.45}$$

Proof: We begin with the trace upper bound in (3.44). Consider the recursive update formula for the Hessian approximation $\hat{\mathbf{B}}_t$ as defined in (3.43). To simplify notation we define s as a new index such that $s = t - \tau + u$. Introduce this simplified notation in (3.43) and compute the trace of both sides. Since traces are linear function of their arguments we obtain

$$\mathrm{tr}\left(\hat{\mathbf{B}}_{t,u+1}\right) = \mathrm{tr}\left(\hat{\mathbf{B}}_{t,u}\right) - \mathrm{tr}\left(\frac{\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}\right) + \mathrm{tr}\left(\frac{\hat{\mathbf{r}}_s\hat{\mathbf{r}}_s^T}{\mathbf{v}_s^T\hat{\mathbf{r}}_s}\right). \tag{3.46}$$

Recall that the trace of a matrix product is independent of the order of the factors to conclude that the second summand of (3.46) can be simplified to

$$\text{tr}\left(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\right) \;=\; \text{tr}\left(\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\right) \;=\; \mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\hat{\mathbf{B}}_{t,u}\mathbf{v}_s \;=\; \left\|\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\right\|^2, \tag{3.47}$$

where the second equality follows because $\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\hat{\mathbf{B}}_{t,u}\mathbf{v}_s$ is a scalar and the second equality by observing that the term $\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\hat{\mathbf{B}}_{t,u}\mathbf{v}_s$ is the inner product of the vector $\hat{\mathbf{B}}_{t,u}\mathbf{v}_s$ with itself. Use the same procedure for the last summand of (3.46) so as to write $\text{tr}(\hat{\mathbf{r}}_s\hat{\mathbf{r}}_s^T) = \hat{\mathbf{r}}_s^T\hat{\mathbf{r}}_s = \|\hat{\mathbf{r}}_s\|^2$. Substituting this latter observation as well as (3.47) into (3.46) we can simplify the trace of $\hat{\mathbf{B}}_{t,u+1}$ to

$$\text{tr}\left(\hat{\mathbf{B}}_{t,u+1}\right) = \text{tr}\left(\hat{\mathbf{B}}_{t,u}\right) - \frac{\|\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\|^2}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s} + \frac{\|\hat{\mathbf{r}}_s\|^2}{\hat{\mathbf{r}}_s^T\mathbf{v}_s}. \tag{3.48}$$

The second term in the right hand side of (3.48) is negative because, as we have already shown, the matrix $\hat{\mathbf{B}}_{t,u}$ is positive definite. The third term is the one for which we have derived the bound that appears in (3.34) of Lemma 4. Using this two observations we can conclude that the trace of $\hat{\mathbf{B}}_{t,u+1}$ can be bounded as

$$\text{tr}\left(\hat{\mathbf{B}}_{t,u+1}\right) \leq \text{tr}\left(\hat{\mathbf{B}}_{t,u}\right) + \tilde{M}. \tag{3.49}$$

By considering (3.49) as a recursive expression for $u = 0, \ldots \tau - 1$, we can conclude that

$$\text{tr}\left(\hat{\mathbf{B}}_{t,u}\right) \leq \text{tr}\left(\hat{\mathbf{B}}_{t,0}\right) + u\tilde{M}. \tag{3.50}$$

To finalize the proof of (3.44) we need to find a bound for the initial trace $\text{tr}(\hat{\mathbf{B}}_{t,0})$. To do so we consider the definition $\hat{\mathbf{B}}_{t,0} = \mathbf{I}/\hat{\gamma}_t$ with $\hat{\gamma}_t$ as given by (3.26). Using this definition of $\hat{\mathbf{B}}_{t,0}$ as a scaled identity it follows that we can write the trace of $\hat{\mathbf{B}}_{t,0}$ as

$$\text{tr}\left(\hat{\mathbf{B}}_{t,0}\right) \;=\; \text{tr}\left(\frac{\mathbf{I}}{\hat{\gamma}_t}\right) \;=\; \frac{p}{\hat{\gamma}_t}. \tag{3.51}$$

Substituting the definition of $\hat{\gamma}_t$ into the rightmost side of (3.26) it follows that for all times $t \geq 1$,

$$\text{tr}\left(\hat{\mathbf{B}}_{t,0}\right) \;=\; p\,\frac{\hat{\mathbf{r}}_{t-1}^T\hat{\mathbf{r}}_{t-1}}{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}} \;=\; p\,\frac{\|\hat{\mathbf{r}}_{t-1}\|^2}{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}}\,. \tag{3.52}$$

The term $\|\hat{\mathbf{r}}_{t-1}\|^2/\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}$ in (3.64) is of the same form of the rightmost term in (3.48). We can then, as we did in going from (3.48) to (3.49) apply the bound that we provide in

(3.34) of Lemma 4 to conclude that for all times $t \geq 1$

$$\text{tr}\left(\hat{\mathbf{B}}_{t,0}\right) \;\leq\; p\tilde{M}. \tag{3.53}$$

Substituting (3.53) into (3.50) and pulling common factors leads to the conclusion that for all times $t \geq 1$ and indices $0 \leq u \leq \tau$ it holds

$$\text{tr}\left(\hat{\mathbf{B}}_{t,u}\right) \;\leq\; (p+u)\tilde{M}. \tag{3.54}$$

The bound in (3.44) follows by making $u = \tau$ in (3.54) and recalling that, by definition, $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. For time $t = 0$ we have $\hat{\gamma}_t = \hat{\gamma}_0 = 1$ and (3.64) reduces to $\text{tr}(\hat{\mathbf{B}}_{t,0}) = p$ while (3.54) reduces to $\text{tr}(\hat{\mathbf{B}}_{t,\tau}) \leq (1+\tau)\tilde{M}$. Furthermore, for $t < \tau$ we make $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,t}$ instead of $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. In this case the bound in (3.54) can be tightened to $\text{tr}(\hat{\mathbf{B}}_{t,\tau}) \leq (p+t)\tilde{M}$. Given that we are interested in an asymptotic convergence analysis, these bounds are inconsequential.

We consider now the determinant lower bound in (3.45). As we did in (3.46) begin by considering the recursive update in (3.43) and define s as a new index such that $s = t-\tau+u$ to simplify notation. Compute the determinant of both sides of (3.43), factorize $\hat{\mathbf{B}}_{t,u}$ on the right hand side, and use the fact that the determinant of a product is the product of the determinants to conclude that

$$\det\left(\hat{\mathbf{B}}_{t,u+1}\right) = \det\left(\hat{\mathbf{B}}_{t,u}\right)\det\left(\mathbf{I} - \frac{\mathbf{v}_s(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s)^T}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s} + \frac{\hat{\mathbf{B}}_{t,u}^{-1}\hat{\mathbf{r}}_s\hat{\mathbf{r}}_s^T}{\hat{\mathbf{r}}_s^T\mathbf{v}_s}\right). \tag{3.55}$$

To simplify the right hand side of (3.55) we should first know that for any vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$ and $\mathbf{u}_4$, we can write $\det(\mathbf{I}+\mathbf{u}_1\mathbf{u}_2^T+\mathbf{u}_3\mathbf{u}_4^T) = (1+\mathbf{u}_1^T\mathbf{u}_2)(1+\mathbf{u}_3^T\mathbf{u}_4)-(\mathbf{u}_1^T\mathbf{u}_4)(\mathbf{u}_2^T\mathbf{u}_3)$ – see, e.g., [51], Lemma 3.3). Setting $\mathbf{u}_1 = \mathbf{v}_s$, $\mathbf{u}_2 = \hat{\mathbf{B}}_{t,u}\mathbf{v}_s/\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s$, $\mathbf{u}_3 = \hat{\mathbf{B}}_{t,u}^{-1}\hat{\mathbf{r}}_s$ and $\mathbf{u}_4 = \hat{\mathbf{r}}_s/\hat{\mathbf{r}}_s^T\mathbf{v}_s$, implies that $\det(\mathbf{I}+\mathbf{u}_1\mathbf{u}_2^T+\mathbf{u}_3\mathbf{u}_4^T)$ is equivalent to the last term in the right hand side of (3.55). Applying these substitutions implies that $(1+\mathbf{u}_1^T\mathbf{u}_2) = 1-\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s/\mathbf{v}_s\hat{\mathbf{B}}_{t,u}\mathbf{v}_s = 0$ and $\mathbf{u}_1^T\mathbf{u}_4 = -\mathbf{v}_s^T\hat{\mathbf{r}}_s/\hat{\mathbf{r}}_s^T\mathbf{v}_s = -1$. Hence, the term $\det(\mathbf{I}+\mathbf{u}_1\mathbf{u}_2^T+\mathbf{u}_3\mathbf{u}_4^T)$ can be simplified as $\mathbf{u}_2^T\mathbf{u}_3$. By this simplification we can write the right hand side of (3.55) as

$$\det\left[\mathbf{I} - \frac{\mathbf{v}_s(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s)^T}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s} + \frac{\hat{\mathbf{B}}_{t,u}^{-1}\hat{\mathbf{r}}_s\hat{\mathbf{r}}_s^T}{\hat{\mathbf{r}}_s^T\mathbf{v}_s}\right] = \frac{\left(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\right)^T}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}\hat{\mathbf{B}}_{t,u}^{-1}\hat{\mathbf{r}}_s. \tag{3.56}$$

To further simplify (3.56) write $(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s)^T = \mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}^T$ and observer that since $\hat{\mathbf{B}}_{t,u}$ is symmetric we have $\hat{\mathbf{B}}_{t,u}^T\hat{\mathbf{B}}_{t,u}^{-1} = \hat{\mathbf{B}}_{t,u}\hat{\mathbf{B}}_{t,u}^{-1} = \mathbf{I}$. Therefore,

$$\det\left[\mathbf{I} - \frac{\mathbf{v}_s(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s)^T}{\mathbf{v}_i^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s} + \frac{\hat{\mathbf{B}}_{t,u}^{-1}\hat{\mathbf{r}}_s\hat{\mathbf{r}}_s^T}{\hat{\mathbf{r}}_s^T\mathbf{v}_s}\right] = \frac{\hat{\mathbf{r}}_s^T\mathbf{v}_s}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}. \tag{3.57}$$

Substitute the simplification in (3.57) for the corresponding factor in (3.55). Further multiply and divide the right hand side by the nonzero norm $\|\mathbf{v}_s\|$ and regroup terms to obtain

$$\det\left(\hat{\mathbf{B}}_{t,u+1}\right) = \det\left(\hat{\mathbf{B}}_{t,u}\right)\frac{\hat{\mathbf{r}}_s^T\mathbf{v}_s}{\|\mathbf{v}_s\|}\frac{\|\mathbf{v}_s\|}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}. \tag{3.58}$$

To bound the third factor in (3.58) observe that the largest possible value for the normalized quadratic form $\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s/\|\mathbf{v}_s\|^2$ occurs when $\mathbf{v}_s$ is an eigenvector of $\hat{\mathbf{B}}_{t,u}$ associated with its largest eigenvalue. In such case the value attained is precisely the largest eigenvalue of $\hat{\mathbf{B}}_{t,u}$ implying that we can write

$$\frac{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}{\|\mathbf{v}_s\|} \leq \lambda_{\max}\left(\hat{\mathbf{B}}_{t,u}\right). \tag{3.59}$$

But to bound the largest eigenvalue $\lambda_{\max}(\hat{\mathbf{B}}_{t,u})$ we can just use the fact that the trace of a matrix coincides with the sum of its eigenvalues. In particular, it must be that $\lambda_{\max}(\hat{\mathbf{B}}_{t,u}) \leq \text{tr}(\hat{\mathbf{B}}_{t,u})$ because all the eigenvalues of the positive definite matrix $\hat{\mathbf{B}}_{t,u}$ are positive. Combining this observation with the trace bound in (3.54) leads to

$$\frac{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}{\|\mathbf{v}_s\|} \;\leq\; \text{tr}\left(\hat{\mathbf{B}}_{t,u}\right) \;\leq\; (p+u)\tilde{M}. \tag{3.60}$$

We can also bound the second factor in the right hand side of (3.58) if we reorder the inequality in (3.33) of Lemma 4 to conclude that $\hat{\mathbf{r}}_s^T\mathbf{v}_s/\|\mathbf{v}_s\| \leq \tilde{m}$. This bound, along with the inverse of the inequality in (3.60) substituted in (3.58) leads to

$$\det\left(\hat{\mathbf{B}}_{t,u+1}\right) \geq \frac{\tilde{m}}{p\tilde{M} + u\tilde{M}}\det\left(\hat{\mathbf{B}}_{t,u}\right). \tag{3.61}$$

Apply (3.61) recursively between indexes $u = 0$ and $u = \tau - 1$ and further observing that $u \leq \tau$ in all of the resulting factors it follows that

$$\det\left(\hat{\mathbf{B}}_{t,\tau}\right) \;\geq\; \left[\frac{\tilde{m}}{(p+\tau)\tilde{M}}\right]^{\tau}\det\left(\hat{\mathbf{B}}_{t,0}\right). \tag{3.62}$$

To finalize the derivation of (3.45) we just need to bound the determinant of the initial curvature approximation matrix $\hat{\mathbf{B}}_{t,0}$. To do so we consider, again, the definition $\hat{\mathbf{B}}_{t,0} = \mathbf{I}/\hat{\gamma}_t$

with $\hat{\gamma}_t$ as given by (3.26). Using this definition of $\hat{\mathbf{B}}_{t,0}$ as a scaled identity it follows that we can write the determinant of $\hat{\mathbf{B}}_{t,0}$ as

$$\det\left(\hat{\mathbf{B}}_{t,0}\right) \;=\; \det\left(\frac{\mathbf{I}}{\hat{\gamma}_t}\right) \;=\; \frac{1}{\hat{\gamma}_t^p}. \tag{3.63}$$

Substituting the definition of $\hat{\gamma}_t$ into the rightmost side of (3.63) it follows that for all times $t \geq 1$,

$$\det\left(\hat{\mathbf{B}}_{t,0}\right) \;=\; \left(\frac{\hat{\mathbf{r}}_{t-1}^T\hat{\mathbf{r}}_{t-1}}{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}}\right)^p \;=\; \left(\frac{\|\hat{\mathbf{r}}_{t-1}\|^2}{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}}\right)^p. \tag{3.64}$$

The term $\|\hat{\mathbf{r}}_{t-1}\|^2/\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}$ has lower and upper bounds that we provide in (3.34) of Lemma 4. Using the lower bound in (3.34) it follows that the initial determinant must be such that

$$\det\left(\hat{\mathbf{B}}_{t,0}\right) \geq \tilde{m}^p. \tag{3.65}$$

Substituting the upper bound in (3.65) for the determinant of the initial curvature approximation matrix in (3.62) allows us to conclude that for all times $t \geq 1$

$$\det\left(\hat{\mathbf{B}}_{t,\tau}\right) \;\geq\; \tilde{m}^p\left[\frac{\tilde{m}}{(p+\tau)\tilde{M}}\right]^\tau. \tag{3.66}$$

The bound in (3.45) follows by making $u = \tau$ in (3.66) and recalling that, by definition, $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. At time $t = 0$ the initialization constant is set to $\hat{\gamma}_t = \hat{\gamma}_0 = 1$ and (3.65) reduces to $\det(\hat{\mathbf{B}}_{t,0}) = 1$ while (3.66) reduces to $\det(\hat{\mathbf{B}}_{t,\tau}) \leq [\tilde{m}/(1+\tau)\tilde{M}]^\tau$. For $t < \tau$ we make $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,t}$ instead of $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. In this case the bound in (3.54) can be tightened to $\det(\hat{\mathbf{B}}_{t,\tau}) \leq \tilde{m}[\tilde{m}^p/(1+\tau)\tilde{M}]^\tau$. As in the case of the trace, given that we are interested in an asymptotic convergence analysis, these bounds are inconsequential. ■

Lemma 5 states that the trace and determinants of the Hessian approximation matrix $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$ are bounded for all times $t \geq 1$. For time $t = 0$ we can write a similar bound that takes into account the fact that the constant γ_t that initializes the recursion in (3.43) is $\gamma_0 = 1$. Given that we are interested in an asymptotic convergence analysis, this bound in inconsequential. The bounds on the trace and determinant of $\hat{\mathbf{B}}_t$ are respectivey equivalent to bounds in the sum and product of its eigenvalues. Further considering that the matrix $\hat{\mathbf{B}}_t$ is positive definite, as it follows from Lemma 4, these bounds can be further transformed into bounds on the smalls and largest eigenvalue of $\hat{\mathbf{B}}_t$. The resulting bounds are formally stated in the following lemma.

Lemma 6 *Consider the Hessian approximation $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$ defined by the recursion in* (3.43) *with $\hat{\mathbf{B}}_{t,0} = \hat{\gamma}_t^{-1}\mathbf{I}$ and $\hat{\gamma}_t$ as given by* (3.26)*. Define the strictly positive constant $0 < c := \tilde{m}^{p+\tau}/[(p+\tau)\tilde{M}]^{p+\tau-1}$ and the finite constant $C := (p+\tau)\tilde{M} < \infty$. If Assumption 4 holds*

true, the range of eigenvalues of $\hat{\mathbf{B}}_t$ *is bounded by* c *and* C *for all time steps* $t \geq 1$*, i.e.,*

$$\frac{\tilde{m}^{p+\tau}}{[(p+\tau)\tilde{M}]^{p+\tau-1}}\,\mathbf{I} \;=:\; c\mathbf{I} \;\preceq\; \hat{\mathbf{B}}_t \;\preceq\; C\mathbf{I} \;:=\; (p+\tau)\tilde{M}\,\mathbf{I}. \tag{3.67}$$

Proof: We first prove the upper bound inequality in (3.67). Let us define λ_i as the ith largest eigenvalue of matrix $\hat{\mathbf{B}}_t$. Considering the result in Lemma 5 that $\text{tr}(\hat{\mathbf{B}}_t) \leq (p+\tau)\tilde{M}$ for all steps $t \geq 1$, we obtain that the sum of eigenvalues of the Hessian approximation $\hat{\mathbf{B}}_t$ satisfy

$$\sum_{i=1}^{p} \lambda_i \;=\; \text{tr}\left(\hat{\mathbf{B}}_t\right) \;\leq\; (p+\tau)\tilde{M}. \tag{3.68}$$

Considering the upper bound for the sum of eigenvalues in (3.68) and recalling that all the eigenvalues of the matrix $\hat{\mathbf{B}}_t$ are positive because $\hat{\mathbf{B}}_t$ is positive definite, we can conclude that each of the eigenvalues of $\hat{\mathbf{B}}_t$ is less than the upper bound for their sum in (3.68). We then have $\lambda_i \leq (p+\tau)\tilde{M}$ for all i from where the right inequality in (3.67) follows.

To prove the lower bound inequality in (3.67) consider the second result of Lemma 5 which provides a lower bound for the determinant of the Hessian approximation matrix $\hat{\mathbf{B}}_t$. According to the fact that determinant of a matrix is the product of its eigenvalues, it follows that the product of the eigenvalues of $\hat{\mathbf{B}}_t$ is bounded below by the lower bound in (3.45), or, equivalently, $\prod_{i=1}^{p} \lambda_i \geq \tilde{m}^{p+\tau}/[(p+\tau)\tilde{M}]^{\tau}$. Hence, for any given eigenvalue of $\hat{\mathbf{B}}_t$, say λ_j, we have

$$\lambda_j \;\geq\; \frac{1}{\prod_{k=1,k\neq j}^{p} \lambda_k} \times \frac{\tilde{m}^{p+\tau}}{\left[(p+\tau)\tilde{M}\right]^{\tau}}. \tag{3.69}$$

But in the first part of this proof we have already showed that $(p+\tau)\tilde{M}$ is a lower bound for the eigenvalues of $\hat{\mathbf{B}}_t$. We can then conclude that the product of the $p-1$ eigenvalues $\prod_{k=1,k\neq j}^{p} \lambda_k$ is bounded above by $[(p+\tau)\tilde{M}]^{p-1}$, i.e.,

$$\prod_{k=1,k\neq j}^{n} \lambda_k \leq \left[(p+\tau)\tilde{M}\right]^{p-1}. \tag{3.70}$$

Combining the inequalities in (3.69) and (3.70) we conclude that for any specific eigenvalue of $\hat{\mathbf{B}}_t$ can be lower bounded as

$$\lambda_j \;\geq\; \frac{1}{\left[(p+\tau)\tilde{M}\right]^{p-1}} \times \frac{\tilde{m}^{p+\tau}}{\left[(p+\tau)\tilde{M}\right]^{\tau}}. \tag{3.71}$$

Since inequality (3.71) is true for all the eigenvalues of $\hat{\mathbf{B}}_t$, the left inequality (3.67) holds true. ■

The bounds in Lemma 6 imply that their respective inverses are bounds on the range of the eigenvalues of the Hessian inverse approximation matrix $\hat{\mathbf{B}}_t^{-1}$. Specifically, the minimum eigenvalue of the Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1}$ is larger than $1/C$ and the maximum eigenvalue of $\hat{\mathbf{B}}_t^{-1}$ does not exceed $1/c$, or, equivalently,

$$\frac{1}{C}\,\mathbf{I} \;\preceq\; \hat{\mathbf{B}}_t^{-1} \;\preceq\; \frac{1}{c}\,\mathbf{I}\,. \tag{3.72}$$

We further emphasize that the bounds in (3.72), or (3.67) for that matter, limit the conditioning of $\hat{\mathbf{B}}_t^{-1}$ for all realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$, irrespective of the particular random draw. Having matrices $\hat{\mathbf{B}}_t^{-1}$ that are strictly positive definite with eigenvalues uniformly upper bounded by $1/c$ leads to the conclusion that if $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is a descent direction, the same holds true of $\hat{\mathbf{B}}_t^{-1}\,\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. The stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is not a descent direction in general, but we know that this is true for its conditional expectation $\mathbb{E}[\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\,\big|\,\mathbf{w}_t] = \nabla F(\mathbf{w}_t)$. Hence, we conclude that $\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is an average descent direction since $\mathbb{E}[\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\big|\mathbf{w}_t] = \hat{\mathbf{B}}_t^{-1}\nabla F(\mathbf{w}_t)$. Stochastic optimization methods whose displacements $\mathbf{w}_{t+1} - \mathbf{w}_t$ are descent directions on average are expected to approach optimal arguments. We show that this is true of oLBFGS in the following lemma.

Lemma 7 *Consider the online Limited memory BFGS algorithm as defined by the descent iteration in* (3.10) *with matrices* $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ *obtained after* τ *recursive applications of the update in* (3.13) *initialized with* $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t\mathbf{I}$ *and* $\hat{\gamma}_t$ *as given by* (3.26)*. If Assumptions 4 and 5 hold true, the sequence of average function values* $F(\mathbf{w}_t)$ *satisfies*

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\,\big|\,\mathbf{w}_t\right] \leq F(\mathbf{w}_t) - \frac{\epsilon_t}{C}\|\nabla F(\mathbf{w}_t)\|^2 + \frac{MS^2\epsilon_t^2}{2c^2}. \tag{3.73}$$

Proof: The proof is standard in stochastic optimization and provided here for reference. As it follows from Assumption 1 the eigenvalues of the Hessian $\mathbf{H}(\mathbf{w}_t) = \mathbb{E}_{\tilde{\boldsymbol{\theta}}}[\hat{\mathbf{H}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)] = \nabla^2_{\mathbf{w}} F(\mathbf{w}_t)$ are bounded between $0 < m$ and $M < \infty$ as stated in (3.32). Taking a Taylor's expansion of the function $F(\mathbf{w})$ around $\mathbf{w} = \mathbf{w}_t$ and using the upper bound in the Hessian eigenvalues we can write

$$F(\mathbf{w}_{t+1}) \;\leq F(\mathbf{w}_t) + \nabla F(\mathbf{w}_t)^T(\mathbf{w}_{t+1} - \mathbf{w}_t) + \frac{M}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2. \tag{3.74}$$

From the definition of the oLBFGS update in (3.3) we can write the difference of two consecutive variables $\mathbf{w}_{t+1} - \mathbf{w}_t$ as $-\epsilon_t\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. Making this substitution in (3.74), taking expectation with $\mathbf{w}_t$ given in both sides of the resulting inequality, and observing

the fact that when $\mathbf{w}_t$ is given the Hessian approximation $\hat{\mathbf{B}}_t^{-1}$ is deterministic we can write

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] \leq F(\mathbf{w}_t) - \epsilon_t \nabla F(\mathbf{w}_t)^T \hat{\mathbf{B}}_t^{-1} \mathbb{E}\left[\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) \,\middle|\, \mathbf{w}_t\right] + \frac{\epsilon^2 M}{2} \mathbb{E}\left[\left\|\hat{\mathbf{B}}_t^{-1} \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\right\|^2 \,\middle|\, \mathbf{w}_t\right]. \tag{3.75}$$

We proceed to bound the third term in the right hand side of (3.75). Start by observing that the 2-norm of a product is not larger than the product of the 2-norms and that, as noted above, with $\mathbf{w}_t$ given the matrix $\hat{\mathbf{B}}_t^{-1}$ is also given to write

$$\mathbb{E}\left[\left\|\hat{\mathbf{B}}_t^{-1} \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\right\|^2 \,\middle|\, \mathbf{w}_t\right] \leq \left\|\hat{\mathbf{B}}_t^{-1}\right\|^2 \mathbb{E}\left[\left\|\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\right\|^2 \middle| \mathbf{w}_t\right] \tag{3.76}$$

Notice that, as stated in (3.72), $1/c$ is an upper bound for the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$. Further observe that the second moment of the norm of the stochastic gradient is bounded by $\mathbb{E}\left[\|\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\|^2 \,\middle|\, \mathbf{w}_t\right] \leq S^2$, as stated in Assumption 2. These two upper bounds substituted in (3.76) yield

$$\mathbb{E}\left[\left\|\hat{\mathbf{B}}_t^{-1} \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\right\|^2 \,\middle|\, \mathbf{w}_t\right] \leq \frac{S^2}{c^2}. \tag{3.77}$$

Substituting the upper bound in (3.77) for the third term of (3.75) and further using the fact that $\mathbb{E}\left[\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) \,\middle|\, \mathbf{w}_t\right] = \nabla F(\mathbf{w}_t)$ in the second term leads to

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] \leq F(\mathbf{w}_t) - \epsilon_t \nabla F(\mathbf{w}_t)^T \hat{\mathbf{B}}_t^{-1} \nabla F(\mathbf{w}_t) + \frac{\epsilon_t^2 M S^2}{2c^2}. \tag{3.78}$$

We now find a lower bound for the second term in the right hand side of (3.78). As stated in (3.72), $1/C$ is a lower bound for the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$. This lower bound implies that

$$\nabla F(\mathbf{w}_t)^T \hat{\mathbf{B}}_t^{-1} \nabla F(\mathbf{w}_t) \geq \frac{1}{C} \|\nabla F(\mathbf{w}_t)\|^2 \tag{3.79}$$

By substituting the lower bound in (3.79) for the corresponding summand in (3.78) the result in (3.73) follows. ■

Setting aside the term $MS^2\epsilon_t^2/2c^2$ for the sake of argument, (3.73) defines a supermartingale relationship for the sequence of average functions $F(\mathbf{w}_t)$. This implies that the sequence $\epsilon_t \|\nabla F(\mathbf{w}_t)\|^2/C$ is almost surely summable which, given that the step sizes ϵ_t are nonsummable as per (3.31), further implies that the limit infimum $\liminf_{t\to\infty} \|\nabla F(\mathbf{w}_t)\|$ of the gradient norm $\|\nabla F(\mathbf{w}_t)\|$ is almost surely null. This latter observation is equivalent to having $\liminf_{t\to\infty} \|\mathbf{w}_t - \mathbf{w}^*\|^2 = 0$ with probability 1 over realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$. The term $MS^2\epsilon_t^2/2c^2$ is a relatively minor nuisance that can be taken care of with a technical argument that we present in the proof of the following theorem.

Theorem 3 *Consider the online Limited memory BFGS algorithm as defined by the descent iteration in* (3.10) *with matrices* $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ *obtained after* τ *recursive applications of the update in* (3.13) *initialized with* $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t \mathbf{I}$ *and* $\hat{\gamma}_t$ *as given by* (3.26)*. If Assumptions 4-6 hold true the limit of the squared Euclidean distance to optimality* $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ *converges to zero almost surely, i.e.,*

$$\Pr\Big[\lim_{t\to\infty} \|\mathbf{w}_t - \mathbf{w}^*\|^2 = 0\Big] = 1, \tag{3.80}$$

where the probability is over realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$.

Proof: The proof uses the relationship in the statement (3.73) of Lemma 7 to build a supermartingale sequence. This is also a standard technique in stochastic optimization and provided here for reference. Subtract the optimal objective function value $F(\mathbf{w}^*)$ from the both sides of (3.73) to obtain

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] - F(\mathbf{w}^*) \leq F(\mathbf{w}_t) - F(\mathbf{w}^*) - \frac{\epsilon_t}{C}\|\nabla F(\mathbf{w}_t)\|^2 + \frac{MS^2\epsilon_t^2}{2c^2}. \tag{3.81}$$

To construct the supermartingale sequence define the stochastic process α_t with values

$$\alpha_t := F(\mathbf{w}_t) - F(\mathbf{w}^*) + \frac{MS^2}{2c^2}\sum_{u=t}^{\infty}\epsilon_u^2. \tag{3.82}$$

Observe that α_t is well defined because the $\sum_{u=t}^{\infty}\epsilon_u^2 < \sum_{u=0}^{\infty}\epsilon_u^2 < \infty$ is summable. Further define the sequence β_t with values

$$\beta_t := \frac{\epsilon_t}{C}\|\nabla F(\mathbf{w}_t)\|^2. \tag{3.83}$$

Let now $\mathcal{F}_t$ be a sigma-algebra measuring α_t, β_t, and $\mathbf{w}_t$. The conditional expectation of α_{t+1} given $\mathcal{F}_t$ can be written as

$$\mathbb{E}\left[\alpha_{t+1} \,\middle|\, \mathcal{F}_t\right] = \mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathcal{F}_t\right] - F(\mathbf{w}^*) + \frac{MS^2}{2c^2}\sum_{u=t+1}^{\infty}\epsilon_u^2, \tag{3.84}$$

because the term $(MS^2/2c^2)\sum_{u=t+1}^{\infty}\epsilon_u^2$ is just a deterministic constant. Substituting (3.73) of Lemma 7 into (3.84) and using the definitions of α_t in (3.82) and β_t in (3.83) yields

$$\mathbb{E}\left[\alpha_{t+1} \,\middle|\, \alpha_t\right] \;\leq\; \alpha_t - \beta_t \tag{3.85}$$

Since the sequences α_t and β_t are nonnegative it follows from (3.85) that they satisfy the conditions of the supermartingale convergence theorem – see e.g. (Theorem E7.4 in [114]) . Therefore, we conclude that: (i) The sequence α_t converges almost surely. (ii) The sum $\sum_{t=0}^{\infty}\beta_t < \infty$ is almost surely finite. Using the explicit form of β_t in (3.83) we have that

$\sum_{t=0}^{\infty} \beta_t < \infty$ is equivalent to

$$\sum_{t=0}^{\infty} \frac{\epsilon_t}{C} \|\nabla F(\mathbf{w}_t)\|^2 < \infty, \qquad \text{a.s.} \tag{3.86}$$

Since the sequence of stepsizes is nonsummable, for (3.86) to be true we need to have a vanishing subsequence embedded in $\|\nabla F(\mathbf{w}_t)\|^2$. By definition, this implies that the limit infimum of the sequence $\|\nabla F(\mathbf{w}_t)\|^2$ is null almost surely,

$$\liminf_{t\to\infty} \|\nabla F(\mathbf{w}_t)\|^2 = 0, \qquad \text{a.s.} \tag{3.87}$$

We transform the gradient bound in (3.87) into a bound pertaining to the objective function value optimality $F(\mathbf{w}_t) - F(\mathbf{w}^*)$. To do so, simply observe that the strong convexity of the average function F implies that for any points $\mathbf{z}$ and $\mathbf{y}$

$$F(\mathbf{y}) \geq \; F(\mathbf{z}) + \nabla F(\mathbf{z})^T(\mathbf{y} - \mathbf{z}) + \frac{m}{2}\|\mathbf{y} - \mathbf{z}\|^2. \tag{3.88}$$

For fixed $\mathbf{z}$, the right hand side of (3.88) is a quadratic function of $\mathbf{y}$ whose minimum argument we can find by setting its gradient to zero. Doing this yields the minimizing argument $\hat{\mathbf{y}} = \mathbf{z} - (1/m)\nabla F(\mathbf{z})$ implying that for all $\mathbf{y}$ we must have

$$\begin{aligned} F(\mathbf{y}) \geq \; & F(\mathbf{z}) + \nabla F(\mathbf{z})^T(\hat{\mathbf{y}} - \mathbf{z}) + \frac{m}{2}\|\hat{\mathbf{y}} - \mathbf{z}\|^2 \\ = \; & F(\mathbf{z}) - \frac{1}{2m}\|\nabla F(\mathbf{z})\|^2. \end{aligned} \tag{3.89}$$

Observe that the bound in (3.89) holds true for all $\mathbf{y}$ and $\mathbf{z}$. Setting values $\mathbf{y} = \mathbf{w}^*$ and $\mathbf{z} = \mathbf{w}_t$ in (3.89) and rearranging the terms yields a lower bound for the squared gradient norm $\|\nabla F(\mathbf{x}^t)\|^2$ as

$$\|\nabla F(\mathbf{w}_t)\|^2 \geq 2m(F(\mathbf{w}_t) - F(\mathbf{w}^*)) \tag{3.90}$$

Notice that according to the result in (3.87) a subsequence of $\|\nabla F(\mathbf{w}_t)\|^2$ converges to null and $\liminf_{t\to\infty} \|\nabla F(\mathbf{w}_t)\|^2 = 0$ almost surely. Observing the relationship in (3.90), we can conclude that a subsequence of the objective value error $F(\mathbf{w}_t) - F(\mathbf{w}^*)$ sequence converges to null which implies

$$\liminf_{t\to\infty} \; F(\mathbf{w}_t) - F(\mathbf{w}^*) = 0. \qquad \text{a.s.} \tag{3.91}$$

Based on the martingale convergence theorem for the sequences α_t and β_t in relation (3.85), the sequence α_t almost surely converges to a limit. Consider the definition of α_t in (3.82) and observe that the sum $\sum_{u=t}^{\infty}(\gamma^u)^2$ is deterministic and its limit is null. Therefore, the limit $\lim_{t\to\infty} F(\mathbf{w}_t) - F(\mathbf{w}^*)$ of the nonnegative objective function errors $F(\mathbf{w}_t) - F(\mathbf{w}^*)$ almost surely exists. This observation in association with the result in (3.92) implies that

the whole sequence of $F(\mathbf{w}_t) - F(\mathbf{w}^*)$ converges almost surely to zero,

$$\lim_{t \to \infty} \; F(\mathbf{w}_t) - F(\mathbf{w}^*) = 0. \qquad \text{a.s.} \tag{3.92}$$

The result in (3.92) holds because the sequence $F(\mathbf{w}_t) - F(\mathbf{w}^*)$ converges almost surely to a limit, while a subsequence of this sequence converges to zero with probability 1 as stated in (3.91). Combining these two observations, the limit that the whole sequence converges to should be 0. To transform the objective function optimality bound in (3.92) into a bound pertaining to the squared distance to optimality $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ simply observe that the lower bound m on the eigenvalues of $\mathbf{H}(\mathbf{w}^*)$ applied to a Taylor's expansion around the optimal argument $\mathbf{w}_t$ implies that

$$F(\mathbf{w}_t) \geq F(\mathbf{w}^*) + \nabla F(\mathbf{w}^*)^T(\mathbf{w}_t - \mathbf{w}^*) + \frac{m}{2}\|\mathbf{w}_t - \mathbf{w}^*\|^2 \tag{3.93}$$

Notice that the optimal point gradient $\nabla F(\mathbf{x}^*)$ is null. This observation and rearranging the terms in (3.93) imply that

$$F(\mathbf{w}_t) - F(\mathbf{w}^*) \geq \frac{m}{2}\|\mathbf{w}_t - \mathbf{w}^*\|^2. \tag{3.94}$$

The upper bound in (3.94) for the squared norm $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ in association with the fact that the sequence $F(\mathbf{w}_t) - F(\mathbf{w}^*)$ almost surely converges to null, leads to the conclusion that the sequence $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ almost surely converges to null. Hence, the claim in (3.80) is valid. ■

Theorem 3 establishes convergence of a subsequence of the oLBFGS algorithm summarized in Algorithm 3. The lower and upper bounds on the eigenvalues of $\hat{\mathbf{B}}_t$ derived in Lemma 6 play a fundamental role in the proofs of the prerequisite Lemma 7 and Theorem 3 proper. Roughly speaking, the lower bound on the eigenvalues of $\hat{\mathbf{B}}_t$ results in an upper bound on the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$ which limits the effect of random variations on the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. If this bound does not exist – as is the case, e.g., of regular stochastic BFGS – we may observe catastrophic amplification of random variations of the stochastic gradient. The upper bound on the eigenvalues of $\hat{\mathbf{B}}_t$, which results in a lower bound on the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$, guarantees that the random variations in the curvature estimate $\hat{\mathbf{B}}_t$ do not yield matrices with arbitrarily small norm. If this bound does not hold, it is possible to end up halting progress before convergence as the stochastic gradient is nullified by multiplication with an arbitrarily small eigenvalue.

The result in Theorem 3 is strong because it holds almost surely over realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$ but not stronger than the same convergence guarantees that hold for SGD. We complement the convergence result in Theorem 3 with a characterization of

the expected convergence rate that we introduce in the following theorem.

Theorem 4 *Consider the online Limited memory BFGS algorithm as defined by the descent iteration in* (3.10) *with matrices* $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ *obtained after* τ *recursive applications of the update in* (3.13) *initialized with* $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t \mathbf{I}$ *and* $\hat{\gamma}_t$ *as given by* (3.26)*. Let Assumptions 4 and 5 hold, and further assume that the stepsize sequence is of the form* $\epsilon_t = \epsilon_0/(t + T_0)$ *with the parameters* ϵ_0 *and* T_0 *satisfying the inequality* $2m\epsilon_0 T_0/C > 1$*. Then, the difference between the expected optimal objective* $\mathbb{E}\left[F(\mathbf{w}_t)\right]$ *and the optimal objective* $F(\mathbf{w}^*)$ *is bounded as*

$$\mathbb{E}\left[F(\mathbf{w}_t)\right] - F(\mathbf{w}^*) \;\leq\; \frac{C_0}{T_0 + t}\;, \tag{3.95}$$

where the constant C_0 *is defined as*

$$C_0 := \max\left\{\frac{\epsilon_0^2\; T_0^2 CMS^2}{2c^2(2m\epsilon_0 T_0 - C)}\;,\; T_0\left(F(\mathbf{w}_0) - \;F(\mathbf{w}^*)\right)\right\}. \tag{3.96}$$

Proof: Consider the result in (3.73) of Lemma 7 and subtract the average function optimal value $F(\mathbf{w}^*)$ from both sides of the inequality to conclude that the sequence of optimality gaps in the RES algorithm satisfies

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\,\middle|\,\mathbf{w}_t\right] - \;F(\mathbf{w}^*) \;\leq\; F(\mathbf{w}_t) - \;F(\mathbf{w}^*) - \frac{\epsilon_t}{C}\|\nabla F(\mathbf{w}_t)\|^2 + \frac{\epsilon_t^2 MS^2}{2c^2}. \tag{3.97}$$

We proceed to find a lower bound for the gradient norm $\|\nabla F(\mathbf{w}_t)\|$ in terms of the error of the objective value $F(\mathbf{w}_t) - \;F(\mathbf{w}^*)$ – this is a standard derivation which we include for completeness, see, e.g., [20]. As it follows from Assumption 1 the eigenvalues of the Hessian $\mathbf{H}(\mathbf{w}_t)$ are bounded between $0 < m$ and $M < \infty$ as stated in (3.32). Taking a Taylor's expansion of the objective function $F(\mathbf{y})$ around $\mathbf{w}$ and using the lower bound in the Hessian eigenvalues we can write

$$F(\mathbf{y}) \;\geq\; F(\mathbf{w}) + \nabla F(\mathbf{w})^T(\mathbf{y} - \mathbf{w}) + \frac{m}{2}\|\mathbf{y} - \mathbf{w}\|^2. \tag{3.98}$$

For fixed $\mathbf{w}$, the right hand side of (3.98) is a quadratic function of $\mathbf{y}$ whose minimum argument we can find by setting its gradient to zero. Doing this yields the minimizing argument $\hat{\mathbf{y}} = \mathbf{w} - (1/m)\nabla F(\mathbf{w})$ implying that for all $\mathbf{y}$ we must have

$$\begin{aligned} F(\mathbf{y}) \;&\geq\; F(\mathbf{w}) + \nabla F(\mathbf{w})^T(\hat{\mathbf{y}} - \mathbf{w}) + \frac{m}{2}\|\hat{\mathbf{y}} - \mathbf{w}\|^2 \\ &=\; F(\mathbf{w}) - \frac{1}{2m}\|\nabla F(\mathbf{w})\|^2. \end{aligned} \tag{3.99}$$

The bound in (3.99) is true for all $\mathbf{w}$ and $\mathbf{y}$. In particular, for $\mathbf{y} = \mathbf{w}^*$ and $\mathbf{w} = \mathbf{w}_t$ (3.99)

yields

$$F(\mathbf{w}^*) \;\geq\; F(\mathbf{w}_t) - \frac{1}{2m}\|\nabla F(\mathbf{w}_t)\|^2. \tag{3.100}$$

Rearrange terms in (B.8) to obtain a bound on the gradient norm squared $\|\nabla F(\mathbf{w}_t)\|^2$. Further substitute the result in (3.97) and regroup terms to obtain the bound

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] - \; F(\mathbf{w}^*) \;\leq\; \left(1 - \frac{2m\epsilon_t}{C}\right)\left(F(\mathbf{w}_t) - \; F(\mathbf{w}^*)\right) + \frac{\epsilon_t^2 M S^2}{2c^2}. \tag{3.101}$$

Take now expected values on both sides of (3.101). The resulting double expectation in the left hand side simplifies to $\mathbb{E}\left[\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right]\right] = \mathbb{E}\left[F(\mathbf{w}_{t+1})\right]$, which allow us to conclude that (3.101) implies that

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\right] - \; F(\mathbf{w}^*) \;\leq\; \left(1 - \frac{2m\epsilon_t}{C}\right)\left(\mathbb{E}\left[F(\mathbf{w}_t)\right] - \; F(\mathbf{w}^*)\right) + \frac{\epsilon_t^2 M S^2}{2c^2}. \tag{3.102}$$

Furhter substituting $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$, which is the assumed form of the step size sequence by hypothesis, we can rewrite (3.102) as

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\right] - \; F(\mathbf{w}^*) \;\leq\; \left(1 - \frac{2m\epsilon_0 T_0}{(T_0 + t)C}\right)\left(\mathbb{E}\left[F(\mathbf{w}_t)\right] - F(\mathbf{w}^*)\right) + \left(\frac{\epsilon_0 T_0}{T_0 + t}\right)^2 \frac{M S^2}{2c^2}. \tag{3.103}$$

Given that the product $2m\epsilon_0 T_0/C > 1$ as per the hypothesis, the sequence $\mathbb{E}\left[F(\mathbf{w}_{t+1})\right] - F(\mathbf{w}^*)$ satisfies the hypotheses of Lemma 3 in Chapter 2 with the variables $a = 2m\epsilon_0 T_0/C$, $b = \epsilon_0^2 T_0^2 M S^2/2c^2$. It then follows from (2.65) and (2.66) that (3.95) is true for the C_0 constant defined in (3.96) upon identifying u_t with $\mathbb{E}\left[F(\mathbf{x}_{t+1})\right] - \; F(\mathbf{x}^*)$, C_0 with Q, and substituting $c = 2m\epsilon_0 T_0/C$, $b = \epsilon_0^2 T_0^2 M S^2/2c^2$ and $t_0 = T_0$ for their explicit values. ■

Theorem 4 shows that under specified assumptions the expected error in terms of the objective value after t oLBFGS iterations is of order $O(1/t)$. As is the case of Theorem 3, this result is not better than the convergence rate of conventional SGD. As can be seen in the proof of Theorem 4, the convergence rate is dominated by the noise term introduced by the difference between stochastic and regular gradients. This noise term would be present even if exact Hessians were available and in that sense the best that can be proven of oLBFGS is that the convergence rate is not worse than that of SGD. Given that theorems 3 and 4 parallel the theoretical guarantees of SGD it is perhaps fairer to describe oLBFGS as an adaptive reconditioning strategy instead of a stochastic quasi-Newton method. The latter description refers to the genesis of the algorithm, but the former is more accurate description of its behavior. Do notice that while the convergence rate doesn't change, improvements in convergence time are significant as we illustrate with the numerical experiments that we

present in the next two sections.

3.4 Support vector machines

Given a training set with points whose classes are known the goal of an SVM is to find a hyperplane that best separates the training set. Let $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be a training set containing N pairs of the form $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^p$ is a feature vector and $y_i \in \{-1, 1\}$ is the corresponding class. The goal is to find a hyperplane supported by a vector $\mathbf{w} \in \mathbb{R}^p$ which separates the training set so that $\mathbf{w}^T\mathbf{x}_i > 0$ for all points with $y_i = 1$ and $\mathbf{w}^T\mathbf{x}_i < 0$ for all points with $y_i = -1$. A loss function $l((\mathbf{x}, y); \mathbf{w})$ defines a measure of distance between the point $\mathbf{x}_i$ and the hyperplane supported by $\mathbf{w}$. We then select the hyperplane supporting vector as

$$\mathbf{w}^* := \underset{\mathbf{w}}{\operatorname{argmin}} \ \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} l((\mathbf{x}_i, y_i); \mathbf{w}), \tag{3.104}$$

where we have also added the regularization term $\lambda\|\mathbf{w}\|^2/2$ for some constant $\lambda > 0$. Common selections for the loss function are the hinge loss $l((\mathbf{x}, y); \mathbf{w}) = \max(0, 1 - y(\mathbf{w}^T\mathbf{x}))$ and the squared hinge loss $l((\mathbf{x}, y); \mathbf{w}) = \max(0, 1 - y(\mathbf{w}^T\mathbf{x}))^2$. See, e.g., [15]. To model (3.104) as a problem in the form of (3.1), define $\boldsymbol{\theta}_i = (\mathbf{x}_i, y_i)$ as a given training point and the probability distribution of θ as uniform on the training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N = \{\boldsymbol{\theta}_i\}_{i=1}^N$. It then suffices to define

$$f(\mathbf{w}, \boldsymbol{\theta}) = \ f(\mathbf{w}, (\mathbf{x}, y)) \ := \ \frac{\lambda}{2}\|\mathbf{w}\|^2 + l((\mathbf{x}, y); \mathbf{w}), \tag{3.105}$$

as sample functions to see that the objective in (3.104) can be written as the average $F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ as in (3.1). We can then use SGD, oBFGS, RES, and oLBFGS to find the optimal classifier $\mathbf{w}^*$. Recall that SGD utilizes stochastic gradients as descent directions and can be formally defined by making $\hat{\mathbf{B}}_t^{-1} = \mathbf{I}$ in (3.10). The (deterministic) BFGS algorithm utilizes recursions of the form in (3.3) with the Hessian approximation matrix computed as in (3.5) with the gradient and variable variations as defined in (3.4). The oBFGS algorithm utilizes stochastic gradients in lieu of stochastic gradients in both, the descent iteration in (3.3) and the gradient variation computation in (3.4); see [105] for details. RES differs in that it introduces a regularization of (3.3) to yield an algorithm that retains its convergence advantages while improving theoretical convergence guarantees and numerical behavior; see Chapter 2 for details. Finally, oLBFGS is defined be algorithms 2 and 3.

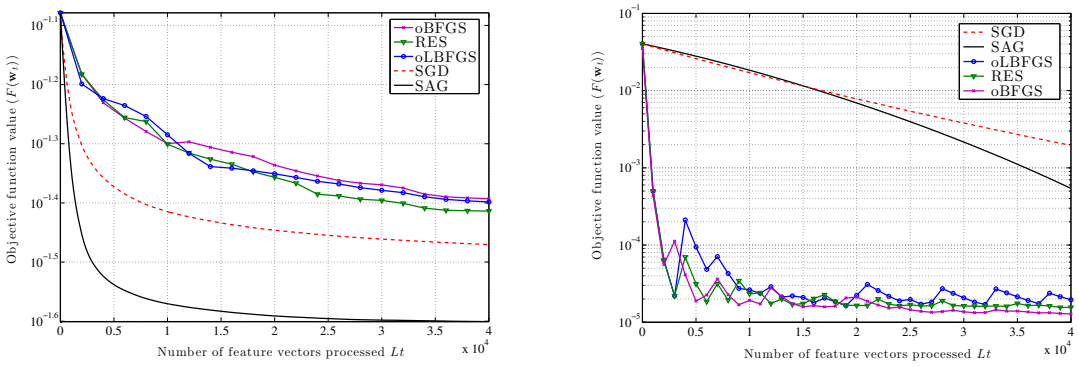
There are also several algorithms that accelerate SGD through the use of memory. These algorithms reduce execution times because they reduce randomness, not because they

improve curvature, but are nonetheless alternatives to oLBFGS that work well for problems with small condition number. We therefore add Stochastic Average Gradient (SAG) to the comparison set as a representative of this class. SAG is a variant of SGD that uses an average of stochastic gradients as a descent direction where only one of the elements of the average is updated per iteration; see [104] for details. For these five algorithms – SGD, oBFGS, RES, oLBFGS, and SAG – we compare achieved objective values with respect to the number of feature vectors processed (Section 3.4.1) as well as with respect to processing times (Section 3.4.2).

3.4.1 Convergence versus number of feature vectors processed

For numerical tests we use the squared hinge loss $l((\mathbf{x}, y); \mathbf{w}) = \max(0, 1 - y(\mathbf{x}^T\mathbf{w}))^2$ in (3.104). The training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ contains $N = 10^4$ feature vectors, half of which belong to the class $y_i = -1$ with the other half belonging to the class $y_i = 1$. For the class $y_i = -1$ each of the p components of each of the feature vectors $\mathbf{x}_i \in \mathbb{R}^p$ is chosen uniformly at random from the interval $[-0.8, 0.2]$. Likewise, each of the p components of each of the feature vectors $\mathbf{x}_i \in \mathbb{R}^p$ is chosen uniformly at random from the interval $[-0.2, 0.8]$ for the class $y_i = 1$. In all of our numerical experiments the parameter λ in (3.104) is set to $\lambda = 10^{-4}$. In order to study the advantages of oLBFGS we consider two different cases where the dimensions of the feature vectors p are different. The size of memory for oLBFGS is set to $\tau = 10$ in all cases. For SGD and SAG the sample size in (3.9) is $L = 1$ and for RES, oBFGS and oLBFGS is $L = 5$. In all tests, the number of feature vectors processed is represented by the product Lt between the iteration index and the sample size used to compute stochastic gradients. This is done because the sample sizes are different. For SGD, oBFGS, RES, and oLBFGS we use a decreasing stepsize sequence of the form $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$, while for the SAG algorithm the stepsize is a constant $\epsilon_t = \epsilon$. This is done because the first four algorithms require diminishing stepsizes to converge to the optimal argument [81], while SAG achieves exact convergence with properly chosen constant stepsize [104]. We report results for $\epsilon_0 = 2 \times 10^{-2}$ and $T_0 = 10^2$ for RES, oLBFGS and oBFGS, which are the values that yield best average performance after processing 4×10^4 feature vectors. Further improvements can be obtained by tuning stepsize parameters individually for each individual algorithm and feature dimension p. Since these improvements are minor we report common parameters for easier reproducibility. For SGD and SAG, whose performance is more variable, we tune the various parameters individually for each dimension p and report results for the combination that yields best average performance after processing 4×10^4 feature vectors.

Figure 3.1 illustrates sample convergence paths for SGD, SAG, oLBFGS, RES, and oBFGS. The figures are shown for $Lt = 4 \leq 10^4$, which is equivalent to 4 passes over the

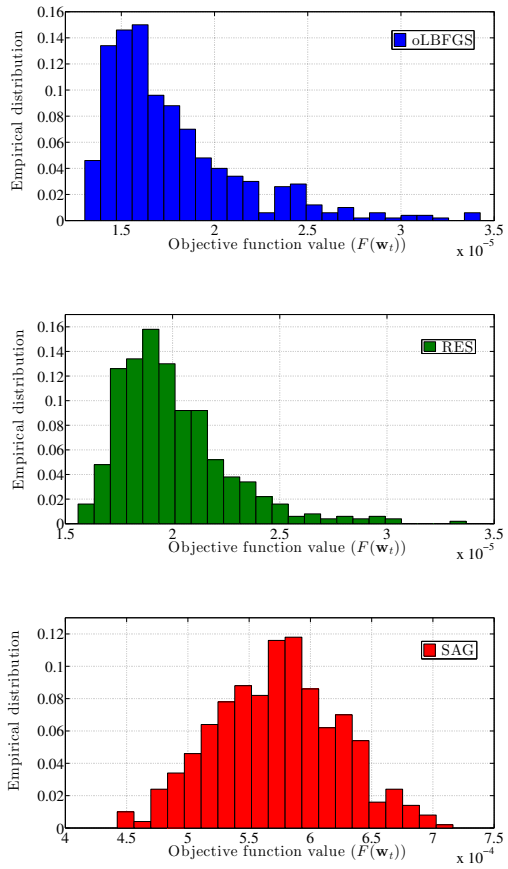


(a) (b)

Figure 3.1: Illustrations of objective function values $F(\mathbf{w}_t)$ for SGD, SAG, oBFGS, RES, and oLBFGS after processing $Lt = 4 \times 10^4$ feature vectors for the cases $p = 5$ (left) and $p = 10^2$ (right). For the case that $p = 5$, SAG and SGD outperform oLBFGS, RES, and oBFGS since the condition number is small. By increasing the dimension of feature vectors to $p = 10^2$, the condition number of the problem increases and stochastic quasi-Newton methods reach a smaller objective function relative to SGD and SAG after processing the same number of feature vectors.

dataset. Results for feature vector dimension $p = 5$ and $p = 10^2$ are shown in subfigures (a) and (b), respectively. When the feature vector dimension p is small, the condition number of the optimization argument is not large. In this case SGD and its accelerated version SAG converge to the optimal argument faster than any of the stochastic quasi-Newton methods as can be seen in Figure 3.1(a). E.g., at the end of the period shown when we have processed $Lt = 4 \times 10^4$ the objective function values for SGD and SAG are 3.4×10^{-2} and 2.5×10^{-2}, while the values of objective function for oLBFGS, RES, and oBFGS are $(3.9 \pm 0.1) \times 10^{-2}$, 3.8×10^{-2}. For these well conditioned problems SAG is the method of choice. The situation is reversed when the feature vector dimension is $p = 10^2$ as the quasi-Newton methods oBFGS, RES, and oLBFGS do better than SGD and SAG. According to Figure 3.1(b) the objective function values for oLBFGS, oBFGS, and RES are 1.9×10^{-5}, 1.3×10^{-5}, and 1.5×10^{-5}, respectively. The objective function values for SGD and SAG after processing the same number of feature vectors $Lt = 4^4$ are 1.8×10^{-3} and 5.4×10^{-4}, respectively. For these ill conditioned problems the quasi-Newton methods are preferable.

To have a more comprehensive comparison of oLBFGS, RES, oBFGS, SGD, and SAG we run these algorithms for $J = 10^3$ realizations to compare the empirical distributions of objective function value for these five algorithms. Figures 3.2 and 3.3 show the empirical distributions of the objective function value $F(\mathbf{w}_t)$ attained after processing $Lt = 4 \times 10^4$ feature vectors using $J = 10^3$ realizations for the cases that $p = 10^2$ and $p = 10^3$, respectively. According to Figure 3.2 the averages of objective value function for oLBFGS, oBFGS and RES are 1.7×10^{-5}, 1.4×10^{-5} and 1.9×10^{-5}, respectively. These numbers show that the performance of oLBFGS is very close to the performances of oBFGS and RES. This

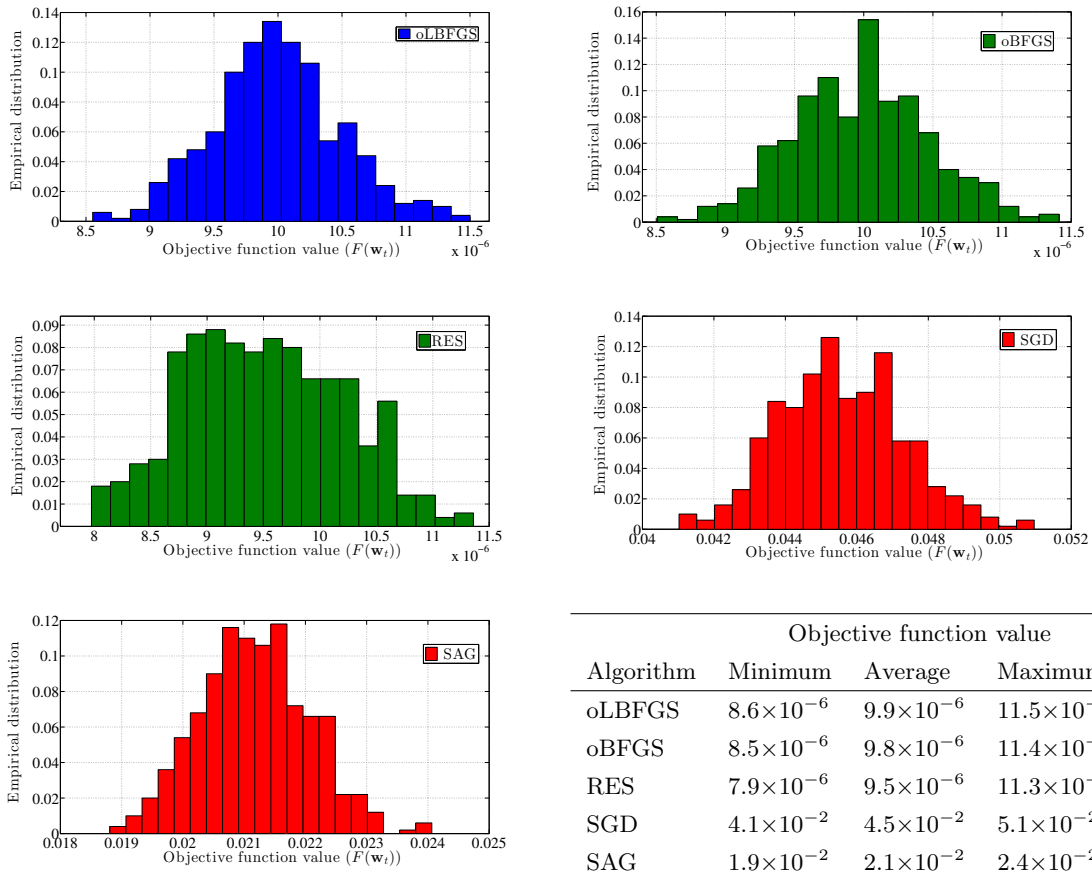


	Objective function value		
Algorithm	Minimum	Average	Maximum
oLBFGS	1.3×10^{-5}	1.7×10^{-5}	3.4×10^{-5}
oBFGS	1.2×10^{-5}	1.4×10^{-5}	2.0×10^{-5}
RES	1.5×10^{-5}	1.9×10^{-5}	3.3×10^{-5}
SGD	1.2×10^{-3}	1.6×10^{-3}	1.9×10^{-3}
SAG	4.4×10^{-4}	5.7×10^{-4}	7.1×10^{-4}

Figure 3.2: Histograms of objective function value $F(\mathbf{w}_t)$ after processing $Lt = 4 \times 10^4$ feature vectors for $p = 10^2$. The values of objective function for oLBFGS, oBFGS and RES are close to each other and smaller than the objective function values for SAG and SGD.

similarity holds despite the fact that oLBFGS uses only the last $\tau = 10$ stochastic gradients to estimate curvature whereas oBFGS and RES utilize all past stochastic gradients to do so. The advantage of oLBFGS is in the smaller computational cost of processing feature vectors as we discuss in Section 3.4.2. The corresponding average objective values achieved by SGD and SAG after processing $Lt = 4\times10^4$ feature vectors are 1.6×10^{-3} and 5.7×10^{-4}, respectively. Both of these are at least an order of magnitude larger than the average objective value achieved by oLBFGS – or RES and oBFGS for that matter.

Figure 3.3 repeats the study in Figure 3.2 for the case in which the feature vector dimension is increased to $p = 10^3$. The performance of oLBGS is still about the same as the performances of oBFGS and RES. The average objective function values achieved after processing $Lt = 4 \times 10^4$ feature vectors are 9.9×10^{-6}, 9.8×10^{-6} and 9.5×10^{-6} for oLBFGS, oBFGS and RES, respectively. The relative performance with respect to SGD and SAG, however, is now larger. The averages of objective function values for SAG



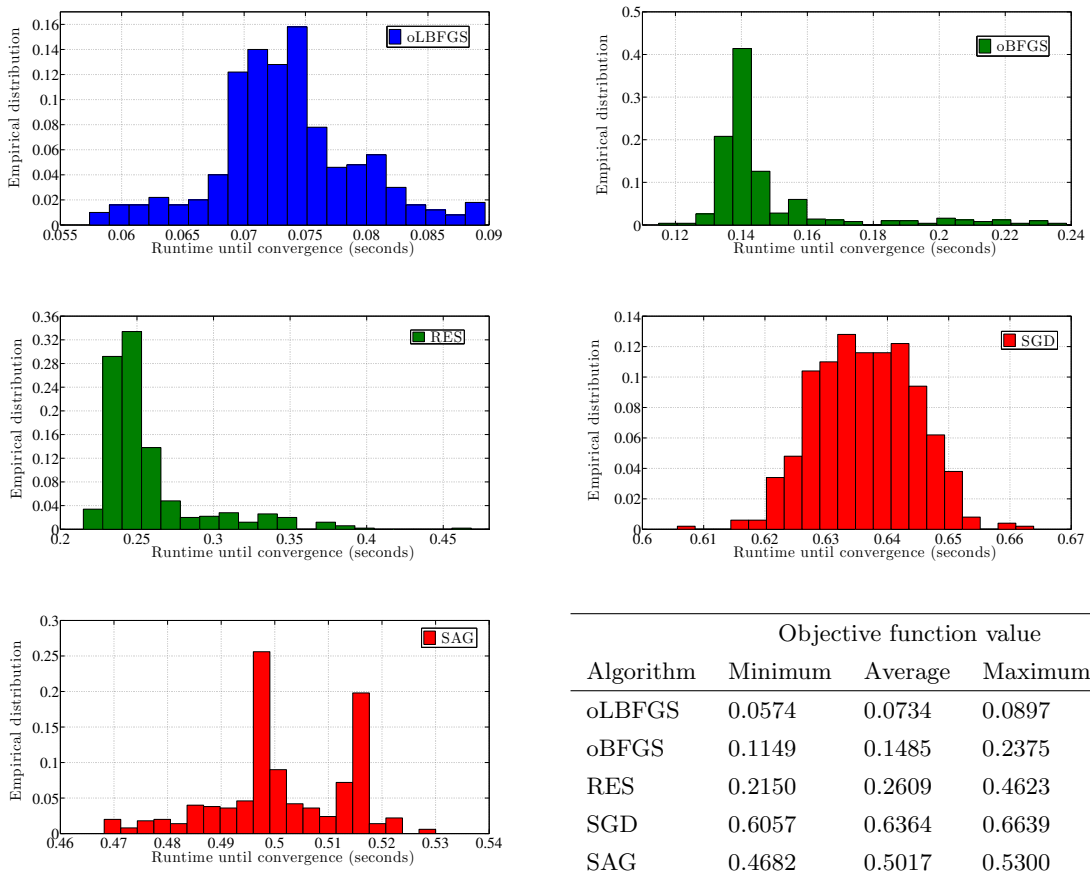
	Objective function value		
Algorithm	Minimum	Average	Maximum
oLBFGS	8.6×10^{-6}	9.9×10^{-6}	11.5×10^{-6}
oBFGS	8.5×10^{-6}	9.8×10^{-6}	11.4×10^{-6}
RES	7.9×10^{-6}	9.5×10^{-6}	11.3×10^{-6}
SGD	4.1×10^{-2}	4.5×10^{-2}	5.1×10^{-2}
SAG	1.9×10^{-2}	2.1×10^{-2}	2.4×10^{-2}

Figure 3.3: Histograms of objective function value $F(\mathbf{w}_t)$ after processing $Lt = 4 \times 10^4$ feature vectors for $p = 10^3$. The values of objective function for oBFGS, oLBFGS and RES are close to each other and smaller than the objective function values for SAG and SGD.

and SGD in this case are 2.1×10^{-2} and 4.5×10^{-2}, respectively. These values are more than 3 orders of magnitude larger than the corresponding values achieved by oLBFGS. This relative improvement can be further increased if we consider problems of even larger dimension. Further observe that oBFGS and RES start to become impractical if we further increase the feature vector dimension since the respective iterations have computational costs of order $O(p^2)$ and $O(p^3)$. We analyze this in detail in the following section.

3.4.2 Convergence versus processing time

The analysis in Section 3.4.1 is relevant for online implementations in which the goal is to make the best possible use of the information provided by each new acquired feature vector. In implementations where computational cost is of dominant interest we have to account for the fact that the respective iteration costs are of order $O(p)$ for SGD and SAG, of order

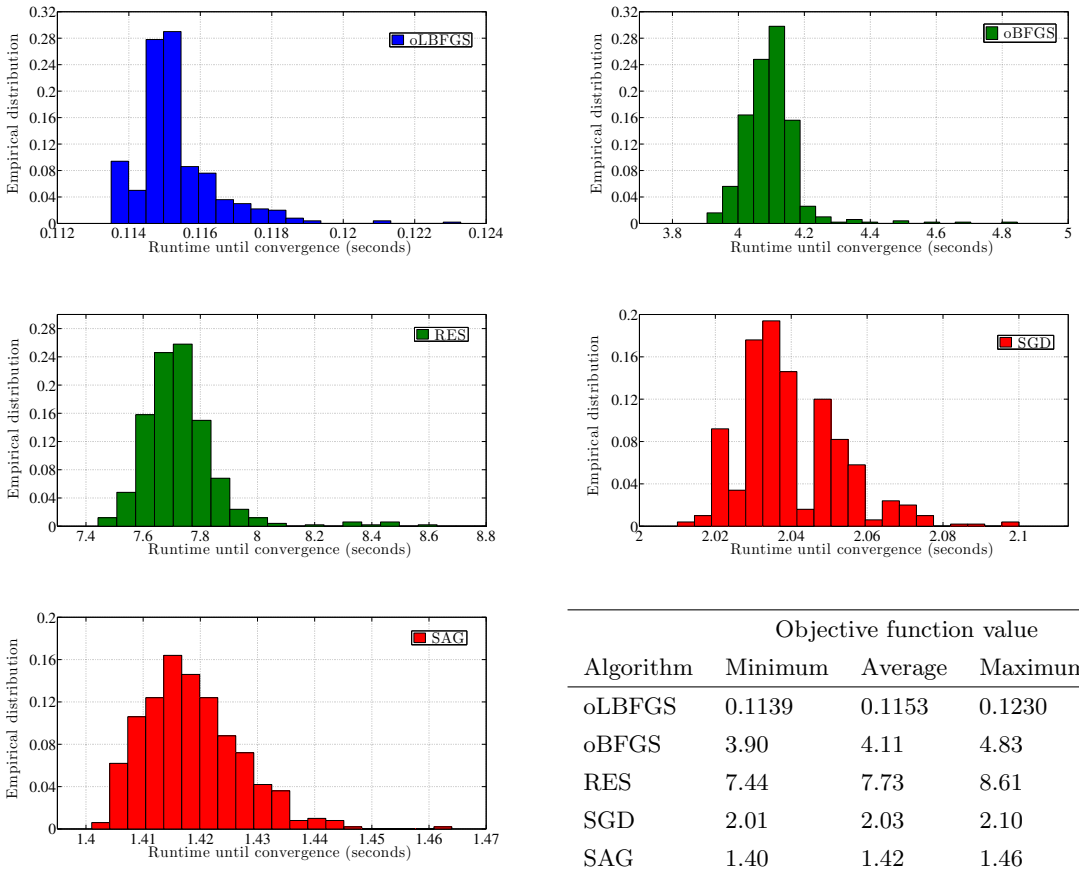


	Objective function value		
Algorithm	Minimum	Average	Maximum
oLBFGS	0.0574	0.0734	0.0897
oBFGS	0.1149	0.1485	0.2375
RES	0.2150	0.2609	0.4623
SGD	0.6057	0.6364	0.6639
SAG	0.4682	0.5017	0.5300

Figure 3.4: Histograms of required CPU runtime for achieving objective function value $F(\mathbf{w}_t) = 10^{-4}$ when $p = 10^2$. The convergence time of oLBFGS is smaller than the required runtimes of oBFGS and RES, while SAG and SGD are slower than all the three quasi-Newton methods.

$O(\tau p)$ for oLBFGS, and of orders $O(p^2)$ and $O(p^3)$ for oBFGS and RES. As we increase the problem dimension we expect the convergence time advantages of oBFGS and RES in terms of number of feature vectors processed to be overwhelmed by the increased computational cost of each iteration. For oLBFGS, on the contrary, we expect the convergence time advantages in terms of number of feature vectors processed to persist in terms of processing time. To demonstrate that this is the case we repeat the experiments in Section 3.4.1 but record the processing time required to achieve a target objective value. The parameters used here are the same parameters of Section 3.4.1.

In Figure 3.4 we consider $p = 10^2$ and record the processing time required to achieve the objective function value $F(\mathbf{w}_t) = 10^{-4}$. Histograms representing empirical distributions of execution times measured in seconds (s) are shown for oLBFGS, oBFGS, RES, SGD, and SAG. We also summarize the average minimum and maximum times observed for each algorithm. The average run times for oBFGS and RES are 0.14 s and 0.26 s which are better



	Objective function value		
Algorithm	Minimum	Average	Maximum
oLBFGS	0.1139	0.1153	0.1230
oBFGS	3.90	4.11	4.83
RES	7.44	7.73	8.61
SGD	2.01	2.03	2.10
SAG	1.40	1.42	1.46

Figure 3.5: Histograms of required CPU runtime for achieving objective function value $F(\mathbf{w}_t) = 10^{-5}$ when $p = 10^3$. SAG and SGD have a faster convergence time in comparison to oBFGS and RES, while oLBFGS is the fastest algorithm among all.

than the average run times of SGD and SAG that stand at 0.63 s and 0.50 s. The advantage, however, is less marked than when measured with respect to the number of feature vector processed. For oLBGS the advantage with respect to SGD and SAG is still close to one order of magnitude since the average convergence time stands at 0.073 s. When measured in computation time oLBGS is also better than RES and oBFGS, as expected.

Figure 3.5 presents the analogous histograms and summary statistics when the feature vector dimension is $p = 10^3$ and the algorithm is run until achieving the objective value $F(\mathbf{w}_t) = 10^{-5}$. For this problem and metric the performances of RES and oBFGS are worse than the corresponding performances of SGD and SAG. The respective average convergence times are 7.7 s and 4.1 s for RES and oBFGS and 1.4 s and 2.0 s for SAG and SGD. The oLBFGS algorithm, however, has an average convergence time of 0.11 s. This is still an order of magnitude faster than the first order methods SAG and SGD – and has an even larger advantage with respect to oBFGS and RES, by extension. The relative reduction of

execution times of oLBGS relative to all other 4 methods becomes more marked for problems of larger dimension. We investigate these advantages on the search engine advertising problem that we introduce in the following section.

3.5 Search engine advertising

We apply oLBFGS to the problem of predicting the click-through rate (CTR) of an advertisement displayed in response to a specific search engine query by a specific visitor. In these problems we are given meta information about an advertisement, the words that appear in the query, as well as some information about the visitor and are asked to predict the likelihood that this particular ad is clicked by this particular user when performing this particular query. The information specific to the ad includes descriptors of different characteristics such as the words that appear in the title, the name of the advertiser, keywords that identify the product, and the position on the page where the ad is to be displayed. The information specific to the user is also heterogeneous and includes gender, age, and propensity to click on ads. To train a classifier we are given information about past queries along with the corresponding click success of the ads displayed in response to the query. The ad metadata along with user data and search words define a feature vector that we use to train a logistic regressor that predicts the CTR of future ads. Given the heterogeneity of the components of the feature vector we expect a logistic cost function with skewed level sets and consequent large benefits from the use of oLBFGS.

3.5.1 Feature vectors

For the CTR problem considered here we use the Tencent search engine data set [116]. This data set contains the outcomes of 236 million (236×10^6) searches along with information about the ad, the query, and the user. The information contained in each sample point is the following:

- User profile: If known, age and gender of visitor performing query.
- Depth: Total number of advertisements displayed in the search results page.
- Position: Position of the advertisement in the search page.
- Impression: Number of times the ad was displayed to the user who issued the query.
- Query: The words that appear in the user's query.
- Title: The words that appear in the title of ad.
- Keywords: Selected keywords that specify the type of product.

- Ad ID: Unique identifier assigned to each specific advertisement.
- Advertiser ID: Unique identifier assigned to each specific advertiser.
- Clicks: Number of times the user clicked on the ad.

From this information we create a set of feature vectors $\{\mathbf{x}_i\}_{i=1}^N$, with corresponding labels $y_i \in \{-1, 1\}$. The label associated with feature vector $\mathbf{x}_i$ is $y_i = 1$ if the number of clicks in the ad is more than 0. Otherwise the label is $y_i = -1$. We use a binary encoding for all the features in the vector $\mathbf{x}_i$. For the age of the user we use the six age intervals $(0, 12]$, $(12, 18]$, $(18, 24]$, $(24, 30]$, $(30, 40]$, and $(40, \infty)$ to construct six indicator entries in $\mathbf{x}_i$ that take the value 1 if the age of the user is known to be in the corresponding interval. E.g., a 21 year old user has an age that falls in the third interval which implies that we make $[\mathbf{x}_i]_3 = 1$ and $[\mathbf{x}_i]_k = 0$ for all other k between 1 and 6. If the age of the user is unknown we make $[\mathbf{x}_i]_k = 0$ for all k between 1 and 6. For the gender of the visitors we use the next three components of $\mathbf{x}_i$ to indicate male, female, or unknown gender. For a male user we make $[\mathbf{x}_i]_7 = 1$, for a female user $[\mathbf{x}_i]_8 = 1$, and for visitors of unknown gender we make $[\mathbf{x}_i]_9 = 1$. The next three components of $\mathbf{x}_i$ are used for the depth feature. If the the number of advertisements displayed in the search page is 1 we make $[\mathbf{x}_i]_{10} = 1$, if 2 different ads are shown we make $[\mathbf{x}_i]_{11} = 1$, and for depths of 3 or more we make $[\mathbf{x}_i]_{12} = 1$. To indicate the position of the ad in the search page we also use three components of $\mathbf{x}_i$. We use $[\mathbf{x}_i]_{13} = 1$, $[\mathbf{x}_i]_{14} = 1$, and $[\mathbf{x}_i]_{15} = 1$ to indicate that the ad is displayed in the first, second, and third position, respectively. Likewise we use $[\mathbf{x}_i]_{16}$, $[\mathbf{x}_i]_{17}$ and $[\mathbf{x}_i]_{18}$ to indicate that the impression of the ad is 1, 2 or more than 3.

For the words that appear in the query we have in the order of 10^5 distinct words. To reduce the number of elements necessary for this encoding we create 20,000 bags of words through random hashing with each bag containing 5 or 6 distinct words. Each of these bags is assigned an index k. For each of the words in the query we find the bag in which this word appears. If the word appears in the kth bag we indicate this occurrence by setting the $k + 18$th component of the feature vector to $[\mathbf{x}_i]_{k+18} = 1$. Observe that since we use 20,000 bags, components 19 through 20,018 of $\mathbf{x}_i$ indicate the presence of specific words in the query. Further note that we may have more than one $\mathbf{x}_i$ component different from zero because there may be many words in the query, but that the total number of nonzero elements is much smaller than 20,000. On average, 3.0 of these elements of the feature vector are nonzero. The same bags of words are used to encode the words that appear in the title of the ad and the product keywords. We encode the words that appear in the title of the ad by using the next $20,000$ components of vector $\mathbf{x}_i$, i.e. components $20,019$ through $40,018$. Components $40,019$ through $60,018$ are used to encode product keywords. As in the case of the words in the search just a few of these components are nonzero. On average,

Table 3.1: Components of the feature vectors for prediction of advertisements click-through rates. For each feature class we report the total number of components in the feature vector as well as the maximum and average number of nonzero components.

		Nonzero components	
Feature type	**Total components**	**Max. (observed/structure)**	**Mean (observed)**
Age	6	1 (structure)	1.0
Gender	3	1 (structure)	1.0
Impression	3	1 (structure)	1.0
Depth	3	1 (structure)	1.0
Position	3	1 (structure)	1.0
Query	20,000	125 (observed)	3.0
Title	20,000	29 (observed)	8.8
Keyword	20,000	16 (observed)	2.1
Advertiser ID	5,184	1 (structure)	1.0
Advertisement ID	108,824	1 (structure)	1.0
Total	**174,026**	**148 (observed)**	**20.9**

the number of non-zero components of feature vectors that describe the title features is 8.8. For product keywords the average is 2.1. Since the number of distinct advertisers in the training set is $5,184$ we use feature components $60,019$ through 65202 to encode this information. For the kth advertiser ID we set the $k + 60,018th$ component of the feature vector to $[\mathbf{x}_i]_{k+60,018} = 1$. Since the number of distinct advertisements is $108,824$ we allocate the last $108,824$ components of the feature vector to encode the ad ID. Observe that only one out of $5,184$ advertiser ID components and one of the $108,824$ advertisement ID components are nonzero.

In total, the length of the feature vector is 174,026 where each of the components are either 0 or 1. The vector is very sparse. We observe a maximum of 148 nonzero elements and an average of 20.9 nonzero elements in the training set – see Table 3.1. This is important because the cost of implementing inner products in the oLBFGS training of the logistic regressor that we introduce in the following section is proportional to the number of nonzero elements in $\mathbf{x}_i$.

3.5.2 Logistic regression of click-through rate

We use the training set to estimate the CTR with a logistic regression. For that purpose let $\mathbf{x} \in \mathbb{R}^p$ be a vector containing the features described in Section 3.5.1, $\mathbf{w} \in \mathbb{R}^p$ a classifier that we want to train, and $y \in -1, 1$ an indicator variable that takes the value $y = 1$ when the ad presented to the user is clicked and $y = -1$ when the ad is not clicked by the user. We hypothesize that the CTR, defined as the probability of observing $y = 1$, can be written

as the logistic function

$$\text{CTR}(\mathbf{x};\mathbf{w}) \;:=\; \text{P}\left[y=1 \,\middle|\, \mathbf{x};\mathbf{w}\right] \;=\; \frac{1}{1+\exp\left(-\mathbf{x}^T\mathbf{w}\right)}\;. \tag{3.106}$$

We read (3.106) as stating that for a feature vector $\mathbf{x}$ the CTR is determined by the inner product $\mathbf{x}^T\mathbf{w}$ through the given logistic transformation.

Consider now the training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ which contains N realizations of features $\mathbf{x}_i$ and respective click outcomes y_i and further define the sets $\mathcal{S}_1 := \{(\mathbf{x}_i, y_i) \in \mathcal{S} : y_i = 1\}$ and $\mathcal{S}_{-1} := \{(\mathbf{x}_i, y_i) \in \mathcal{S} : y_i = -1\}$ containing clicked and unclicked advertisements, respectively. With the data given in $\mathcal{S}$ we define the optimal classifier $\mathbf{w}^*$ as a maximum likelihood estimate (MLE) of $\mathbf{w}$ given the model in (3.106) and the training set $\mathcal{S}$. This MLE can be found as the minimizer of the log-likelihood loss

$$\begin{aligned}\mathbf{w}^* \;&:=\; \text{argmin}\,\frac{\lambda}{2}\|\mathbf{w}\|^2 \;+\; \frac{1}{N}\sum_{i=1}^{N}\log\Big(1+\exp\big(-y_i\mathbf{x}_i^T\mathbf{w}\big)\Big)\\ &=\; \text{argmin}\,\frac{\lambda}{2}\|\mathbf{w}\|^2 \;+\; \frac{1}{N}\Bigg[\sum_{\mathbf{x}_i\in\mathcal{S}_1}\log\Big(1+\exp(-\mathbf{x}_i^T\mathbf{w})\Big)+\sum_{\mathbf{x}_i\in\mathcal{S}_{-1}}\log\Big(1+\exp(\mathbf{x}_i^T\mathbf{w})\Big)\Bigg],\end{aligned} \tag{3.107}$$

where we have added the regularization term $\lambda\|\mathbf{w}\|^2/2$ to disincentivize large values in the weight vector $\mathbf{w}^*$; see e.g., [86].

The practical use of (3.106) and (3.107) is as follows. We use the data collected in the training set $\mathcal{S}$ to determine the vector $\mathbf{w}^*$ in (3.107). When a user issues a query we concatenate the user and query specific elements of the feature vector with the ad specific elements of several candidate ads. We then proceed to display the advertisement with, say, the largest CTR. We can interpret the set $\mathcal{S}$ as having been acquired offline or online. In the former case we want to use a stochastic optimization algorithm because computing gradients is infeasible – recall that we are considering training samples with a number of elements N in the order of 10^6. The performance metric of interest in this case is the logistic cost as a function of computational time. If elements of $\mathcal{S}$ are acquired online we update $\mathbf{w}$ whenever a new vector becomes available so as to adapt to changes in preferences. In this case we want to exploit the information in new samples as much as possible. The correct metric in this case is the logistic cost as a function of the number of feature vectors processed. We use the latter metric for the numerical experiments in the following section.

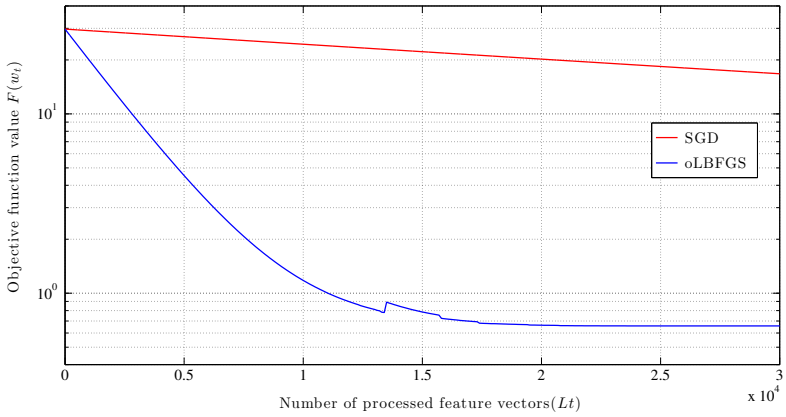


Figure 3.6: Illustration of Negative log-likelihood value for oLBFGS and SGD after processing certain amount of feature vectors. The accuracy of oLBFGS is better than SGD after processing a specific number of feature vectors.

3.5.3 Numerical results

Out of the 236×10^6 in the Tencent dataset we select 10^6 sample points to use as the training set $\mathcal{S}$ and 10^5 sample points to use as a test set $\mathcal{T}$. To select elements of the training and test set we divide the first 1.1×10^6 sample points of the complete dataset in 10^5 consecutive blocks with 11 elements. The first 10 elements of the block are assigned to the training set and the 11th element to the test set. To solve for the optimal classifier we implement SGD and oLBFGS by selecting feature vectors $\mathbf{x}_i$ at random from the training set $\mathcal{S}$. In all of our numerical experiments the regularization parameter in (3.107) is $\lambda = 10^{-6}$. The stepsizes for both algorithms are of the form $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$. We set $\epsilon_0 = 10^{-2}$ and $T_0 = 10^4$ for oLBFGS and $\epsilon_0 = 10^{-1}$ and $T_0 = 10^6$ for SGD. For SGD the sample size in (3.9) is set to $L = 20$ whereas for oLBFGS it is set to $L = 100$. The values of parameters ϵ_0, T_0, and L are chosen to yield best convergence times in a rough parameter optimization search. Observe the relatively large values of L that are used to compute stochastic gradients. This is necessary due to the extreme sparsity of the feature vectors $\mathbf{x}_i$ that contain an average of only 20.9 nonzero out 174,026 elements. Even when considering $L = 100$ vectors they are close to orthogonal. The size of memory for oLBFGS is set to $\tau = 10$. With $L = 100$ features with an average sparsity of 20.9 nonzero elements and memory $\tau = 10$ the cost of each LBGS iteration is in the order of 2.1×10^4 operations.

Figure 3.6 illustrates the convergence path of SGD and oLBFGS on the advertising training set. We depict the value of the log likelihood objective in (3.107) evaluated at $\mathbf{w} = \mathbf{w}_t$ where $\mathbf{w}_t$ is the classifier iterate determined by SGD or oLBFGS. The horizontal axis is scaled by the number of feature vectors L that are used in the evaluation of stochastic gradients. This results in a plot of log likelihood cost versus the number Lt of feature

vectors processed. To read iteration indexes from Figure 3.6 divide the horizontal axis values by $L = 100$ for oLBGS and $L = 20$ for SGD. Consistent with the synthetic data results in Section 3.4, the curvature correction of oLBFGS results in significant reductions in convergence time. For way of illustration observe that after processing $Lt = 3 \times 10^4$ feature vectors the objective value achieved by oLBFGS is $F(\mathbf{w}_t) = 0.65$, while for SGD it still stands at $F(\mathbf{w}_t) = 16$ which is a meager reduction from the random initialization point at which $F(\mathbf{w}_0) = 30$. In fact, oLBFGS converges to the minimum possible log likelihood cost $F(\mathbf{w}_t) = 0.65$ after processing 1.7×10^4 feature vectors. This illustration hints that oLBGS makes better use of the information available in feature vectors.

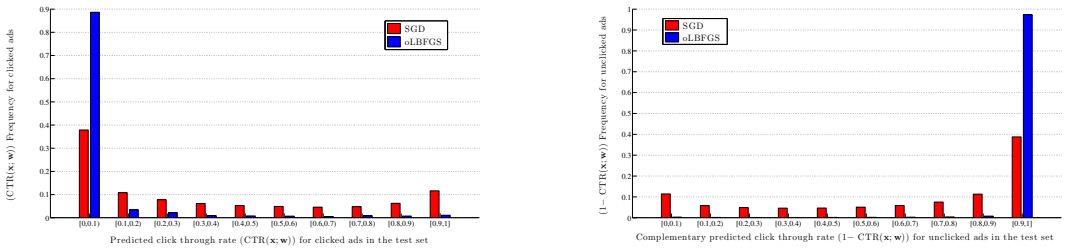
To corroborate that the advantage of oLBGS is not just an artifact of the structure of the log likelihood cost in (3.107) we process 2×10^4 feature vectors with SGD and oLBFGS and evaluate the predictive accuracy of the respective classifiers on the test set. As measures of predictive accuracy we adopt the frequency histogram of the predicted click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$ for all clicked ads and the frequency histogram of the complementary predicted click through rate $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ for all the ads that were *not* clicked. To do so we separate the test set by defining the set $\mathcal{T}_1 := \{(\mathbf{x}_i, y_i) \in \mathcal{T} : y_i = 1\}$ of clicked ads and the set $\mathcal{T}_{-1} := \{(\mathbf{x}_i, y_i) \in \mathcal{T} : y_i = -1\}$ of ads in the test set that were not clicked. For a given classifier $\mathbf{w}$ we compute the predicted probability $\text{CTR}(\mathbf{x}_i; \mathbf{w})$ for each of the ads in the clicked set $\mathcal{T}_1$. We then consider a given interval $[a, b]$ and define the frequency histogram of the predicted click through rate as the fraction of clicked ads for which the prediction $\text{CTR}(\mathbf{x}_i; \mathbf{w})$ falls in $[a, b]$,

$$\mathcal{H}_1(\mathbf{w}; a, b) := \frac{1}{\#(\mathcal{T}_1)} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}_1} \mathbb{I}\Big\{\text{CTR}(\mathbf{x}_i; \mathbf{w}) \in [a, b]\Big\}, \tag{3.108}$$

where $\#(\mathcal{T}_1)$ denotes the cardinality of the set $\mathcal{T}_1$. Likewise, we consider the ads in the set $\mathcal{T}_{-1}$ that were not clicked and compute the prediction $1 - \text{CTR}(\mathbf{x}_i; \mathbf{w})$ on the probability of the ad not being clicked. We then consider a given interval $[a, b]$ and define the frequency histogram $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ as the fraction of unclicked ads for which the prediction $1 - \text{CTR}(\mathbf{x}_i; \mathbf{w})$ falls in $[a, b]$,

$$\mathcal{H}_{-1}(\mathbf{w}; a, b) := \frac{1}{\#(\mathcal{T}_{-1})} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}_{-1}} \mathbb{I}\Big\{1 - \text{CTR}(\mathbf{x}_i; \mathbf{w}) \in [a, b]\Big\}. \tag{3.109}$$

The histogram $\mathcal{H}_1(\mathbf{w}; a, b)$ in (3.108) allows us to study how large the predicted probability $\text{CTR}(\mathbf{x}_i; \mathbf{w})$ is for the clicked ads. Conversely, the histogram $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ in (3.109) gives an indication of how large the predicted probability $1 - \text{CTR}(\mathbf{x}_i; \mathbf{w})$ is for the unclicked ads. An ideal classifier is one for which the frequency counts in $\mathcal{H}_1(\mathbf{w}; a, b)$ accumulate at $\text{CTR}(\mathbf{x}_i; \mathbf{w}) = 1$ and for which $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ accumulates observations at $1 - \text{CTR}(\mathbf{x}_i; \mathbf{w}) = 1$.



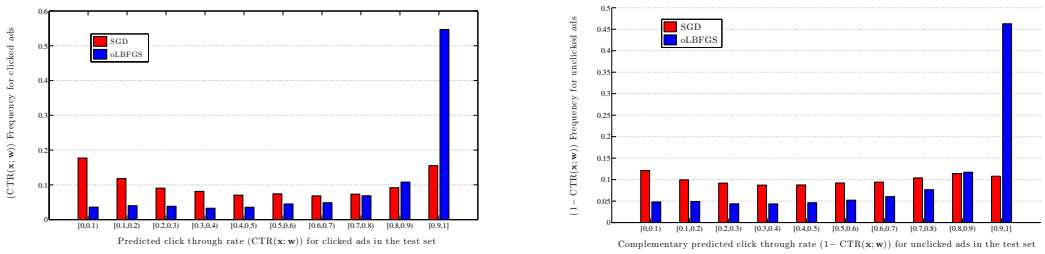
(a) (b)

Figure 3.7: Performance of classifier after processing 2×10^4 feature vectors with SGD and oLBFGS for the cost in (3.107). Histograms for: (a) predicted click through rate $\mathrm{CTR}(\mathbf{x}; \mathbf{w})$ for all clicked ads; and (b) complementary predicted click through rate $1 - \mathrm{CTR}(\mathbf{x}; \mathbf{w})$ for all unclicked ads. For an ideal classifier that predicts a click probability $\mathrm{CTR}(\mathbf{x}; \mathbf{w}) = 1$ for all clicked ads and a click probability $\mathrm{CTR}(\mathbf{x}; \mathbf{w}) = 0$ for all unclicked ads the frequency counts in $\mathcal{H}_1(\mathbf{w}; a, b)$ and $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ would accumulate in the $[0.9, 1]$ bin. Neither SGD nor oLBFGS compute acceptable classifiers because the number of clicked ads in the test set is very small and predicting $\mathrm{CTR}(\mathbf{x}; \mathbf{w}) = 0$ for all ads is close to the minimum of (3.107).

This corresponds to a classifier that predicts a click probability of 1 for all ads that were clicked and a click probability of 0 for all ads that were not clicked.

Fig. 3.7(a) shows the histograms of predicted click through rate $\mathrm{CTR}(\mathbf{x}; \mathbf{w})$ for all clicked ads by oLBFGS and SGD classifiers after processing 2×10^4 training sample points. oLBFGS classifier for 88% of test points in $\mathcal{T}_1$ predicts $\mathrm{CTR}(\mathbf{x}; \mathbf{w})$ in the interval $[0, 0.1]$ and the classifier computed by SGD estimates the click through rate $\mathrm{CTR}(\mathbf{x}; \mathbf{w})$ in the same interval for 37% of clicked ads in the test set. These numbers shows the inaccurate click through rate predictions of both classifiers for the test points with label $y = 1$. Although, SGD and oLBFGS classifiers have catastrophic performances in predicting click through rate $\mathrm{CTR}(\mathbf{x}; \mathbf{w})$ for the clicked ads in the test set, they perform well in estimating complementary predicted click through rate $1 - \mathrm{CTR}(\mathbf{x}; \mathbf{w})$ for the test points with label $y = -1$. This observation implied by Fig. 3.7(b) which shows the histograms of complementary predicted click through rate $1 - \mathrm{CTR}(\mathbf{x}; \mathbf{w})$ for all *not* clicked ads by oLBFGS and SGD classifiers after processing 2×10^4 training sample points. As it shows after processing 2×10^4 sample points of the training set the predicted probability $1 - \mathrm{CTR}(\mathbf{x}; \mathbf{w})$ by the SGD classifier for 38.8% of the test points are in the interval $[0.9, 1]$, while for the classifier computed by oLBFGS 97.3% of predicted probability $1 - \mathrm{CTR}(\mathbf{x}; \mathbf{w})$ are in the interval $[0.9, 1]$ which is a significant performance.

The reason for the inaccurate predictions of both classifiers is that most elements in the training set $\mathcal{S}$ are unclicked ads. Thus, the minimizer $\mathbf{w}^*$ of the log likelihood cost in (3.107) is close to a classifier that predicts $\mathrm{CTR}(\mathbf{x}; \mathbf{w}^*) \approx 0$ for most ads. Indeed, out of the



(a) (b)

Figure 3.8: Performance of classifier after processing 2×10^4 feature vectors with SGD and oLBFGS for the cost in (3.110). Histograms for: (a) predicted click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$ for all clicked ads; and (b) complementary predicted click through rate $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ for all unclicked ads. For an ideal classifier that predicts a click probability $\text{CTR}(\mathbf{x}; \mathbf{w}) = 1$ for all clicked ads and a click probability $\text{CTR}(\mathbf{x}; \mathbf{w}) = 0$ for all unclicked ads the frequency counts in $\mathcal{H}_1(\mathbf{w}; a, b)$ and $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ would accumulate in the $[0.9, 1]$ bin. The classifier computed by oLBFGS is much more accurate than the one computed by SGD.

10^6 elements in the training set, 94.8% of them have labels $y_i = -1$ and only the remaining 5.2×10^4 feature vectors correspond to clicked ads. To overcome this problem we replicate observations with labels $y_i = 1$ to balance the representation of both labels in the training set. Equivalently, we introduce a constant γ and redefine the log likelihood objective in (3.107) to give a larger weight to feature vectors that correspond to clicked ads,

$$\mathbf{w}^* \;=\; \text{argmin}\, \frac{\lambda}{2}\|\mathbf{w}\|^2 \;+\; \frac{1}{M}\Bigg[\, \gamma \sum_{\mathbf{x}_i \in \mathcal{S}_1} \log\Big(1 + \exp(-\mathbf{x}_i^T \mathbf{w})\Big) + \sum_{\mathbf{x}_i \in \mathcal{S}_{-1}} \log\Big(1 + \exp(\mathbf{x}_i^T \mathbf{w})\Big)\Bigg], \tag{3.110}$$

where we defined $M := \gamma \#(\mathcal{S}_1) + \#(\mathcal{S}_{-1})$ to account for the replication of clicked featured vectors that is implicit in (3.110). To implement SGD and oLBFGS in the weighted log function in (3.110) we need to bias the random choice of feature vector so that vectors in $\mathcal{S}_1$ are γ times more likely to be selected than vectors in $\mathcal{S}_2$. Although our justification to introduce γ is to balance the types of feature vectors, γ is just a tradeoff constant to increase the percentage of correct predictions for clicked ads – which is close to zero in Figure 3.7 – at the cost of reducing the accuracy of correct predictions of unclicked ads – which is close to one in Figure 3.7.

We repeat the experiment of processing 2×10^4 feature vectors that we sumamrized in Figure 3.7 but now we use the objective cost in (3.110) instead of the cost in (3.107). We set $\gamma = 18.2$ which makes replicated clicked ads as numerous as unclicked ads. The resulting SGD and oLBFGS histograms of the predicted click through rates for all clicked ads and complementary predicted click through rates for all unclicked ads are shown in Figure 3.8. In particular, Figure 7.8(a) shows the histograms of predicted click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$

for all clicked ads after processing 2×10^4 training sample points. The modification of the log likelihood cost increases the accuracy of the oLBFGS classifier which is now predicting a click probability $\mathrm{CTR}(\mathbf{x}; \mathbf{w}) \in [0.9, 1]$ for 54.7% of the ads that were indeed clicked. There is also improvement for the SGD classifier but the prediction is much less impressive. Only 15.5% of the clicked ads are associated with a click probability prediction in the interval $[0.9, 1]$. This improvement is at the cost of reducing the complementary predicted click through rate $1 - \mathrm{CTR}(\mathbf{x}; \mathbf{w})$ for the ads that were indeed not clicked. However, the classifier computed by oLBFGS after processing 2×10^4 feature vectors still predicts a probability $1 - \mathrm{CTR}(\mathbf{x}; \mathbf{w}) \in [0.9, 1]$ for 46.3% of the unclicked ads. The corresponding frequency for the SGD classifier is 10.8%.

Do note that the relatively high prediction accuracies in Figure 3.8 are a reflection of sample bias to some extent. Since ads were chosen for display because they were deemed likely to be clicked they are not a completely random test set. Still, the point to be made here is that oLBFGS succeeds in finding an optimal classifier when SGD fails. It would take the processing of about 10^6 feature vectors for SGD to achieve the same accuracy of oLBFGs.

Chapter 4

Superlinearly convergent incremental quasi-Newton method

4.1 Context and background

In this chapter, we focus on large scale optimization problems with objective functions expressed as the sum of a set of components which arise often in application domains such as machine learning [14,15,26,108], control [22,25,56], and wireless communications [96,97,103]. This class of problems is also called convex finite sum minimization. Formally, we consider a variable $\mathbf{w} \in \mathbb{R}^p$ and a function f which is defined as the average of N smooth and strongly convex functions labelled $f_i : \mathbb{R}^p \to \mathbb{R}$ for $i = 1, \ldots, N$. We refer to individual functions f_i as sample functions and to the total number of functions N as the sample size. Our goal is to find the optimal argument $\mathbf{w}^*$ that solves the strongly convex program

$$\mathbf{w}^* := \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^p} f(\mathbf{w}) := \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^p} \frac{1}{N} \sum_{i=1}^{N} f_i(\mathbf{w}). \tag{4.1}$$

As mentioned in Chapter 1, the finite sum minimization (FSM) in (4.1) is a general formulation which contains the empirical risk minimization (ERM) problem. To keep the results as general as possible we state the results for the FSM problem but, indeed, the results in this chapter also hold for ERM problems.

We restrict attention to cases where the component functions f_i are strongly convex and their gradients are Lipschitz continuous. We further focus in problems where N is large enough so as to warrant application of stochastic or iterative methods. Our goal is to propose an iterative quasi-Newton method to solve (4.1) which is shown to exhibit a local superlinear convergence rate. This is achieved while performing local iterations with a cost of order $\mathcal{O}(p^2)$ independent of the number of samples N.

Setting temporarily aside the complications related to the number of component functions, the minimization of f in (4.1) can be carried out using iterative descent algorithms. A simple solution is to use gradient descent (GD) which iteratively descends along gradient directions $\nabla f(\mathbf{w}) = (1/N)\sum_{i=1}^{N} \nabla f_i(\mathbf{w})$. GD incurs a per iteration computational cost of order $\mathcal{O}(Np)$ and is known to converge at a linear rate towards $\mathbf{w}^*$ under the hypotheses we have placed on f. Whether the linear convergence rate of GD is acceptable depends on the desired accuracy and on the condition number of f which, when large, can make the convergence constant close to one. As one or both of these properties often limit the applicability of GD, classical alternatives to improve convergence rates have been developed. Newton's method adapts to the curvature of the objective by computing Hessian inverses and converges at a quadratic rate in a local neighborhood of the optimal argument irrespective of the problem's condition number. To achieve this quadratic convergence rate, we must evaluate and invert Hessians resulting in a per iteration cost of order $\mathcal{O}(Np^2 + p^3)$. Quasi-Newton methods build on the idea of approximating the Newton step using first-order information of the objective function and exhibit local superlinear convergence [21, 32, 89]. An important feature of quasi-Newton methods is that they have a per iteration cost of order $\mathcal{O}(Np + p^2)$, where the term $\mathcal{O}(Np)$ corresponds to the cost of gradient computation and the cost $\mathcal{O}(p^2)$ indicates the computational complexity of updating the approximate Hessian inverse matrix.

The combination of a local superlinear convergence rate and the smaller computational cost per iteration relative to Newton – a reduction by a factor of p operations per iteration – make quasi-Newton methods an appealing choice. In the context of optimization problems having the form in (4.1), quasi-Newton methods also have the advantage that curvature is estimated using gradient evaluations. To see why this is meaningful we must recall that the customary approach to avoid the $\mathcal{O}(Np)$ computational cost of GD iterations is to replace gradients $\nabla f(\mathbf{w})$ by their stochastic approximations $\nabla f_i(\mathbf{w})$, which can be evaluated with a cost of order $\mathcal{O}(p)$. One can then think of using stochastic versions of these gradients to develop stochastic quasi-Newton methods with per iterations cost of order $\mathcal{O}(p + p^2)$. This idea was demonstrated to be feasible in [105] which introduces a stochastic (online) version of the BFGS quasi-Newton method as well as a stochastic version of its limited memory variant. Although [105] provides numerical experiments illustrating significant improvements in convergence times relative to stochastic (S) GD, theoretical guarantees are not established.

The issue of proving convergence of stochastic quasi-Newton methods is tackled in Chapters 2 and 3. In Chapter 2, we observed that stochastic BFGS may not be convergent because the Hessian approximation matrices can become close to singular. A regularized stochastic BFGS (RES) method was proposed by changing the proximity condition of BFGS to ensure

that the eigenvalues of the Hessian inverse approximation are uniformly bounded. Enforcing this property yields a provably convergent algorithm. In Chapter 3, we showed that the limited memory version of stochastic (online) BFGS proposed in [105] is almost surely convergent and has a sublinear convergence rate in expectation. This is achieved without using regularizations. An alternative provably convergent stochastic quasi-Newton method is proposed in [23]. This method differs from those in [70,72,105] in that it collects (stochastic) second order information to estimate the objective's curvature. This is in contrast to estimating curvature using the difference of two consecutive stochastic gradients.

Although the methods in [23, 70, 72, 105] are successful in expanding the application of quasi-Newton methods to stochastic settings, their convergence rate is sublinear. This is not better than the convergence rate of SGD and, as is also the case in SGD, is a consequence of the stochastic approximation noise which necessitates the use of diminishing stepsizes. The stochastic quasi-Newton methods in [58, 78] resolve this issue by using the variance reduction technique proposed in [45]. The fundamental idea of the work in [45] is to reduce the noise of the stochastic gradient approximation by computing the exact gradient in an outer loop to use it in an inner loop for gradient approximation. The methods in [58, 78], which incorporate the variance reduction scheme presented in [45] into the update of quasi-Newton methods, are successful in achieving a linear convergence rate.

At this point, we must remark on an interesting mismatch. The convergence rate of SGD is sublinear, and the convergence rate of deterministic GD is linear. The use of variance reduction techniques in SGD recovers the linear convergence rate of GD, [45]. On the other hand, the convergence rate of stochastic quasi-Newton methods is sublinear, and the convergence rate of deterministic quasi-Newton methods is superlinear. The use of variance reduction in stochastic quasi-Newton methods achieves linear convergence but does not recover a superlinear rate. Hence, a fundamental question remains unanswered: Is it possible to design an incremental quasi-Newton method that recovers the superlinear convergence rate of deterministic quasi-Newton algorithms? In this paper, we show that the answer to this open problem is positive by proposing an incremental quasi-Newton method (IQN) with a local superlinear convergence rate. This is the first quasi-Newton method to achieve superlinear convergence while having a per iteration cost independent of the number of functions N – the cost per iteration is of order $\mathcal{O}(p^2)$.

There are three major differences between the IQN method and state-of-the-art incremental (stochastic) quasi-Newton methods that lead to the former's superlinear convergence rate. First, the proposed IQN method uses the aggregated information of variables, gradients, and Hessian approximation matrices to reduce the noise of approximation for both gradients and Hessian approximation matrices. This is different to the variance-reduced stochastic quasi-Newton methods in [58, 78] that attempt to reduce only the noise of gra-

dient approximations. Second, in IQN the index of the updated function is chosen in a cyclic fashion, rather than the random selection scheme used in the incremental methods in [23, 70, 72, 105]. The cyclic routine in IQN allows to bound the error at each iteration as a function of the errors of the last N iterates, something that is not possible when using a random scheme. To explain the third and most important difference we point out that the form of quasi-Newton updates is the solution of a local second order Taylor approximation of the objective. It is possible to understand stochastic quasi-Newton methods as an analogous approximation of individual sample functions. However, it turns out that the state-of-the-art stochastic quasi-methods evaluate the linear and quadratic terms of the Taylor's expansion at different points yielding and inconsistent approximation (Remark 4.7). The IQN method utilizes a consistent Taylor series which yields a more involved update which we nonetheless show can be implemented with the same computational cost. These three properties together lead to an incremental quasi-Newton method with a local superlinear convergence rate.

4.1.1 Related work

Various methods have been studied in the literature to improve the performance of traditional full-batch optimization algorithms. The most famous method for reducing the computational complexity of gradient descent (GD) is stochastic gradient descent (SGD), which uses the gradient of a single randomly chosen function to approximate the full-gradient [15]. Incremental gradient descent method (IGD) is similar to SGD except the function is chosen in a cyclic routine [11]. Both SGD and IGD suffer from slow sublinear convergence rate because of the noise of gradient approximation. The incremental aggregated methods, which use memory to aggregate the gradients of all N functions, are successful in reducing the noise of gradient approximation to achieve linear convergence rate [31,45,49,104]. The work in [49] suggests a random selection of functions which leads to stochastic average gradient method (SAG), while the works in [11, 38, 62] use a cyclic scheme.

Moving beyond first order information, there have been stochastic quasi-Newton methods to approximate Hessian information [37, 70, 72, 78, 105]. All of these stochastic quasi-Newton methods reduce computational cost of quasi-Newton methods by updating only a randomly chosen single or small subset of gradients at each iteration. However, they are not able to recover the superlinear convergence rate of quasi-Newton methods [21, 32, 89]. The incremental Newton method (NIM) in [99] is the only incremental method shown to have a superlinear convergence rate; however, the Hessian function is not always available or computationally feasible. Moreover, the implementation of NIM requires computation of the incremental aggregated Hessian inverse which has the computational complexity of the order $\mathcal{O}(p^3)$.

4.1.2 Outline

We start the paper by recapping the BFGS quasi-Newton method and the Dennis-Moré condition which is sufficient and necessary to prove superlinear convergence rate of the BFGS method (Section 4.2). Then, we present the proposed Incremental Quasi-Newton method (IQN) as an incremental aggregated version of the traditional BFGS method (Section 4.3). We first explain the difference between the Taylor's expansion used in IQN and state-of-the-art incremental (stochastic) quasi-Newton methods. Further, we explain the mechanism for aggregation of the functions information and the scheme for updating the stored information. Moreover, we present an efficient implementation of the proposed IQN method with a computational complexity of the order $\mathcal{O}(p^2)$ (Section 4.3.1). The convergence analysis of the IQN method is then presented (Section 4.4). We use the classic analysis of quasi-Newton methods to show that in a local neighborhood of the optimal solution the sequence of variables converges to the optimal argument $\mathbf{w}^*$ linearly after each pass over the set of functions (Lemma 11). We use this result to show that for each component function f_i the Dennis-Moré condition holds (Proposition 3). However, this condition is not sufficient to prove superlinear convergence of the sequence of errors $\|\mathbf{w}^t - \mathbf{w}^*\|$, since it does not guarantee the Dennis-Moré condition for the global objective f. To overcome this issue we introduce a novel convergence analysis approach which exploits the local linear convergence of IQN to present a more general version of the Dennis-Moré condition for each component function f_i (Lemma 13). We exploit this result to establish local superlinear convergence of the sequence of residuals with respect to the average sequence (Theorem 5). Further, we show that there exists a superlinearly convergent sequence that is an upper bound for the original sequence of errors $\|\mathbf{w}^t - \mathbf{w}^*\|$ (Theorem 6). Then, we present numerical simulation results, comparing the performance of IQN to that of first-order incremental and stochastic methods (Section 4.5). We test the performance on a set of large-scale regression problems and observe strong numerical gain in total computation time relative to existing methods.

Notation Vectors are written as lowercase $\mathbf{w} \in \mathbb{R}^p$ and matrices as uppercase $\mathbf{A} \in \mathbb{R}^{p \times p}$. We use $\|\mathbf{w}\|$ and $\|\mathbf{A}\|$ to denote the Euclidean norm of vector $\mathbf{w}$ and matrix $\mathbf{A}$, respectively. Given a positive definite matrix $\mathbf{M}$, the weighted matrix norm $\|\mathbf{A}\|_{\mathbf{M}}$ is defined as $\|\mathbf{A}\|_{\mathbf{M}} := \|\mathbf{MAM}\|_{\mathbf{F}}$, where $\|.\|_{\mathbf{F}}$ is the Frobenius norm. Given a function f its gradient and Hessian at point $\mathbf{w}$ are denoted as $\nabla f(\mathbf{w})$ and $\nabla^2 f(\mathbf{w})$, respectively.

4.2 BFGS quasi-Newton method

Consider the problem in (4.1) for relatively large N. In a conventional optimization setting, this can be solved using a quasi-Newton method that iteratively updates a variable $\mathbf{w}^t$ for

$t = 0, 1, \ldots$ based on the general recursive expression

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta^t (\mathbf{B}^t)^{-1} \nabla f(\mathbf{w}^t), \tag{4.2}$$

where η^t is a scalar stepsize and $\mathbf{B}^t$ is a positive definite matrix that approximates the exact Hessian of the objective function $\nabla^2 f(\mathbf{w}^t)$. The stepsize η^t is evaluated based on a line search routine for the global convergence of quasi-Newton methods. Our focus in this paper, however, is on the local convergence of quasi-Newton methods, which requires the unit stepsize $\eta^t = 1$. Therefore, throughout the paper we assume that the variable $\mathbf{w}^t$ is close to the optimal solution $\mathbf{w}^*$ – we will formalize the notion of being close to the optimal solution – and the stepsize is $\eta^t = 1$.

The goal of quasi-Newton methods is to compute the Hessian approximation matrix $\mathbf{B}^t$ and its inverse $(\mathbf{B}^t)^{-1}$ by using only the first-order information, i.e., gradients, of the objective. Their use is widespread due to the many applications in which the Hessian information required in Newton's method is either unavailable or computationally intensive. There are various approaches to approximate the Hessian, but the common feature among quasi-Newton methods is that the Hessian approximation must satisfy the secant condition. To be more precise, consider $\mathbf{s}^t$ and $\mathbf{y}^t$ as the variable and gradient variations, explicitly defined as

$$\mathbf{s}^t := \mathbf{w}^{t+1} - \mathbf{w}^t, \qquad \mathbf{y}^t := \nabla f(\mathbf{w}^{t+1}) - \nabla f(\mathbf{w}^t). \tag{4.3}$$

Then, given the variable variation $\mathbf{s}^t$ and gradient variation $\mathbf{y}^t$, the Hessian approximation matrix in all quasi-Newton methods must satisfy the secant condition

$$\mathbf{B}^{t+1}\mathbf{s}^t = \mathbf{y}^t. \tag{4.4}$$

This condition is fundamental in quasi-Newton methods because the exact Hessian $\nabla^2 f(\mathbf{w}^t)$ satisfies this equality when the iterates $\mathbf{w}^{t+1}$ and $\mathbf{w}^t$ are close to each other. If we consider the matrix $\mathbf{B}^{t+1}$ as the unknown matrix, the system of equations in (4.4) does not have a unique solution. Different quasi-Newton methods enforce different conditions on the matrix $\mathbf{B}^{t+1}$ to come up with a unique update. This extra condition is typically a proximity condition that ensures that $\mathbf{B}^{t+1}$ is close to the previous Hessian approximation matrix $\mathbf{B}^t$ [21,32,89]. In particular, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method defines the update of Hessian approximation matrix as

$$\mathbf{B}^{t+1} = \mathbf{B}^t + \frac{\mathbf{y}^t \mathbf{y}^{tT}}{\mathbf{y}^{tT}\mathbf{s}^t} - \frac{\mathbf{B}^t \mathbf{s}^t \mathbf{s}^{tT} \mathbf{B}^t}{\mathbf{s}^{tT}\mathbf{B}^t\mathbf{s}^t}. \tag{4.5}$$

The BFGS method is popular not only for its strong numerical performance relative to the gradient descent method, but also because it is shown to exhibit a superlinear convergence rate [21], thereby providing a theoretical guarantee of superior performance. In fact, it can be shown that, the BFGS update satisfies the condition

$$\lim_{t\to\infty} \frac{\|(\mathbf{B}^t - \nabla^2 f(\mathbf{w}^*))\mathbf{s}^t\|}{\|\mathbf{s}^t\|} = 0, \tag{4.6}$$

known as the Dennis-Moré condition, which is both necessary and sufficient for superlinear convergence [32]. This result solidifies quasi-Newton methods as a strong alternative to first order methods when exact second-order information is unavailable. However, implementation of the BFGS method is not feasible when the number of functions N is large, due to its high computational complexity on the order $\mathcal{O}(Np + p^2)$. In the following section, we propose a novel incremental BFGS method that has the computational complexity of $\mathcal{O}(p^2)$ per iteration and converges at a superlinear rate.

4.3 IQN: Incremental aggregated BFGS

We propose an incremental aggregated BFGS algorithm, which we call the Incremental Quasi-Newton (IQN) method. The IQN method is incremental in that, at each iteration, only the information associated with a single function f_i is updated. The particular function is chosen by cyclicly iterating through the N functions. The IQN method is aggregated in that the aggregate of the most recently observed information of all functions $f_1, \ldots, f_n$ is used to compute the updated variable $\mathbf{w}^{t+1}$.

In the proposed method, we consider $\mathbf{z}_1^t, \ldots, \mathbf{z}_n^t$ as the copies of the variable $\mathbf{w}$ at time t associated with the functions $f_1, \ldots, f_n$, respectively. Likewise, define $\nabla f_i(\mathbf{z}_i^t)$ as the gradient corresponding to the i-th function. Further, consider $\mathbf{B}_i^t$ as a positive definite matrix which approximates the i-th component Hessian $\nabla^2 f_i(\mathbf{w}^t)$. We refer to $\mathbf{z}_i^t$, $\nabla f_i(\mathbf{z}_i^t)$, and $\mathbf{B}_i^t$ as the information corresponding to the i-th function f_i at step t. Note that the functions' information is stored in a shared memory as shown in Fig. 4.1. To introduce the IQN method, we first explain the mechanism for computing the updated variable $\mathbf{w}^{t+1}$ using the stored information $\{\mathbf{z}_i^t, \nabla f_i(\mathbf{z}_i^t), \mathbf{B}_i^t\}_{i=1}^N$. Then, we elaborate on the scheme for updating the information of the functions.

To derive the full variable update, consider the second order approximation of the objective function $f_i(\mathbf{w})$ centered around its current iterate $\mathbf{z}_i^t$,

$$f_i(\mathbf{w}) \approx f_i(\mathbf{z}_i^t) + \nabla f_i(\mathbf{z}_i^t)^T(\mathbf{w} - \mathbf{z}_i^t) + \frac{1}{2}(\mathbf{w} - \mathbf{z}_i^t)^T \nabla^2 f_i(\mathbf{z}_i^t)(\mathbf{w} - \mathbf{z}_i^t). \tag{4.7}$$

As in traditional quasi-Newton methods, we replace the i-th Hessian $\nabla^2 f_i(\mathbf{z}_i^t)$ by $\mathbf{B}_i^t$. Using

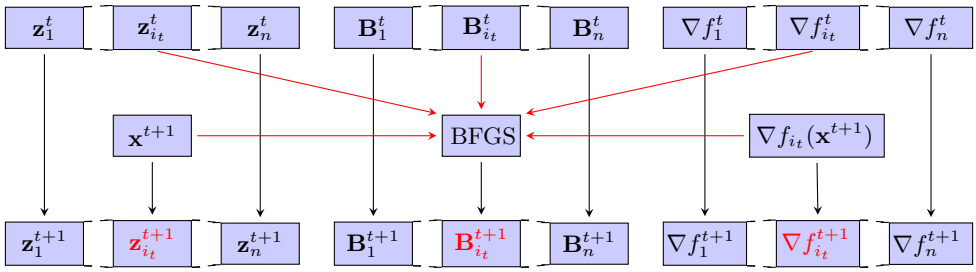


Figure 4.1: The updating scheme for variables, gradients, and Hessian approximation matrices of function f_{i_t} at step t. The red arrows indicate the terms used in the update of $\mathbf{B}_{i_t}^{t+1}$ using the BFGS update in (4.15). The black arrows show the updates of all variables and gradients. The terms $\mathbf{z}_{i_t}^{t+1}$ and $\nabla f_{i_t}^{t+1}$ are updated as $\mathbf{w}^{t+1}$ and $\nabla f_{i_t}(\mathbf{w}^{t+1})$, respectively. All others $\mathbf{z}_j^{t+1}$ and ∇f_j^{t+1} are set as $\mathbf{z}_j^t$ and ∇f_j^t, respectively.

the approximation matrices in place of Hessians, the complete (aggregate) function $f(\mathbf{w})$ can be approximated with

$$f(\mathbf{w}) \approx \frac{1}{N}\sum_{i=1}^{N}\left[f_i(\mathbf{z}_i^t) + \nabla f_i(\mathbf{z}_i^t)^T(\mathbf{w}-\mathbf{z}_i^t) + \frac{1}{2}(\mathbf{w}-\mathbf{z}_i^t)^T\mathbf{B}_i^t(\mathbf{w}-\mathbf{z}_i^t)\right]. \tag{4.8}$$

Note that the right hand side of (4.8) is a quadratic approximation of the function f based on the available information at step t. Hence, the updated iterate $\mathbf{w}^{t+1}$ can be defined as the minimizer of the quadratic program in (4.8), explicitly given by

$$\mathbf{w}^{t+1} = \left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_i^t\right)^{-1}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_i^t\mathbf{z}_i^t - \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\mathbf{z}_i^t)\right]. \tag{4.9}$$

First note that the update in (4.9) shows that the updated variable $\mathbf{w}^{t+1}$ is a function of the stored information of all functions $f_1,\ldots,f_n$. Furthermore, we use the aggregated information of variables, gradients, and the quasi-Newton Hessian approximations to evaluate the updated variable. This is done to vanish the noise in approximating both gradients and Hessians as the sequence approaches the optimal argument.

Remark 3 *Given the BFGS Hessian approximation matrices $\{\mathbf{B}_i^t\}_{i=1}^N$ and the gradients $\{\nabla f_i(\mathbf{z}_i^t)\}_{i=1}^N$, one may consider an update more akin to traditional descent-based methods, i.e.,*

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_i^t\right)^{-1}\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\mathbf{z}_i^t). \tag{4.10}$$

To evaluate the advantage of the proposed update for IQN in (4.9) *relative to the update in*

(4.10), *we proceed to study the Taylor's expansion that leads to the update in* (4.10). *It can be shown that the update in* (4.10) *is the outcome of the following approximation*

$$f(\mathbf{w}) \approx \frac{1}{N}\sum_{i=1}^{N}\left[f_i(\mathbf{z}_i^t) + \nabla f_i(\mathbf{z}_i^t)^T(\mathbf{w}-\mathbf{z}_i^t) + \frac{1}{2}(\mathbf{w}-\mathbf{w}^t)^T\mathbf{B}_i^t(\mathbf{w}-\mathbf{w}^t)\right]. \tag{4.11}$$

Observe that the linear term in (4.11) *is centered at* $\mathbf{z}_i^t$, *while the quadratic term is approximated near the iterate* $\mathbf{w}^t$. *This inconsistency in the Taylor's expansion of each function* f_i *leads to an inaccurate second-order approximation, and subsequently a slower incremental quasi-Newton method.*

Thus far we have discussed the procedure to compute the updated variable $\mathbf{w}^{t+1}$ given the local iterates, gradients, and Hessian approximations at time t. Now, it remains to show how we update the local information of functions $f_1, \ldots, f_n$ using the variable $\mathbf{w}^{t+1}$. In each iteration of the IQN method, we update the local information of only a single function, chosen in a cyclic manner. Defining i_t to be the index of the function selected at time t, we update the local variables $\mathbf{z}_{i_t}^{t+1}$, $\nabla f_{i_t}(\mathbf{z}_i^{t+1})$, and $\mathbf{B}_i^{t+1}$ using the updated variable $\mathbf{w}^{t+1}$ while all other local variables remain unchanged. In particular, the variables $\mathbf{z}_i$ are updated as

$$\mathbf{z}_{i_t}^{t+1} = \mathbf{w}^{t+1}, \qquad \mathbf{z}_i^{t+1} = \mathbf{z}_i^t \quad \text{for all } i \neq i_t. \tag{4.12}$$

Observe in the update in (4.12) that the variable associated with the function f_{i_t} is set to be the updated variable $\mathbf{w}^{t+1}$ while the other iterates are simply kept as their previous value. Likewise, we update the table of gradients accordingly with the gradient of f_{i_t} evaluated at the new variable $\mathbf{w}^{t+1}$. The rest of gradients stored in the memory will stay unchanged, i.e.,

$$\nabla f_{i_t}(\mathbf{z}_i^{t+1}) = \nabla f_{i_t}(\mathbf{w}^{t+1}), \qquad \nabla f_i(\mathbf{z}_i^{t+1}) = \nabla f_i(\mathbf{z}_i^t) \quad \text{for all } i \neq i_t. \tag{4.13}$$

To update the curvature information, it would be ideal to compute the Hessian matrix $\nabla^2 f_{i_t}(\mathbf{w}^{t+1})$ and update the curvature information following the schemes for variables in (4.12) and gradients in (4.13). However, our focus is on the applications that the computation of the Hessian is either impossible or computationally expensive. Hence, to the update curvature approximation matrix $\mathbf{B}_{i_t}^t$ corresponding to the function f_{i_t}, we use the steps of BFGS in (4.5). To do so, we define variable and gradient variations associated with each individual function f_i as

$$\mathbf{s}_i^t := \mathbf{z}_i^{t+1} - \mathbf{z}_i^t, \qquad \mathbf{y}_i^t := \nabla f_i(\mathbf{z}_i^{t+1}) - \nabla f_i(\mathbf{z}_i^t), \tag{4.14}$$

respectively. The Hessian approximation $\mathbf{B}_{i_t}^t$ corresponding to the function f_{i_t} can be computed using the update of BFGS as

$$\mathbf{B}_i^{t+1} = \mathbf{B}_i^t + \frac{\mathbf{y}_i^t \mathbf{y}_i^{tT}}{\mathbf{y}_i^{tT} \mathbf{s}_i^t} - \frac{\mathbf{B}_i^t \mathbf{s}_i^t \mathbf{s}_i^{tT} \mathbf{B}_i^t}{\mathbf{s}_i^{tT} \mathbf{B}_i^t \mathbf{s}_i^t}, \quad \text{for} \quad i = i_t. \tag{4.15}$$

Again, the Hessian approximation matrices for all other functions remain unchanged, i.e., $\mathbf{B}_i^{t+1} = \mathbf{B}_i^t$ for $i \neq i_t$. The system of updates in (4.12)-(4.15) explains the mechanism of updating the information of the function f_{i_t} at step t. Notice that to update the Hessian approximation matrix for the i_t-th function there is no need to store the variations in (4.14), since the old variables $\mathbf{z}_i^t$ and $\nabla f_i(\mathbf{z}_i^t)$ are available in memory and the updated versions $\mathbf{z}_i^{t+1} = \mathbf{w}^{t+1}$ and $\nabla f_i(\mathbf{z}_i^{t+1}) = \nabla f_i(\mathbf{w}^{t+1})$ are evaluated at step t; see Fig. 4.1 for more details.

Because of the cyclic update scheme, the set of iterates $\{\mathbf{z}_1^t, \mathbf{z}_2^t, \ldots, \mathbf{z}_n^t\}$ is equal to the set $\{\mathbf{w}^t, \mathbf{w}^{t-1}, \ldots, \mathbf{w}^{t-n+1}\}$, and, therefore, the set of variables used in the update of IQN is the set of the last N iterates. The update of IQN in (4.9) incorporates the information of all the functions $f_1, \ldots, f_n$ to compute the updated variable $\mathbf{w}^{t+1}$; however, it uses delayed variables, gradients, and Hessian approximations rather than the the updated variable $\mathbf{w}^{t+1}$ for all functions as in classic quasi-Newton methods. The use of delay allows IQN to update the information of a single function at each iteration, thus reducing the computational complexity relative to classic quasi-Newton methods.

Although the update in (4.9) is helpful in understanding the rationale behind the IQN method, it cannot be implemented at a low computation cost, since it requires computation of the sums $\sum_{i=1}^N \mathbf{B}_i^t$, $\sum_{i=1}^N \mathbf{B}_i^t \mathbf{z}_i^t$, and $\sum_{i=1}^N \nabla f_i(\mathbf{z}_i^t)$ as well as computing the inversion $(\sum_{i=1}^N \mathbf{B}_i^t)^{-1}$. In the following section, we introduce an efficient implementation of the IQN method that has the computational complexity of $\mathcal{O}(p^2)$.

4.3.1 Efficient implementation of IQN

To see that the updating scheme in (4.9) requires evaluation of only a single gradient and Hessian approximation matrix per iteration, consider writing the update as

$$\mathbf{w}^{t+1} = (\tilde{\mathbf{B}}^t)^{-1} \left(\mathbf{u}^t - \mathbf{g}^t \right), \tag{4.16}$$

where we define $\tilde{\mathbf{B}}^t := \sum_{i=1}^N \mathbf{B}_i^t$ as the aggregate Hessian approximation, $\mathbf{u}^t := \sum_{i=1}^N \mathbf{B}_i^t \mathbf{z}_i^t$ as the aggregate Hessian-variable product, and $\mathbf{g}^t := \sum_{i=1}^N \nabla f_i(\mathbf{z}_i^t)$ as the aggregate gradient. Then, given that at step t only a single index i_t is updated, we can evaluate these variables

Algorithm 4 Incremental Quasi-Newton (IQN) method

Require: $\mathbf{w}^0$,$\{\nabla f_i(\mathbf{w}^0)\}_{i=1}^N$, $\{\mathbf{B}_i^0\}_{i=1}^N$
1: Set $\mathbf{z}_1^0 = \cdots = \mathbf{z}_n^0 = \mathbf{w}^0$
2: Set $(\tilde{\mathbf{B}}^0)^{-1} = (\sum_{i=1}^N \mathbf{B}_i^0)^{-1}$, $\mathbf{u}^0 = \sum_{i=1}^N \mathbf{B}_i^0 \mathbf{w}^0$, $\mathbf{g}^0 = \sum_{i=1}^N \nabla f_i(\mathbf{w}^0)$
3: **for** $t = 0, 1, 2, \ldots$ **do**
4: Set $i_t = (t \mod n) + 1$
5: Compute $\mathbf{w}^{t+1} = (\tilde{\mathbf{B}}^t)^{-1}(\mathbf{u}^t - \mathbf{g}^t)$ [cf. (4.16)]
6: Compute $\mathbf{s}_{i_t}^{t+1}$, $\mathbf{y}_{i_t}^{t+1}$ [cf. (4.14)], and $\mathbf{B}_{i_t}^{t+1}$ [cf. (4.15)]
7: Update $\mathbf{u}^{t+1}$ [cf. (4.18)], $\mathbf{g}^{t+1}$ [cf. (4.19)], and $(\tilde{\mathbf{B}}^{t+1})^{-1}$ [cf. (4.21), (4.22)]
8: Update the functions' information tables as in (4.12), (4.13), and (4.15)
9: **end for**

for step $t+1$ as

$$\tilde{\mathbf{B}}^{t+1} = \tilde{\mathbf{B}}^t + \left(\mathbf{B}_{i_t}^{t+1} - \mathbf{B}_{i_t}^t\right), \tag{4.17}$$

$$\mathbf{u}^{t+1} = \mathbf{u}^t + \left(\mathbf{B}_{i_t}^{t+1}\mathbf{z}_{i_t}^{t+1} - \mathbf{B}_{i_t}^t\mathbf{z}_{i_t}^t\right), \tag{4.18}$$

$$\mathbf{g}^{t+1} = \mathbf{g}^t + \left(\nabla f_{i_t}(\mathbf{z}_{i_t}^{t+1}) - \nabla f_{i_t}(\mathbf{z}_{i_t}^t)\right). \tag{4.19}$$

Thus, only $\mathbf{B}_{i_t}^{t+1}$ and $\nabla f_{i_t}(\mathbf{z}_{i_t}^{t+1})$ are required to be computed at step t.

Although the updates in (4.17)-(4.19) have low computational complexity, the update in (4.16) requires computing $(\tilde{\mathbf{B}}^t)^{-1}$ which has a computational complexity of $\mathcal{O}(p^3)$. This inversion can be avoided by simplifying the update in (4.17) as

$$\tilde{\mathbf{B}}^{t+1} = \tilde{\mathbf{B}}^t + \frac{\mathbf{y}_{i_t}^t \mathbf{y}_{i_t}^{tT}}{\mathbf{y}_i^{tT}\mathbf{s}_t^{i_t}} - \frac{\mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^t \mathbf{s}_{i_t}^{tT} \mathbf{B}_{i_t}^t}{\mathbf{s}_{i_t}^{tT}\mathbf{B}_{i_t}^t\mathbf{s}_{i_t}^t}. \tag{4.20}$$

To derive the expression in (4.20) we have substituted the difference $\mathbf{B}_{i_t}^{t+1} - \mathbf{B}_{i_t}^t$ by its rank two expression in (4.15). Given the matrix $(\tilde{\mathbf{B}}^t)^{-1}$, by applying the Sherman-Morrison formula twice to the update in (4.20) we can compute $(\tilde{\mathbf{B}}^{t+1})^{-1}$ as

$$(\tilde{\mathbf{B}}^{t+1})^{-1} = \mathbf{U}^t + \frac{\mathbf{U}^t(\mathbf{B}_{i_t}^t\mathbf{s}_{i_t}^t)(\mathbf{B}_{i_t}^t\mathbf{s}_{i_t}^t)^T\mathbf{U}^t}{{\mathbf{s}_{i_t}^t}^T\mathbf{B}_{i_t}^t\mathbf{s}_{i_t}^t - (\mathbf{B}_{i_t}^t\mathbf{s}_{i_t}^t)^T\mathbf{U}^t(\mathbf{B}_{i_t}^t\mathbf{s}_{i_t}^t)}, \tag{4.21}$$

where the matrix $\mathbf{U}^t$ is evaluated as

$$\mathbf{U}^t = (\tilde{\mathbf{B}}^t)^{-1} - \frac{(\tilde{\mathbf{B}}^t)^{-1}\mathbf{y}_{i_t}^t\mathbf{y}_{i_t}^{tT}(\tilde{\mathbf{B}}^t)^{-1}}{\mathbf{y}_{i_t}^{tT}\mathbf{s}_{i_t}^t + \mathbf{y}_{i_t}^{tT}(\tilde{\mathbf{B}}^t)^{-1}\mathbf{y}_{i_t}^t}. \tag{4.22}$$

The computational complexity of the updates in (4.21) and (4.22) is of the order $\mathcal{O}(p^2)$ rather than the $\mathcal{O}(p^3)$ cost of computing the inverse directly. Therefore, the overall cost of IQN is of the order $\mathcal{O}(p^2)$ which is substantially lower than $\mathcal{O}(Np^2)$ of deterministic quasi-Newton methods.

The complete IQN algorithm is outlined in Algorithm 4. Beginning with initial variable $\mathbf{w}^0$ and gradient and Hessian estimates $\nabla f_i(\mathbf{w}^0)$ and $\mathbf{B}_i^0$ for all i, each variable copy $\mathbf{z}_i^0$ is set to $\mathbf{w}^0$ in Step 1 and initial values are set for $\mathbf{u}^0$, $\mathbf{g}^0$ and $(\tilde{\mathbf{B}}^0)^{-1}$ in Step 2. For all t, in Step 4 the index i_t of the next function to update is selected cyclically. The variable $\mathbf{w}^{t+1}$ is computed according to the update in (4.16) in Step 5. In Step 6, the variable $\mathbf{s}_{i_t}^{t+1}$ and gradient $\mathbf{y}_{i_t}^{t+1}$ variations are evaluated as in (4.14) to compute the BFGS matrix $\mathbf{B}_{i_t}^{t+1}$ from the update in (4.15). This information, as well as the updated variable and its gradient, are used in Step 7 to update $\mathbf{u}^{t+1}$ and $\mathbf{g}^{t+1}$ as in (4.18) and (4.19), respectively. The inverse matrix $(\tilde{\mathbf{B}}^{t+1})^{-1}$ is also computed by following the expressions in (4.21) and (4.22). Finally in Step 8, we update the variable, gradient, and Hessian approximation tables based on the policies in (4.12), (4.13), and (4.15), respectively.

4.4 Convergence analysis

In this section, we study the convergence rate of the proposed IQN method. We first establish its local linear convergence rate, then demonstrate limit properties of the Hessian approximations, and finally show that in a region local to the optimal point the sequence of residuals converges at a superlinear rate. To prove these results we make two main assumptions, both of which are standard in the analysis of quasi-Newton methods.

Assumption 7 *There exist positive constants* $0 < \mu \leq L$ *such that, for all* i *and* $\mathbf{w}, \hat{\mathbf{w}} \in \mathbb{R}^p$*, we can write*

$$\mu\|\mathbf{w} - \hat{\mathbf{w}}\|^2 \leq (\nabla f_i(\mathbf{w}) - \nabla f_i(\hat{\mathbf{w}}))^T(\mathbf{w} - \hat{\mathbf{w}}) \leq L\|\mathbf{w} - \hat{\mathbf{w}}\|^2. \tag{4.23}$$

Assumption 8 *There exists a positive constant* $0 < \tilde{L}$ *such that, for all* i *and* $\mathbf{w}, \hat{\mathbf{w}} \in \mathbb{R}^p$*, we can write*

$$\|\nabla^2 f_i(\mathbf{w}) - \nabla^2 f_i(\hat{\mathbf{w}})\| \leq \tilde{L}\|\mathbf{w} - \hat{\mathbf{w}}\|. \tag{4.24}$$

The lower bound in (4.23) implies that the functions f_i are strongly convex with constant μ, and the upper bound shows that the gradients ∇f_i are Lipschitz continuous with parameter L.

The condition in Assumption 8, states that the Hessians $\nabla^2 f_i$ are Lipschitz continuous with constant $\tilde{L}$. This assumption is commonly made in the analyses of Newton's method [84] and quasi-Newton algorithms [21, 32, 89]. According to Lemma 3.1 in [21], Lipschitz continuity of the Hessians with constant $\tilde{L}$ implies that for $i = 1, \ldots, n$ and arbitrary vectors $\mathbf{w}, \tilde{\mathbf{w}}, \hat{\mathbf{w}} \in \mathbb{R}^p$ we can write

$$\left\|\nabla^2 f_i(\tilde{\mathbf{w}})(\mathbf{w} - \hat{\mathbf{w}}) - (\nabla f_i(\mathbf{w}) - \nabla f_i(\hat{\mathbf{w}}))\right\| \leq \tilde{L}\|\mathbf{w} - \hat{\mathbf{w}}\| \max\left\{\|\mathbf{w} - \tilde{\mathbf{w}}\|, \|\hat{\mathbf{w}} - \tilde{\mathbf{w}}\|\right\}. \tag{4.25}$$

We use the inequality in (4.25) in the process of proving the convergence of IQN.

The goal of BFGS quasi-Newton methods is to approximate the objective function Hessian using the first-order information. Likewise, in the incremental BFGS method, we aim to show that the Hessian approximation matrices for all functions $f_1, \ldots, f_n$ are close to the exact Hessian. In the following lemma, we study the difference between the i-th optimal Hessian $\nabla^2 f_i(\mathbf{w}^*)$ and its approximation $\mathbf{B}_i^t$ over time.

Lemma 8 *Consider the proposed IQN method in* (4.9). *Further, let i be the index of the updated function at step t, i.e., $i = i_t$. Define the residual sequence for function f_i as $\sigma_i^t := \max\{\|\mathbf{z}_i^{t+1} - \mathbf{w}^*\|, \|\mathbf{z}_i^t - \mathbf{w}^*\|\}$ and set $\mathbf{M} = \nabla^2 f_i(\mathbf{w}^*)^{-1/2}$. If Assumptions 7 and 8 hold and the condition $\sigma_i^t < m/(3\tilde{L})$ is satisfied then*

$$\left\|\mathbf{B}_i^{t+1} - \nabla^2 f_i(\mathbf{w}^*)\right\|_{\mathbf{M}} \leq \left[(1 - \alpha\theta_i^{t\,2})^{1/2} + \alpha_3\sigma_i^t\right] \left\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\right\|_{\mathbf{M}} + \alpha_4\sigma_i^t, \tag{4.26}$$

where α, α_3, and α_4 are some positive bounded constants and

$$\theta_i^t = \frac{\|\mathbf{M}(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))\mathbf{s}_i^t\|}{\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}}\|\mathbf{M}^{-1}\mathbf{s}_i^t\|} \quad \text{for } \mathbf{B}_i^t \neq \nabla^2 f_i(\mathbf{w}^*), \quad \theta_i^t = 0 \quad \text{for } \mathbf{B}_i^t = \nabla^2 f_i(\mathbf{w}^*). \tag{4.27}$$

Proof: To prove the claim in Lemma 8, we first prove the the following lemma which is based on the result in [21, Lemma 5.2].

Lemma 9 *Consider the proposed IQN method in* (4.9). *Let $\mathbf{M}$ be a nonsingular symmetric matrix such that*

$$\|\mathbf{M}\mathbf{y}_i^t - \mathbf{M}^{-1}\mathbf{s}_i^t\| \leq \beta\|\mathbf{M}^{-1}\mathbf{s}_i^t\|, \tag{4.28}$$

for some $\beta \in [0, 1/3]$ and vectors $\mathbf{s}_i^t$ and $\mathbf{y}_i^t$ in $\mathbb{R}^p$ with $\mathbf{s}_i^t \neq \mathbf{0}$. Consider i as the index of the updated function at step t, i.e., $i = i_t$, and let $\mathbf{B}_i^t$ be symmetric and computed according to the update in (4.15). *Then, there exist positive constants α, α_1, and α_2 such that, for any symmetric $\mathbf{A} \in \mathbb{R}^{p\times p}$ we have,*

$$\|\mathbf{B}_i^{t+n} - \mathbf{A}\|_{\mathbf{M}} \leq \left[(1 - \alpha\theta^2)^{1/2} + \alpha_1\frac{\|\mathbf{M}\mathbf{y}_i^t - \mathbf{M}^{-1}\mathbf{s}_i^t\|}{\|\mathbf{M}^{-1}\mathbf{s}_i^t\|}\right] \|\mathbf{B}_i^t - \mathbf{A}\|_{\mathbf{M}} + \alpha_2\frac{\|\mathbf{y}_i^t - \mathbf{A}\mathbf{s}_i^t\|}{\|\mathbf{M}^{-1}\mathbf{s}_i^t\|}, \tag{4.29}$$

where $\alpha = (1 - 2\beta)/(1 - \beta^2) \in [3/8, 1]$, $\alpha_1 = 2.5(1 - \beta)^{-1}$, $\alpha_2 = 2(1 + 2\sqrt{p})\|\mathbf{M}\|_{\mathbf{F}}$, and

$$\theta = \frac{\|\mathbf{M}(\mathbf{B}_i^t - \mathbf{A})\mathbf{s}_i^t\|}{\|\mathbf{B}_i^t - \mathbf{A}\|_{\mathbf{M}}\|\mathbf{M}^{-1}\mathbf{s}_i^t\|} \quad \text{for } \mathbf{B}_i^t \neq \mathbf{A}, \qquad \theta = 0 \quad \text{for } \mathbf{B}_i^t = \mathbf{A}. \tag{4.30}$$

Note that the Hessian approximation $\mathbf{B}_i^{t+n}$ is equal to $\mathbf{B}_i^{t+1}$ if the function f_i is updated at step t. Considering this observation and the result of Lemma 5.2. in [21] the claim in (4.29) follows.

The result in Lemma 9 provides an upper bound for the difference between the Hessian approximation matrix $\mathbf{B}_i^{t+n}$ and any positive definite matrix $\mathbf{A}$ with respect to the difference between the previous Hessian approximation $\mathbf{B}_i^t$ and the matrix $\mathbf{A}$. The interesting choice for the arbitrary matrix $\mathbf{A}$ is the Hessian of the i-th function at the optimal argument, i.e., $\mathbf{A} = \nabla^2 f_i(\mathbf{w}^*)$, which allows us to capture the difference between the sequence of Hessian approximation matrices for function f_i and the Hessian $\nabla^2 f_i(\mathbf{w}^*)$ at the optimal argument. We proceed to use the result in Lemma 9 for $\mathbf{M} = \nabla^2 f_i(\mathbf{w}^*)^{-1/2}$ and $\mathbf{A} = \nabla^2 f_i(\mathbf{w}^*)$ to prove the claim in (4.26). To do so, we first need to show that the condition in (4.28) is satisfied. Note that according to the condition in Assumptions 7 and 8 we can write

$$\frac{\|\mathbf{y}_i^t - \nabla^2 f_i(\mathbf{w}^*)\mathbf{s}_i^t\|}{\|\nabla^2 f_i(\mathbf{w}^*)^{1/2}\mathbf{s}_i^t\|} \leq \frac{\tilde{L}\|\mathbf{s}_i^t\| \max\{\|\mathbf{z}_i^t - \mathbf{w}^*\|, \|\mathbf{z}_i^{t+1} - \mathbf{w}^*\|\}}{\sqrt{m}\|\mathbf{s}_i^t\|} = \frac{\tilde{L}}{\sqrt{m}}\sigma_i^t. \tag{4.31}$$

Hence, the left hand side of the condition in (4.28) for $\mathbf{M} = \nabla^2 f_i(\mathbf{w}^*)^{-1/2}$ is bounded above by

$$\frac{\|\mathbf{M}\mathbf{y}_i^t - \mathbf{M}^{-1}\mathbf{s}_i^t\|}{\|\mathbf{M}^{-1}\mathbf{s}_i^t\|} \leq \frac{\|\nabla^2 f_i(\mathbf{w}^*)^{-1/2}\|\|\mathbf{y}_i^t - \nabla^2 f_i(\mathbf{w}^*)\mathbf{s}_i^t\|}{\|\nabla^2 f_i(\mathbf{w}^*)^{1/2}\mathbf{s}_i^t\|} \leq \frac{\tilde{L}}{m}\sigma_i^t. \tag{4.32}$$

Thus, the condition in (4.28) is satisfied since $\tilde{L}\sigma_i^t/m < 1/3$. Replacing the upper bounds in (4.31) and (4.32) into the expression in (4.29) implies the claim in (4.26) with

$$\beta = \frac{\tilde{L}}{m}\sigma_i^t,\ \alpha = \frac{1-2\beta}{1-\beta^2},\ \alpha_3 = \frac{5\tilde{L}}{2m(1-\beta)},\ \alpha_4 = \frac{2(1+2\sqrt{p})\tilde{L}}{\sqrt{m}}\|\nabla^2 f_i(\mathbf{w}^*)^{-\frac{1}{2}}\|_{\mathbf{F}}, \tag{4.33}$$

and the proof is complete. ■

The result in (4.26) establishes an upper bound for the weighted norm $\|\mathbf{B}_i^{t+1} - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}}$ with respect to its previous value $\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}}$ and the sequence $\sigma_i^t := \max\{\|\mathbf{z}_i^{t+1} - \mathbf{w}^*\|, \|\mathbf{z}_i^t - \mathbf{w}^*\|\}$, when the variables are in a neighborhood of the optimal solution such that $\sigma_i^t < m/(3\tilde{L})$. Indeed, the result in (4.26) holds only for the index $i = i_t$ and for the rest of indices we have $\|\mathbf{B}_i^{t+1} - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}} = \|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}}$ simply by definition of the cyclic update. Note that if the residual sequence σ_i^t associated with f_i approaches zero, we can simplify (4.26) as

$$\|\mathbf{B}_i^{t+1} - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}} \lesssim (1 - \alpha\theta_i^{t^2})^{1/2}\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}}. \tag{4.34}$$

The equation in (4.34) implies that if θ_i^t is always strictly larger than zero, the sequence $\|\mathbf{B}_i^{t+1} - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}}$ approaches zero. If not, then the sequence θ_i^t converges to zero which implies the Dennis-Moré condition from (4.6), i.e.

$$\lim_{t\to\infty} \frac{\|(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))\mathbf{s}_i^t\|}{\|\mathbf{s}_i^t\|} = 0. \tag{4.35}$$

Therefore, under both conditions the result in (4.35) holds. This is true since the limit $\lim_{t\to\infty} \|\mathbf{B}_i^{t+1} - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}} = 0$ yields the result in (4.35).

Based on this intuition, we proceed to show that the sequence σ_i^t converges to zero for all $i = 1, \ldots, n$. To do so, we show that the sequence $\|\mathbf{z}_i^t - \mathbf{w}^*\|$ is linearly convergent for all $i = 1, \ldots, n$. To achieve this goal we first prove an upper bound for the error $\|\mathbf{w}^{t+1} - \mathbf{w}^*\|$ of IQN in the following lemma.

Lemma 10 *Consider the proposed IQN method in* (4.9)*. If the conditions in Assumptions 7 and 8 hold, then the sequence of iterates generated by IQN satisfies*

$$\|\mathbf{w}^{t+1} - \mathbf{w}^*\| \leq \frac{\tilde{L}\Gamma^t}{n} \sum_{i=1}^{N} \left\|\mathbf{z}_i^t - \mathbf{w}^*\right\|^2 + \frac{\Gamma^t}{n} \sum_{i=1}^{N} \left\| \left(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\right) \left(\mathbf{z}_i^t - \mathbf{w}^*\right) \right\|, \tag{4.36}$$

where $\Gamma^t := \|((1/n)\sum_{i=1}^{N} \mathbf{B}_i^t)^{-1}\|$.

Proof: Start by subtracting $\mathbf{w}^*$ from both sides of (4.9) to obtain

$$\mathbf{w}^{t+1} - \mathbf{w}^* = \left(\frac{1}{N}\sum_{i=1}^{N} \mathbf{B}_i^t\right)^{-1} \left(\frac{1}{N}\sum_{i=1}^{N} \mathbf{B}_i^t \mathbf{z}_i^t - \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\mathbf{z}_i^t) - \frac{1}{N}\sum_{i=1}^{N} \mathbf{B}_i^t \mathbf{w}^*\right). \tag{4.37}$$

As the gradient of f at the optimal point is the vector zero, i.e., $(1/N)\sum_{i=1}^{N} \nabla f_i(\mathbf{w}^*) = \mathbf{0}$, we can subtract $(1/N)\sum_{i=1}^{N} \nabla f_i(\mathbf{w}^*)$ from the right hand side of (4.37) and rearrange terms to obtain

$$\mathbf{w}^{t+1} - \mathbf{w}^* = \left(\frac{1}{N}\sum_{i=1}^{N} \mathbf{B}_i^t\right)^{-1} \left(\frac{1}{N}\sum_{i=1}^{N} \mathbf{B}_i^t \left(\mathbf{z}_i^t - \mathbf{w}^*\right) - \frac{1}{N}\sum_{i=1}^{N} \left(\nabla f_i(\mathbf{z}_i^t) - \nabla f_i(\mathbf{w}^*)\right)\right). \tag{4.38}$$

The expression in (4.38) relates the residual at time $t+1$ to the previous N residuals and the Hessian approximations $\mathbf{B}_i^t$. To analyze this further, we can replace the Hessian approximations $\mathbf{B}_i^t$ with the actual Hessians $\nabla^2 f_i(\mathbf{w}^*)$ and the approximation difference $\nabla^2 f_i(\mathbf{w}^*) - \mathbf{B}_i^t$. To do so, we add and subtract $(1/n)\sum_{i=1}^{N} \nabla^2 f_i(\mathbf{w}^*) \left(\mathbf{z}_i^t - \mathbf{w}^*\right)$ to the right

hand side of (4.38) and rearrange terms to obtain

$$\mathbf{w}^{t+1}-\mathbf{w}^{*}=\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_{i}^{t}\right)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\left[\nabla^{2}f_{i}(\mathbf{w}^{*})\left(\mathbf{z}_{i}^{t}-\mathbf{w}^{*}\right)-\left(\nabla f_{i}(\mathbf{z}_{i}^{t})-\nabla f_{i}(\mathbf{w}^{*})\right)\right]\right)$$
$$+\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_{i}^{t}\right)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{B}_{i}^{t}-\nabla^{2}f_{i}(\mathbf{w}^{*})\right]\left(\mathbf{z}_{i}^{t}-\mathbf{w}^{*}\right)\right). \tag{4.39}$$

We proceed to take the norms of both sides and use the triangle inequality to obtain an upper bound on the norm of the residual $\|\mathbf{w}^{t+1}-\mathbf{w}^{*}\|$,

$$\|\mathbf{w}^{t+1}-\mathbf{w}^{*}\|\leq\left\|\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_{i}^{t}\right)^{-1}\right\|\frac{1}{N}\sum_{i=1}^{N}\left\|\nabla^{2}f_{i}(\mathbf{w}^{*})\left(\mathbf{z}_{i}^{t}-\mathbf{w}^{*}\right)-\left(\nabla f_{i}(\mathbf{z}_{i}^{t})-\nabla f_{i}(\mathbf{w}^{*})\right)\right\|$$
$$+\left\|\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_{i}^{t}\right)^{-1}\right\|\frac{1}{N}\sum_{i=1}^{N}\left\|\left[\mathbf{B}_{i}^{t}-\nabla^{2}f_{i}(\mathbf{w}^{*})\right]\left(\mathbf{z}_{i}^{t}-\mathbf{w}^{*}\right)\right\|. \tag{4.40}$$

To obtain the quadratic term in (4.36) from the first term in (4.40), we use the Lipschitz continuity of the Hessians $\nabla^{2}f_{i}$ which leads to the inequality

$$\|\nabla^{2}f_{i}(\mathbf{w}^{*})\left(\mathbf{z}_{i}^{t}-\mathbf{w}^{*}\right)-\left(\nabla f_{i}(\mathbf{z}_{i}^{t})-\nabla f_{i}(\mathbf{w}^{*})\right)\|\leq\tilde{L}\left\|\mathbf{z}_{i}^{t}-\mathbf{w}^{*}\right\|^{2}. \tag{4.41}$$

Replacing the expression $\|\nabla^{2}f_{i}(\mathbf{w}^{*})\left(\mathbf{z}_{i}^{t}-\mathbf{w}^{*}\right)-\left(\nabla f_{i}(\mathbf{z}_{i}^{t})-\nabla f_{i}(\mathbf{w}^{*})\right)\|$ in (4.40) by the upper bound in (4.41), the claim in (4.36) follows. ■

Lemma 10 shows that the residual $\|\mathbf{w}^{t+1}-\mathbf{w}^{*}\|$ is bounded above by a sum of quadratic and linear terms of the last N residuals. This can eventually lead to a superlinear convergence rate by establishing the linear term converges to zero at a fast rate, leaving us with an upper bound of quadratic terms only. First, however, we establish a local linear convergence rate in the proceeding theorem to show that the sequence σ_{i}^{t} converges to zero.

Lemma 11 *Consider the proposed IQN method in* (4.9)*. If Assumptions 7 and 8 hold, then, for any $r\in(0,1)$ there are positive constants $\epsilon(r)$ and $\delta(r)$ such that if $\|\mathbf{w}^{0}-\mathbf{w}^{*}\|<\epsilon(r)$ and $\|\mathbf{B}_{i}^{0}-\nabla^{2}f_{i}(\mathbf{w}^{*})\|_{\mathbf{M}}<\delta(r)$ for $\mathbf{M}=\nabla^{2}f_{i}(\mathbf{w}^{*})^{-1/2}$ and $i=1,2,\ldots,n$, the sequence of iterates generated by IQN satisfies*

$$\|\mathbf{w}^{t}-\mathbf{w}^{*}\|\leq r^{[\frac{t-1}{n}]+1}\|\mathbf{w}^{0}-\mathbf{w}^{*}\|. \tag{4.42}$$

Moreover, the sequences of norms $\{\|\mathbf{B}_{i}^{t}\|\}$ and $\{\|(\mathbf{B}_{i}^{t})^{-1}\|\}$ are uniformly bounded.

Proof: In this proof we use some steps in the proof of [21, Theorem 3.2]. To start we use the fact that in a finite-dimensional vector space there always exists a constant $\eta>0$ such

that $\|\mathbf{A}\| \leq \eta \|\mathbf{A}\|_{\mathbf{M}}$. Consider $\gamma = 1/m$ is an upper bound for the norm $\|\nabla^2 f(\mathbf{w}^*)^{-1}\|$. Assume that $\epsilon(r) = \epsilon$ and $\delta(r) = \delta$ are chosen such that

$$(2\alpha_3\delta + \alpha_4)\frac{\epsilon}{1-r} \leq \delta \quad \text{and} \quad \gamma(1+r)[\tilde{L}\epsilon + 2\eta\delta] \leq r. \tag{4.43}$$

Based on the assumption that $\|\mathbf{B}_i^0 - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}} \leq \delta$ we can derive the upper bound $\|\mathbf{B}_i^0 - \nabla^2 f_i(\mathbf{w}^*)\| \leq \eta\delta$. This observation along with the inequality $\|\nabla^2 f_i(\mathbf{w}^*)\| \leq L$ implies that $\|\mathbf{B}_i^0\| \leq \eta\delta + L$. Therefore, we obtain $\|(1/n)\sum_{i=1}^N \mathbf{B}_i^0\| \leq \eta\delta + L$. The second inequality in (4.43) implies that $2\gamma(1+r)\eta\delta \leq r$. Based on this observation and the inequalities $\|\mathbf{B}_i^0 - \nabla^2 f_i(\mathbf{w}^*)\| \leq \eta\delta < 2\eta\delta$ and $\gamma \geq \|\nabla^2 f_i(\mathbf{w}^*)^{-1}\|$, we obtain from Banach Lemma that $\|(\mathbf{B}_i^0)^{-1}\| \leq (1+r)\gamma$. Following the same argument for the matrix $((1/n)\sum_{i=1}^N \mathbf{B}_i^0)^{-1}$ with the inequalities $\|(1/n)\sum_{i=1}^N \mathbf{B}_i^0 - (1/n)\sum_{i=1}^N \nabla^2 f_i(\mathbf{w}^*)\| \leq (1/n)\sum_{i=1}^N \|\mathbf{B}_i^0 - \nabla^2 f_i(\mathbf{w}^*)\| \leq \eta\delta$ and $\|\nabla^2 f(\mathbf{w}^*)^{-1}\| \leq \gamma$ we obtain that

$$\left\| \left(\frac{1}{N}\sum_{i=1}^N \mathbf{B}_i^0 \right)^{-1} \right\| \leq (1+r)\gamma. \tag{4.44}$$

This upper bound in conjunction with the result in (4.36) yields

$$\begin{aligned} \|\mathbf{w}^1 - \mathbf{w}^*\| &\leq (1+r)\gamma \left[\frac{\tilde{L}}{n}\sum_{i=1}^N \left\|\mathbf{z}_i^0 - \mathbf{w}^*\right\|^2 + \frac{1}{N}\sum_{i=1}^N \left\| \left[\mathbf{B}_i^0 - \nabla^2 f_i(\mathbf{w}^*)\right] \left(\mathbf{z}_i^0 - \mathbf{w}^*\right) \right\| \right] \\ &= (1+r)\gamma \left[\tilde{L}\left\|\mathbf{w}^0 - \mathbf{w}^*\right\|^2 + \frac{1}{N}\sum_{i=1}^n \left\| \left[\mathbf{B}_i^0 - \nabla^2 f_i(\mathbf{w}^*)\right] \left(\mathbf{w}^0 - \mathbf{w}^*\right) \right\| \right]. \end{aligned} \tag{4.45}$$

Considering the assumptions that $\|\mathbf{w}^0 - \mathbf{w}^*\| \leq \epsilon$ and $\|\mathbf{B}_i^0 - \nabla^2 f_i(\mathbf{w}^*)\| \leq \eta\delta < 2\eta\delta$ we can write

$$\begin{aligned} \|\mathbf{w}^1 - \mathbf{w}^*\| &\leq (1+r)\gamma[\tilde{L}\epsilon + 2\eta\delta]\|\mathbf{w}^0 - \mathbf{w}^*\| \\ &\leq r\|\mathbf{w}^0 - \mathbf{w}^*\|, \end{aligned} \tag{4.46}$$

where the second inequality follows from the second condition in (4.43). Without loss of generality, assume that $i_0 = 1$. Then, based on the result in (4.26) we obtain

$$\begin{aligned} \left\|\mathbf{B}_1^1 - \nabla^2 f_1(\mathbf{w}^*)\right\|_{\mathbf{M}} &\leq \left[(1 - \alpha\theta_1^{0\,2})^{1/2} + \alpha_3\sigma_1^0\right] \left\|\mathbf{B}_1^0 - \nabla^2 f_1(\mathbf{w}^*)\right\|_{\mathbf{M}} + \alpha_4\sigma_1^0 \\ &\leq (1 + \alpha_3\epsilon)\delta + \alpha_4\epsilon \\ &\leq \delta + 2\alpha_3\epsilon\delta + \alpha_4\epsilon \leq 2\delta. \end{aligned} \tag{4.47}$$

We proceed to the next iteration which leads to the inequality

$$\begin{aligned}\|\mathbf{w}^2 - \mathbf{w}^*\| &\leq (1+r)\gamma \left[\frac{\tilde{L}}{n}\sum_{i=1}^{N}\left\|\mathbf{z}_i^t - \mathbf{w}^*\right\|^2 + \frac{1}{N}\sum_{i=1}^{N}\left\|\left[\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\right]\left(\mathbf{z}_i^t - \mathbf{w}^*\right)\right\|\right] \\ &\leq (1+r)\gamma\left[\tilde{L}\epsilon + 2\eta\delta\right]\left(\frac{n-1}{n}\|\mathbf{w}^0 - \mathbf{w}^*\| + \frac{1}{N}\|\mathbf{w}^1 - \mathbf{w}^*\|\right) \\ &\leq r\left(\frac{n-1}{n}\|\mathbf{w}^0 - \mathbf{w}^*\| + \frac{1}{N}\|\mathbf{w}^1 - \mathbf{w}^*\|\right) \\ &\leq r\|\mathbf{w}^0 - \mathbf{w}^*\|. \end{aligned} \tag{4.48}$$

And since the updated index is $i_1 = 2$ we obtain

$$\begin{aligned}\left\|\mathbf{B}_2^2 - \nabla^2 f_2(\mathbf{w}^*)\right\|_{\mathbf{M}} &\leq \left[(1-\alpha\theta_2^{0\,2})^{1/2} + \alpha_3\sigma_2^0\right]\left\|\mathbf{B}_2^0 - \nabla^2 f_2(\mathbf{w}^*)\right\|_{\mathbf{M}} + \alpha_4\sigma_2^0 \\ &\leq (1+\alpha_3\epsilon)\delta + \alpha_4\epsilon \\ &\leq \delta + 2\alpha_3\epsilon\delta + \alpha_4\epsilon \leq 2\delta. \end{aligned} \tag{4.49}$$

With the same argument we can show that all $\left\|\mathbf{B}_t^t - \nabla^2 f_t(\mathbf{w}^*)\right\|_{\mathbf{M}} \leq 2\delta$ and $\|\mathbf{w}^t - \mathbf{w}^*\| \leq \epsilon$, for all iterates $t = 1, \ldots, n$. Moreover, we have $\|\mathbf{w}^t - \mathbf{w}^*\| \leq r\|\mathbf{w}^0 - \mathbf{w}^*\|$ for $t = 1, \ldots, n$.

Now we use the results for iterates $t = 1, \ldots, n$ as the base of our induction argument. To be more precise, let's assume that for iterates $t = jn+1, jn+2, \ldots, jn+n$ we know that the residuals are bounded above by $\|\mathbf{w}^t - \mathbf{w}^*\| \leq r^{j+1}\|\mathbf{w}^0 - \mathbf{w}^*\|$ and the Hessian approximation matrices $\mathbf{B}_i^t$ satisfy the inequalities $\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\| \leq 2\eta\delta$. Our goal is to show that for iterates $t = (j+1)n+1, (j+1)n+2, \ldots, (j+1)n+n$ the inequalities $\|\mathbf{w}^t - \mathbf{w}^*\| \leq r^{j+2}\|\mathbf{w}^0 - \mathbf{w}^*\|$ and $\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\| \leq 2\eta\delta$ hold.

Based on the inequalities $\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\| \leq 2\eta\delta$ and $\|\nabla^2 f_i(\mathbf{w}^*)^{-1}\| \leq \gamma$ we can show that for all $t = jn+1, jn+2, \ldots, jn+n$ we have

$$\left\|\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_i^t\right)^{-1}\right\| \leq (1+r)\gamma. \tag{4.50}$$

Using (4.50) and the inequality in (4.26) for the iterate $t = (j+1)n+1$, we obtain

$$\begin{aligned}\|\mathbf{w}^{(j+1)n+1} - \mathbf{w}^*\| \leq{}& (1+r)\gamma\frac{\tilde{L}}{n}\sum_{i=1}^{N}\left\|\mathbf{z}_i^{(j+1)n} - \mathbf{w}^*\right\|^2 \\ &+ (1+r)\gamma\frac{1}{N}\sum_{i=1}^{N}\left\|\left[\mathbf{B}_i^{(j+1)n} - \nabla^2 f_i(\mathbf{w}^*)\right]\left(\mathbf{z}_i^{(j+1)n} - \mathbf{w}^*\right)\right\|. \end{aligned} \tag{4.51}$$

Since the variables are updated in a cyclic fashion the set of variables $\{\mathbf{z}_i^{(j+1)n}\}_{i=1}^{i=n}$ is equal to the set $\{\mathbf{w}^{(j+1)n-i}\}_{i=0}^{i=n-1}$. By considering this relation and replacing the norms $\|[\mathbf{B}_i^{(j+1)n} - \nabla^2 f_i(\mathbf{w}^*)](\mathbf{z}_i^{(j+1)n} - \mathbf{w}^*)\|$ by their upper bounds $2\eta\delta\|\mathbf{z}_i^{(j+1)n} - \mathbf{w}^*\|$ we can simplify the right hand side of (4.51) as

$$\|\mathbf{w}^{(j+1)n+1} - \mathbf{w}^*\| \leq (1+r)\gamma \left[\frac{\tilde{L}}{n}\sum_{i=1}^{N}\left\|\mathbf{w}^{jn+i} - \mathbf{w}^*\right\|^2 + \frac{2\eta\delta}{n}\sum_{i=1}^{N}\left\|\mathbf{w}^{jn+i} - \mathbf{w}^*\right\|\right]. \quad (4.52)$$

Since $\|\mathbf{w}^{jn+i} - \mathbf{w}^*\| \leq \epsilon$ for all $j = 1, \ldots, n$, we obtain

$$\|\mathbf{w}^{(j+1)n+1} - \mathbf{w}^*\| \leq (1+r)\gamma\left[\tilde{L}\epsilon + 2\eta\delta\right]\left(\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{w}^{jn+i} - \mathbf{w}^*\right\|\right). \quad (4.53)$$

According to the second inequality in (4.43) and the assumption that for iterates $t = jn+1, jn+2, \ldots, jn+n$ we know that $\|\mathbf{w}^t - \mathbf{w}^*\| \leq r^{j+1}\|\mathbf{w}^0 - \mathbf{w}^*\|$, we can replace the right hand side of (4.53) by the following upper bound

$$\|\mathbf{w}^{(j+1)n+1} - \mathbf{w}^*\| \leq r^{j+2}\|\mathbf{w}^0 - \mathbf{w}^*\|. \quad (4.54)$$

Now we show that the updated Hessian approximation $\mathbf{B}_{i_t}^{(j+1)n+1}$ for $t = (j+1)n+1$ satisfies the inequality $\|\mathbf{B}_{i_t}^{(j+1)n+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\|_{\mathbf{M}} \leq 2\delta$. According to the result in (4.26), we can write

$$\begin{aligned}\left\|\mathbf{B}_{i_t}^{(j+1)n+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\right\|_{\mathbf{M}} - \left\|\mathbf{B}_{i_t}^{jn+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\right\|_{\mathbf{M}} \\ \leq \alpha_3\sigma_{i_t}^{jn+1}\left\|\mathbf{B}_{i_t}^{jn+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\right\|_{\mathbf{M}} + \alpha_4\sigma_{i_t}^{jn+1}. \quad (4.55)\end{aligned}$$

Now observe that $\sigma_{i_t}^{jn+1} = \max\{\|\mathbf{w}^{(j+1)n+1} - \mathbf{w}^*\|, \|\mathbf{w}^{jn+1} - \mathbf{w}^*\|\}$ is bounded above by $r^{j+1}\|\mathbf{w}^0 - \mathbf{w}^*\|$. Applying this substitution into (4.55) and considering the conditions $\|\mathbf{B}_{i_t}^{jn+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\|_{\mathbf{M}} \leq 2\delta$ and $\|\mathbf{w}^0 - \mathbf{w}^*\| \leq \epsilon$ lead to the inequality

$$\left\|\mathbf{B}_{i_t}^{(j+1)n+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\right\|_{\mathbf{M}} - \left\|\mathbf{B}_{i_t}^{jn+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\right\|_{\mathbf{M}} \leq r^{j+1}\epsilon(2\delta\alpha_3 + \alpha_4). \quad (4.56)$$

By writing the expression in (4.56) for previous iterations and using a recursive logic we obtain that

$$\left\|\mathbf{B}_{i_t}^{(j+1)n+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\right\|_{\mathbf{M}} - \left\|\mathbf{B}_{i_t}^{0} - \nabla^2 f_{i_t}(\mathbf{w}^*)\right\|_{\mathbf{M}} \leq \epsilon(2\delta\alpha_3 + \alpha_4)\frac{1}{1-r}. \quad (4.57)$$

Based on the first inequality in (4.43), the right hand side of (4.57) is bounded above by δ. Moreover, the norm $\|\mathbf{B}_{i_t}^0 - \nabla^2 f_{i_t}(\mathbf{w}^*)\|_{\mathbf{M}}$ is also upper bounded by δ. These two bounds

imply that

$$\left\|\mathbf{B}_{i_t}^{(j+1)n+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\right\|_{\mathbf{M}} \leq 2\delta, \tag{4.58}$$

and consequently $\|\mathbf{B}_{i_t}^{(j+1)n+1} - \nabla^2 f_{i_t}(\mathbf{w}^*)\| \leq 2\eta\delta$. By following the steps from (4.51) to (4.58), we can show for all iterates $t = (j+1)n+1, (j+1)n+2, \ldots, (j+1)n+n$ the inequalities $\|\mathbf{w}^t - \mathbf{w}^*\| \leq r^{j+2}\|\mathbf{w}^0 - \mathbf{w}^*\|$ and $\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\| \leq 2\eta\delta$ hold. The induction proof is complete and (4.42) holds. Moreover, the inequality $\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\| \leq 2\eta\delta$ holds for all i and steps t. Hence, the norms $\|\mathbf{B}_i^t\|$ and $\|(\mathbf{B}_i^t)^{-1}\|$, and consequently $\|(1/n)\sum_{i=1}^N \mathbf{B}_i^t\|$ and $\|((1/n)\sum_{i=1}^N \mathbf{B}_i^t)^{-1}\|$ are uniformly bounded. ■

The result in Lemma 11 shows that the sequence of iterates generated by IQN has a local linear convergence rate after each pass over all functions. Consequently, we obtain that the i-th residual sequence σ_i^t is linearly convergent for all i. Note that Lemma 11 can be considered as an extension of Theorem 3.2 in [21] for incremental settings. Following the arguments in (4.34) and (4.35), we use the summability of the sequence σ_i^t along with the result in Lemma 8 to prove Dennis-Moré condition for all functions f_i.

Proposition 3 *Consider the proposed IQN method in* (4.9)*. Assume that the hypotheses in Lemmata 8 and 11 are satisfied. Then, for all $i = 1, \ldots, n$ it holds,*

$$\lim_{t\to\infty} \frac{\|(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))\mathbf{s}_i^t\|}{\|\mathbf{s}_i^t\|} = 0. \tag{4.59}$$

Proof: According to the result in Lemma 11, we can show that the sequence of errors $\sigma_i^t = \max\{\|\mathbf{z}_i^{t+1} - \mathbf{w}^*\|, \|\mathbf{z}_i^t - \mathbf{w}^*\|\}$ is summable for all i. To do so, consider the sum of the sequence σ_i^t which is upper bounded by

$$\sum_{t=0}^{\infty} \sigma_i^t = \sum_{t=0}^{\infty} \max\{\|\mathbf{z}_i^{t+1} - \mathbf{w}^*\|, \|\mathbf{z}_i^t - \mathbf{w}^*\|\} \leq \sum_{t=0}^{\infty} \|\mathbf{z}_i^{t+1} - \mathbf{w}^*\| + \sum_{t=0}^{\infty} \|\mathbf{z}_i^t - \mathbf{w}^*\| \tag{4.60}$$

Note that the last time that the index i is chosen before time t should be in the set $\{t-1, \ldots, t-n\}$. This observation in association with the result in (4.42) implies that

$$\sum_{t=0}^{\infty} \sigma_i^t \leq 2\sum_{t=0}^{\infty} r^{[\frac{t-n-1}{n}]+1}\|\mathbf{w}^0 - \mathbf{w}^*\| = 2\sum_{t=0}^{\infty} r^{[\frac{t-1}{n}]}\|\mathbf{w}^0 - \mathbf{w}^*\| \tag{4.61}$$

Simplifying the sum in the right hand side of (4.61) yields

$$\sum_{t=0}^{\infty} \sigma_i^t \leq \frac{2\|\mathbf{w}^0 - \mathbf{w}^*\|}{r} + 2n\|\mathbf{w}^0 - \mathbf{w}^*\| \sum_{t=0}^{\infty} r^t < \infty. \tag{4.62}$$

Thus, the sequence σ_i^t is summable for all $i = 1, \ldots, n$. To complete the proof we use the following result from Lemma 3.3 in [32].

Lemma 12 *Let $\{\phi^t\}$ and $\{\delta^t\}$ be sequences of nonnegative numbers such that*

$$\phi^{t+1} \leq (1 + \delta^t)\phi^t + \delta^t \quad \textit{and} \quad \sum_{k=1}^{\infty} \delta^t < \infty. \tag{4.63}$$

Then, the sequence $\{\phi^t\}$ converges.

Considering the results in Lemmata 8 and 12, and the fact that σ_i^t is summable as shown in(4.62), we obtain that the sequence $\left\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\right\|_{\mathbf{M}}$ for $\mathbf{M} := \nabla^2 f_i(\mathbf{w}^*)^{-1/2}$ is convergent and the following limit exists

$$\lim_{k\to\infty} \|\nabla^2 f_i(\mathbf{w}^*)^{-1/2}\ \mathbf{B}_i^t\ \nabla^2 f_i(\mathbf{w}^*)^{-1/2}\ -\ \mathbf{I}\|_{\mathbf{F}} = l \tag{4.64}$$

where l is a nonnegative constant. Moreover, following the proof of Theorem 3.4 in [32] we can show that

$$\begin{aligned}\alpha(\theta_i^t)^2\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}} \leq \|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}} - \|\mathbf{B}_i^{t+1} - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}} \\ + \sigma_i^t(\alpha_3\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}} + \alpha_4),\end{aligned} \tag{4.65}$$

and, therefore, summing both sides implies,

$$\sum_{t=0}^{\infty}(\theta_i^t)^2\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}} < \infty \tag{4.66}$$

Replacing θ_i^t in (4.66) by its definition in (4.27) results in

$$\sum_{t=0}^{\infty} \frac{\|\mathbf{M}(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))\mathbf{s}_i^t\|^2}{\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}}\|\mathbf{M}^{-1}\mathbf{s}_i^t\|^2} < \infty \tag{4.67}$$

Since the norm $\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\|_{\mathbf{M}}$ is upper bounded and the eigenvalues of the matrix $\mathbf{M} = \nabla^2 f_i(\mathbf{w}^*)^{-1/2}$ are uniformly lower and upper bounded, we conclude from the result in (4.67) that

$$\lim_{t\to\infty} \frac{\|(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))\mathbf{s}_i^t\|^2}{\|\mathbf{s}_i^t\|^2} = 0, \tag{4.68}$$

which yields the claim in (4.59). ■

The statement in Proposition 3 indicates that for each function f_i the Dennis-Moré condition holds. In the tradition quasi-Newton methods the Dennis-Moré condition is sufficient to show that the method is superlinearly convergent. However, the same argument does not hold for the proposed IQN method, since we can't recover the Dennis-Moré condition for the global objective function f from the result in Proposition 3. In other words, the result in (4.59) does not imply the limit in (4.6) required in the superlinear convergence analysis of quasi-Newton methods. Therefore, here we pursue a different approach and seek to prove that the linear terms $(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))(\mathbf{z}_i^t - \mathbf{w}^*)$ in (4.36) converge to zero at a superlinear rate, i.e., for all i we can write $lim_{t\to\infty}\|(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))(\mathbf{z}_i^t - \mathbf{w}^*)\|/\|\mathbf{z}_i^t - \mathbf{w}^*\| = 0$. If we establish this result, it follows from the result in Lemma 10 that the sequence of residuals $\|\mathbf{w}^t - \mathbf{w}^*\|$ converges to zero superlinearly.

We continue the analysis of the proposed IQN method by establishing a generalized limit property that follows from the Dennis-Moré criterion in (4.6). In the following lemma, we leverage the local linear convergence of the iterates $\mathbf{w}^t$ to show that that the vector $\mathbf{z}_i^t - \mathbf{w}^*$ lies in the null space of $\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)$ as t approaches infinity.

Lemma 13 *Consider the proposed IQN method in* (4.9)*. Assume that the hypotheses in Lemmata 8 and 11 are satisfied. As t approaches infinity, the following holds for all i,*

$$\lim_{t\to\infty} \frac{\|(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))(\mathbf{z}_i^t - \mathbf{w}^*)\|}{\|\mathbf{z}_i^t - \mathbf{w}^*\|} = 0. \tag{4.69}$$

Proof: Consider the sets of variable variations $\mathcal{S}_1 = \{\mathbf{s}_i^{t+n\tau}\}_{\tau=0}^{\tau=T}$ and $\mathcal{S}_2 = \{\mathbf{s}_i^{t+n\tau}\}_{\tau=0}^{\tau=\infty}$. It is trivial to show that $\mathbf{z}_i^t - \mathbf{w}^*$ is in the span of the set $\mathcal{S}_2$, since the sequences of variables $\mathbf{w}^t$ and $\mathbf{z}_i^t$ converge to $\mathbf{w}^*$ and we can write $\mathbf{w}^* - \mathbf{z}_i^t = \sum_{\tau=0}^{\infty} \mathbf{s}_i^{t+n\tau}$. We proceed to show that the vector $\mathbf{z}_i^t - \mathbf{w}^*$ is also in the span of the set $\mathcal{S}_1$ when T is sufficiently large. To do so, we use a contradiction argument. Let's assume that the vector $\mathbf{z}_i^t - \mathbf{w}^*$ does not lie in the span of the set $\mathcal{S}_1$, and, therefore, it can be decomposed as the sum of two non-zero vectors given by

$$\mathbf{z}_i^t - \mathbf{w}^* = \mathbf{v}_{\|}^t + \mathbf{v}_{\perp}^t, \tag{4.70}$$

where $\mathbf{v}_{\|}^t$ lies in the span of $\mathcal{S}_1$ and $\mathbf{v}_{\perp}^t$ is orthogonal to the span of $\mathcal{S}_1$. Since we assume that $\mathbf{z}_i^t - \mathbf{w}^*$ does not lie in the span of $\mathcal{S}_1$, we obtain that $\mathbf{z}_i^{t+nT} - \mathbf{w}^*$ also does not lie in this span, since $\mathbf{z}_i^{t+nT} - \mathbf{w}^*$ can be written as the sum $\mathbf{z}_i^{t+nT} - \mathbf{w}^* = \mathbf{z}_i^t - \mathbf{w}^* + \sum_{\tau=0}^{T} \mathbf{s}_i^{t+n\tau}$. These observations imply that we can also decompose the vector $\mathbf{z}_i^{t+nT} - \mathbf{w}^*$ as

$$\mathbf{z}_i^{t+nT} - \mathbf{w}^* = \mathbf{v}_{\|}^{t+nT} + \mathbf{v}_{\perp}^{t+nT}, \tag{4.71}$$

where $\mathbf{v}_{\parallel}^{t+nT}$ lies in the span of $\mathcal{S}_1$ and $\mathbf{v}_{\perp}^{t+nT}$ is orthogonal to the span of $\mathcal{S}_1$. Moreover, we obtain that $\mathbf{v}_{\perp}^{t+nT}$ is equal to $\mathbf{v}_{\perp}^{t}$, i.e.,

$$\mathbf{v}_{\perp}^{t+nT} = \mathbf{v}_{\perp}^{t}. \tag{4.72}$$

This is true since $\mathbf{z}_i^{t+nT} - \mathbf{w}^*$ can be written as the sum of $\mathbf{z}_i^t - \mathbf{w}^*$ and a group of vectors that lie in the span of $\mathcal{S}_1$. We assume that the norm $\|\mathbf{v}_{\perp}^{t+nT}\| = \|\mathbf{v}_{\perp}^{t}\| = \epsilon$ where $\epsilon > 0$ is a strictly positive constant. According to the linear convergence of the sequence $\|\mathbf{w}^t - \mathbf{w}^*\|$ in Lemma 11 we know that

$$\|\mathbf{z}_i^{t+nT} - \mathbf{w}^*\| \leq r^{[\frac{t+nT-1}{n}]+1}\|\mathbf{w}^0 - \mathbf{w}^*\| \leq r^T\|\mathbf{w}^0 - \mathbf{w}^*\| \tag{4.73}$$

If we pick large enough T such that $r^T\|\mathbf{w}^0 - \mathbf{w}^*\| < \epsilon$, then we obtain $\|\mathbf{z}_i^{t+nT} - \mathbf{w}^*\| < \epsilon$ which contradicts the assumption $\|\mathbf{v}_{\perp}^{t}\| = \epsilon$. Thus, we obtain that the vector $\mathbf{z}_i^t - \mathbf{w}^*$ is also in the span of set $\mathcal{S}_1$.

Since the vector $\mathbf{z}_i^t - \mathbf{w}^*$ is in the span of $\mathcal{S}_1$, we can write the normalized vector $(\mathbf{z}_i^t - \mathbf{w}^*)/\|\mathbf{z}_i^t - \mathbf{w}^*\|$ as a linear combination of the set of normalized vectors $\{\mathbf{s}_i^{t+n\tau}/\|\mathbf{s}_i^{t+n\tau}\|\}_{\tau=0}^{\tau=T}$. This property allows to write

$$\begin{aligned}\lim_{t\to\infty} \frac{\|(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))(\mathbf{z}_i^t - \mathbf{w}^*)\|}{\|\mathbf{z}_i^t - \mathbf{w}^*\|} &= \lim_{t\to\infty} \left\|(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))\frac{(\mathbf{z}_i^t - \mathbf{w}^*)}{\|\mathbf{z}_i^t - \mathbf{w}^*\|}\right\| \\ &= \lim_{t\to\infty} \left\|(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))\sum_{\tau=0}^{T} a_\tau \frac{\mathbf{s}_i^{t+n\tau}}{\|\mathbf{s}_i^{t+n\tau}\|}\right\|, \end{aligned} \tag{4.74}$$

where a_τ is coefficient of the vector $\mathbf{s}_i^{t+n\tau}$ when we write $(\mathbf{z}_i^t - \mathbf{w}^*)/\|\mathbf{z}_i^t - \mathbf{w}^*\|$ as the linear combination of the normalized vectors $\{\mathbf{s}_i^{t+n\tau}/\|\mathbf{s}_i^{t+n\tau}\|\}_{\tau=0}^{\tau=T}$. Now since the index of the difference $\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)$ does not match with the descent directions $\mathbf{s}_i^t + n\tau$. We add and subtract the term $\mathbf{B}_i^{t+n\tau}$ to the expression $\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)$ and use the triangle inequality to write

$$\begin{aligned}&\lim_{t\to\infty} \frac{\|(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*))(\mathbf{z}_i^t - \mathbf{w}^*)\|}{\|\mathbf{z}_i^t - \mathbf{w}^*\|} \\ &\leq \lim_{t\to\infty} \left\|\sum_{\tau=0}^{T} a_\tau \frac{(\mathbf{B}_i^{t+n\tau} - \nabla^2 f_i(\mathbf{w}^*))\mathbf{s}_i^{t+n\tau}}{\|\mathbf{s}_i^{t+n\tau}\|}\right\| + \left\|\sum_{\tau=0}^{T} a_\tau \frac{(\mathbf{B}_i^t - \mathbf{B}_i^{t+n\tau})\mathbf{s}_i^{t+n\tau}}{\|\mathbf{s}_i^{t+n\tau}\|}\right\|. \end{aligned} \tag{4.75}$$

We first simplify the first limit in the right hand side of (4.75). Using the Cauchy-Schwarz inequality and the result in Proposition 3 we can write

$$\lim_{t\to\infty}\left\|\sum_{\tau=0}^{T}a_\tau\frac{(\mathbf{B}_i^{t+n\tau}-\nabla^2 f_i(\mathbf{w}^*))\mathbf{s}_i^{t+n\tau}}{\|\mathbf{s}_i^{t+n\tau}\|}\right\| \leq \lim_{t\to\infty}\sum_{\tau=0}^{T}a_\tau\left\|\frac{(\mathbf{B}_i^{t+n\tau}-\nabla^2 f_i(\mathbf{w}^*))\mathbf{s}_i^{t+n\tau}}{\|\mathbf{s}_i^{t+n\tau}\|}\right\|$$
$$=\sum_{\tau=0}^{T}a_\tau\lim_{t\to\infty}\left\|\frac{(\mathbf{B}_i^{t+n\tau}-\nabla^2 f_i(\mathbf{w}^*))\mathbf{s}_i^{t+n\tau}}{\|\mathbf{s}_i^{t+n\tau}\|}\right\|=0. \tag{4.76}$$

Based on the results in (4.75) and (4.76), to prove the claim in (4.69) it remains to show

$$\lim_{t\to\infty}\left\|\sum_{\tau=0}^{T}a_\tau\frac{(\mathbf{B}_i^{t}-\mathbf{B}_i^{t+n\tau})\mathbf{s}_i^{t+n\tau}}{\|\mathbf{s}_i^{t+n\tau}\|}\right\|=0. \tag{4.77}$$

To reach this goal, we first study the limit of the difference between two consecutive update Hessian approximation matrices $\lim_{t\to\infty}\|\mathbf{B}_i^t-\mathbf{B}_i^{t+n}\|$. Note that if we set $\mathbf{A}=\mathbf{B}_i^t$ in (4.29), we obtain that

$$\|\mathbf{B}_i^{t+n}-\mathbf{B}_i^t\|_{\mathbf{M}}\leq\alpha_2\frac{\|\mathbf{y}_i^t-\mathbf{B}_i^t\mathbf{s}_i^t\|}{\|\mathbf{M}^{-1}\mathbf{s}_i^t\|}. \tag{4.78}$$

where $\mathbf{M}=(\nabla^2 f_i(\mathbf{w}^*))^{-1/2}$. By adding and subtracting the term $\nabla^2 f_i(\mathbf{w}^*)\mathbf{s}_i^t$ and using the result in (4.59), we can show that the difference $\|\mathbf{B}_i^{t+n}-\mathbf{B}_i^t\|_{\mathbf{M}}$ approaches zero asymptotically. In particular,

$$\lim_{t\to\infty}\|\mathbf{B}_i^{t+n}-\mathbf{B}_i^t\|_{\mathbf{M}}\leq\alpha_2\lim_{t\to\infty}\frac{\|\mathbf{y}_i^t-\mathbf{B}_i^t\mathbf{s}_i^t\|}{\|\mathbf{M}^{-1}\mathbf{s}_i^t\|}$$
$$\leq\alpha_2\lim_{t\to\infty}\frac{\|\mathbf{y}_i^t-\nabla^2 f_i(\mathbf{w}^*)\mathbf{s}_i^t\|}{\|\mathbf{M}^{-1}\mathbf{s}_i^t\|}+\alpha_2\lim_{t\to\infty}\frac{\|(\nabla^2 f_i(\mathbf{w}^*)-\mathbf{B}_i^t)\mathbf{s}_i^t\|}{\|\mathbf{M}^{-1}\mathbf{s}_i^t\|}. \tag{4.79}$$

Since $\|\mathbf{y}_i^t-\nabla^2 f_i(\mathbf{w}^*)\mathbf{s}_i^t\|$ is bounded above by $\tilde{L}\|\mathbf{s}_i^t\|\max\{\|\mathbf{z}_i^t-\mathbf{w}^*\|,\|\mathbf{z}_i^{t+1}-\mathbf{w}^*\|\}$ and the eigenvalues of the matrix $\mathbf{M}$ are uniformly bounded we obtain that the first limit in the right hand side of (4.79) converges to zero. Further, the result in (4.59) shows that the second limit in the right hand side of (4.79) also converges to zero. Therefore,

$$\lim_{t\to\infty}\|\mathbf{B}_i^{t+n}-\mathbf{B}_i^t\|_{\mathbf{M}}=0. \tag{4.80}$$

Following the same argument we can show that for any two consecutive Hessian approximation matrices the difference approaches zero asymptotically. Thus, we obtain

$$\begin{aligned}\lim_{t\to\infty}\left\|\mathbf{B}_i^t-\mathbf{B}_i^{t+n\tau}\right\|_{\mathbf{M}} &\leq \lim_{t\to\infty}\left\|\sum_{u=0}^{\tau-1}\left(\mathbf{B}_i^{t+nu}-\mathbf{B}_i^{t+n(u+1)}\right)\right\|_{\mathbf{M}}\\ &\leq \sum_{u=0}^{\tau-1}\lim_{t\to\infty}\left\|\mathbf{B}_i^{t+nu}-\mathbf{B}_i^{t+n(u+1)}\right\|_{\mathbf{M}}=0. \end{aligned} \tag{4.81}$$

Observing the result in (4.81) we can show that

$$\begin{aligned}\lim_{t\to\infty}\left\|\sum_{\tau=0}^{T}a_\tau\frac{(\mathbf{B}_i^t-\mathbf{B}_i^{t+n\tau})\mathbf{s}_i^{t+n\tau}}{\|\mathbf{s}_i^{t+n\tau}\|}\right\| &\leq \sum_{\tau=0}^{T}a_\tau\lim_{t\to\infty}\left\|\frac{(\mathbf{B}_i^t-\mathbf{B}_i^{t+n\tau})\mathbf{s}_i^{t+n\tau}}{\|\mathbf{s}_i^{t+n\tau}\|}\right\|\\ &\leq \sum_{\tau=0}^{T}a_\tau\lim_{t\to\infty}\left\|\mathbf{B}_i^t-\mathbf{B}_i^{t+n\tau}\right\|=0. \end{aligned} \tag{4.82}$$

Therefore, the result in (4.77) holds. The claim in (4.69) follows by combining the results in (4.75), (4.76), and (4.77). ■

The result in Lemma 13 can thus be used in conjunction with Lemma 10 to show that the residual $\|\mathbf{w}^{t+1}-\mathbf{w}^*\|$ is bounded by a sum of quadratic terms of previous residuals and a term that converges to zero superlinearly. This result leads us to the following result, namely the local superlinear convergence of the sequence of residuals with respect to the average sequence, stated in the following theorem.

Theorem 5 *Consider the proposed IQN method in* (4.9)*. Suppose that the conditions in the hypotheses of Lemmata 8 and 11 are valid. Then, the sequence of residuals* $\|\mathbf{w}^t-\mathbf{w}^*\|$ *satisfies*

$$\lim_{t\to\infty}\frac{\|\mathbf{w}^t-\mathbf{w}^*\|}{\frac{1}{N}(\|\mathbf{w}^{t-1}-\mathbf{w}^*\|+\cdots+\|\mathbf{w}^{t-n}-\mathbf{w}^*\|)}=0. \tag{4.83}$$

Proof: The result in Lemma 10 implies

$$\|\mathbf{w}^{t+1}-\mathbf{w}^*\|\leq\frac{\tilde{L}\Gamma^t}{n}\sum_{i=1}^{N}\left\|\mathbf{z}_i^t-\mathbf{w}^*\right\|^2+\frac{\Gamma^t}{n}\sum_{i=1}^{N}\left\|\left(\mathbf{B}_i^t-\nabla^2 f_i(\mathbf{w}^*)\right)\left(\mathbf{z}_i^t-\mathbf{w}^*\right)\right\|. \tag{4.84}$$

Divide both sides of (4.84) by $(1/N)\sum_{i=1}^{N}\left\|\mathbf{z}_i^t-\mathbf{w}^*\right\|$ to obtain

$$\frac{\|\mathbf{w}^{t+1}-\mathbf{w}^*\|}{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{z}_i^t-\mathbf{w}^*\|}\leq\tilde{L}\Gamma^t\sum_{i=1}^{N}\frac{\left\|\mathbf{z}_i^t-\mathbf{w}^*\right\|^2}{\sum_{i=1}^{N}\|\mathbf{z}_i^t-\mathbf{w}^*\|}+\Gamma^t\sum_{i=1}^{N}\frac{\left\|\left(\mathbf{B}_i^t-\nabla^2 f_i(\mathbf{w}^*)\right)\left(\mathbf{z}_i^t-\mathbf{w}^*\right)\right\|}{\sum_{i=1}^{N}\|\mathbf{z}_i^t-\mathbf{w}^*\|} \tag{4.85}$$

Since the error $\|\mathbf{z}_i^t - \mathbf{w}^*\|$ is a lower bound for the sum of errors $\sum_{i=1}^N \|\mathbf{z}_i^t - \mathbf{w}^*\|$, we can replace $\|\mathbf{z}_i^t - \mathbf{w}^*\|$ for $\sum_{i=1}^N \|\mathbf{z}_i^t - \mathbf{w}^*\|$ into (4.85) which implies

$$\begin{aligned}\frac{\|\mathbf{w}^{t+1} - \mathbf{w}^*\|}{\frac{1}{N}\sum_{i=1}^N \|\mathbf{z}_i^t - \mathbf{w}^*\|} &\leq \tilde{L}\Gamma^t \sum_{i=1}^N \frac{\left\|\mathbf{z}_i^t - \mathbf{w}^*\right\|^2}{\|\mathbf{z}_i^t - \mathbf{w}^*\|} + \Gamma^t \sum_{i=1}^N \frac{\left\|\left(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\right)\left(\mathbf{z}_i^t - \mathbf{w}^*\right)\right\|}{\|\mathbf{z}_i^t - \mathbf{w}^*\|} \\ &= \tilde{L}\Gamma^t \sum_{i=1}^N \left\|\mathbf{z}_i^t - \mathbf{w}^*\right\| + \Gamma^t \sum_{i=1}^N \frac{\left\|\left(\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{w}^*)\right)\left(\mathbf{z}_i^t - \mathbf{w}^*\right)\right\|}{\|\mathbf{z}_i^t - \mathbf{w}^*\|}. \quad (4.86)\end{aligned}$$

Since Γ^t is bounded above, computing the limit of both sides in (4.86) yields

$$\lim_{t\to\infty} \frac{\|\mathbf{w}^{t+1} - \mathbf{w}^*\|}{\frac{1}{N}\sum_{i=1}^N \|\mathbf{z}_i^t - \mathbf{w}^*\|} = 0. \quad (4.87)$$

The result in (4.87) in association with the simplification for the sum $\sum_{i=1}^N \left\|\mathbf{z}_i^t - \mathbf{w}^*\right\| = \sum_{i=0}^{n-1} \left\|\mathbf{w}^{t-i} - \mathbf{w}^*\right\|$ leads to the claim in (4.83). ■

The result in (4.83) shows a mean-superlinear convergence rate for the sequence of iterates generated by IQN. To be more precise, it shows that the ratio that captures the error at step t divided by the average of last N errors converges to zero. This is not equivalent to the classic Q-superlinear convergence for full-batch quasi-Newton methods, i.e., $\lim_{t\to\infty} \|\mathbf{w}^{t+1} - \mathbf{w}^*\|/\|\mathbf{w}^t - \mathbf{w}^*\| = 0$. Although Q-superlinear convergence of the residuals $\|\mathbf{w}^t - \mathbf{w}^*\|$ is not provable, we can show that there exists a subsequence of the sequence $\|\mathbf{w}^t - \mathbf{w}^*\|$ that converges to zero superlinearly. In addition, there exists a superlinearly convergent sequence that is an upper bound for the original sequence of errors $\|\mathbf{w}^t - \mathbf{w}^*\|$. We formalize these results in the following theorem.

Theorem 6 *Consider the proposed IQN method in* (4.9)*. Suppose that the conditions in the hypotheses of Lemmata 8 and 11 are valid. Then, there exists a subsequence of* $\|\mathbf{w}^t - \mathbf{w}^*\|$ *that converges to zero superlinearly. Moreover, there exists a sequence* ζ^t *such that* $\|\mathbf{w}^t - \mathbf{w}^*\| \leq \zeta^t$ *for all* $t \geq 0$*, and the sequence* ζ^t *converges to zero at a superlinear rate, i.e.,*

$$\lim_{t\to\infty} \frac{\zeta^{t+1}}{\zeta^t} = 0. \quad (4.88)$$

Proof: Consider the definition of the sequence $\tilde{\mathbf{w}}^t = \text{argmax}_{u\in\{tn,\ldots,tn+n-1\}}\{\|\mathbf{w}^u - \mathbf{w}^*\|\}$ which is a subsequence of the sequence $\{\mathbf{w}^t\}_{t=0}^\infty$. Our goal is to show this subsequence converges superlinearly to $\mathbf{w}^*$, i.e., $\lim_{t\to\infty} \frac{\|\tilde{\mathbf{w}}^{t+1} - \mathbf{w}^*\|}{\|\tilde{\mathbf{w}}^t - \mathbf{w}^*\|} = 0$. To do so, first note that the result in Theorem 5 implies that

$$\lim_{t\to\infty} \frac{\|\mathbf{w}^t - \mathbf{w}^*\|}{\max\{\|\mathbf{w}^{t-1} - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{t-n} - \mathbf{w}^*\|\}} = 0, \quad (4.89)$$

which follows from the inequality $\max\{\|\mathbf{w}^{t-1} - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{t-n} - \mathbf{w}^*\|\} \geq (1/n)(\|\mathbf{w}^{t-1} - \mathbf{w}^*\| + \cdots + \|\mathbf{w}^{t-n} - \mathbf{w}^*\|)$. Based on the limit in (4.89), there exists a large enough t_0 such that for all $t \geq t_0$ the following inequality holds,

$$\|\mathbf{w}^t - \mathbf{w}^*\| < \max\{\|\mathbf{w}^{t-1} - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{t-n} - \mathbf{w}^*\|\}. \tag{4.90}$$

Combining the inequality in (4.90) with the inequalities $\|\mathbf{w}^{t-i} - \mathbf{w}^*\| \leq \max\{\|\mathbf{w}^{t-1} - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{t-n} - \mathbf{w}^*\|\}$ for $i = 1, \ldots, n-1$ yields

$$\max\{\|\mathbf{w}^t - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{t-n+1} - \mathbf{w}^*\|\} \leq \max\{\|\mathbf{w}^{t-1} - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{t-n} - \mathbf{w}^*\|\}, \tag{4.91}$$

and consequently we can generalize this result to obtain

$$\max\{\|\mathbf{w}^t - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{t-n+1} - \mathbf{w}^*\|\} \leq \max\{\|\mathbf{w}^{t-\tau} - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{t-\tau-n+1} - \mathbf{w}^*\|\}, \tag{4.92}$$

for any positive integer τ such that $t - \tau \geq t_0$.

We use the result in (4.92) to build a superlinearly convergent subsequence of the residuals sequence $\|\mathbf{w}^t - \mathbf{w}^*\|$. If we define $\mathbf{w}^{tn+u_t^*}$ as the iterate that has the largest error among the iterates in the $t+1$-th pass, i.e.,

$$\mathbf{w}^{tn+u_t^*} = \underset{u \in \{tn, \ldots, tn+n-1\}}{\operatorname{argmax}} \{\|\mathbf{w}^u - \mathbf{w}^*\|\}, \tag{4.93}$$

then it follows that $\tilde{\mathbf{w}}^t = \mathbf{w}^{tn+u_t^*}$, where $u_t^* \in \{0, 1, \ldots, n-1\}$. Moreover, we obtain

$$\begin{aligned} \frac{\|\tilde{\mathbf{w}}^{t+1} - \mathbf{w}^*\|}{\|\tilde{\mathbf{w}}^t - \mathbf{w}^*\|} &= \frac{\|\mathbf{w}^{(t+1)n+u_{t+1}^*} - \mathbf{w}^*\|}{\max\{\|\mathbf{w}^{tn} - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{tn+n-1} - \mathbf{w}^*\|\}} \\ &\leq \frac{\|\mathbf{w}^{tn+n+u_{t+1}^*} - \mathbf{w}^*\|}{\max\{\|\mathbf{w}^{tn+u_{t+1}^*-1} - \mathbf{w}^*\|, \ldots, \|\mathbf{w}^{tn+n+u_{t+1}^*-1} - \mathbf{w}^*\|\}}. \end{aligned} \tag{4.94}$$

The equality follows from the definition of the iterate $\tilde{\mathbf{w}}^t$ and the definition in (4.93), and the inequality holds because of the result in (4.92). Considering the result in (4.89), computing the limit of both sides leads to the conclusion that the sequence $\|\tilde{\mathbf{w}}^t - \mathbf{w}^*\|$ is superlinearly convergent. In other words, we obtain that the subsequence $\{\|\mathbf{w}^{tn+u_t^*} - \mathbf{w}^*\|\}_{t=0}^{t=\infty}$ superlinearly converges to zero.

Let's define the sequence q^t such that $q^{kn} = \cdots = q^{kn+n-1} = \|\tilde{\mathbf{w}}^k - \mathbf{w}^*\|$ for $k = 0, 1, 2, \ldots$, which means that the value of the sequence q^t is fixed for each pass and is equal to the max error of the corresponding pass. Therefore, it is trivial to show that q^t is always larger than or equal to $\|\mathbf{w}^t - \mathbf{w}^*\|$, i.e., $\|\mathbf{w}^t - \mathbf{w}^*\| \leq q^t$ for all $t \geq 0$. Now define the

sequence ζ^t such that $\zeta^t = q^t$ for $t = 0, \ldots, n-1$, and for $t \geq n$

$$\zeta^{kn+i} = q^{kn-1} \left(\frac{q^{kn+n-1}}{q^{kn-1}} \right)^{\frac{i+1}{n}}, \qquad \text{for} \quad i = 0, \ldots, n-1, \quad k \geq 1. \tag{4.95}$$

According to this definition we can verify that ζ^t is an upper bound for the sequence q^t and, consequently, an upper bound for the sequence of errors $\|\mathbf{w}^t - \mathbf{w}^*\|$. Based on the definition of the sequence ζ^t in (4.95), the ratio ζ^{t+1}/ζ^t is given by $(q^{\lfloor \frac{t+1}{n} \rfloor n+n-1}/q^{\lfloor \frac{t+1}{n} \rfloor n-1})^{1/n}$. This simplification in association with the definitions of the sequences $\|\mathbf{w}^t - \mathbf{w}^*\|$ and $\|\tilde{\mathbf{w}}^t - \mathbf{w}^*\|$ implies that

$$\lim_{t\to\infty} \frac{\zeta^{t+1}}{\zeta^t} = \lim_{t\to\infty} \left(\frac{q^{\lfloor \frac{t+1}{n} \rfloor n+n-1}}{q^{\lfloor \frac{t+1}{n} \rfloor n-1}} \right)^{\frac{1}{N}} = \lim_{t\to\infty} \left(\frac{\|\tilde{\mathbf{w}}^{\lfloor \frac{t+1}{n} \rfloor} - \mathbf{w}^*\|}{\|\tilde{\mathbf{w}}^{\lfloor \frac{t+1}{n} \rfloor - 1} - \mathbf{w}^*\|} \right)^{\frac{1}{N}} = 0, \tag{4.96}$$

which leads to the claim in (4.88). ■

The first result in Theorem 6 states that although the whole sequence $\|\mathbf{w}^t - \mathbf{w}^*\|$ is not necessarily superlinearly convergent, there exists a subsequence of the sequence $\|\mathbf{w}^t - \mathbf{w}^*\|$ that converges at a superlinear rate. The second claim in Theorem 6 establishes R-superlinear convergence rate of the whole sequence $\|\mathbf{w}^t - \mathbf{w}^*\|$. In other words, it guarantees that $\|\mathbf{w}^t - \mathbf{w}^*\|$ is upper bounded by a superlinearly convergent sequence.

4.5 Numerical results

We proceed by simulating the performance of IQN on a variety of machine learning problems on both artificial and real datasets. We compare the performance of IQN against a collection of well known first order stochastic and incremental algorithms—namely SAG, SAGA, and IAG. To begin, we look at a simple quadratic program, also equivalent to the solution of linear least squares estimation problem. Consider the objective function to be minimized,

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}} f(\mathbf{w}) := \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \mathbf{w}^T \mathbf{A}_i \mathbf{w} + \mathbf{b}_i^T \mathbf{w}. \tag{4.97}$$

We generate $\mathbf{A}_i \in \mathbb{R}^{p\times p}$ as a random positive definite matrix and $\mathbf{b}_i \in \mathbb{R}^p$ as a random vector for all i. In particular we set the matrices $\mathbf{A}_i := \operatorname{diag}\{\mathbf{a}_i\}$ and generate random vectors $\mathbf{a}_i$ with the first $p/2$ elements chosen from $[1, 10^{\xi/2}]$ and last $p/2$ elements chosen from $[10^{-\xi/2}, 1]$. The parameter ξ is used to manually set the condition number for the quadratic program in (4.97), ranging from $\xi = 1$ (i.e. small condition number 10^2) and $\xi = 2$ (i.e. large condition number 10^4). The vectors $\mathbf{b}_i$ are chosen uniformly and randomly from the box $[0, 10^3]^p$. The variable dimension is set to be $p = 10$ and number of functions

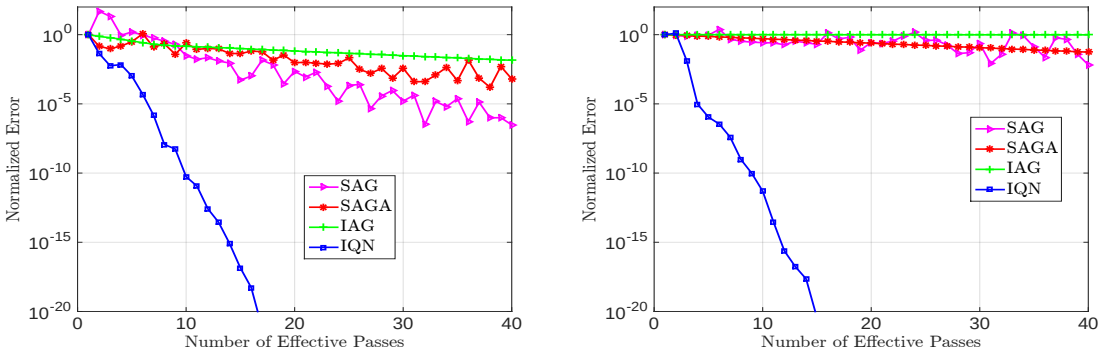


Figure 4.2: Convergence results of proposed IQN method in comparison to SAG, SAGA, and IAG. In the left image, we present a sample convergence path of the normalized error on the quadratic program with a small condition number. In the right image, we show the convergence path for the quadratic program with a large condition number. In all cases, IQN provides significant improvement over first order methods, with the difference increasing for larger condition number.

$N = 1000$. Given that we focus on local convergence, we use a constant step size of $\eta = 1$ for the proposed IQN method while choosing the largest step size allowable by the other methods to converge.

In Figure 4.2 we present a simulation of the convergence path of the normalized error $\|\mathbf{w}^t - \mathbf{w}^*\|/\|\mathbf{w}^0 - \mathbf{w}^*\|$ for the quadratic program. In the the left image, we show a sample simulation path for all methods on the quadratic problem with a small condition number. Step sizes of $\eta = 5 \times 10^{-5}$, $\eta = 10^{-4}$ and $\eta = 10^{-6}$ were used for SAG, SAGA, and IAG, respectively. These step sizes are tuned to compare the best performance of these methods with IQN. The proposed method reaches a error of 10^{-10} after 10 passes through the data. Alternatively, SAGA achieves the same error of 10^{-5} after 30 passes, while SAG and IAG do not reach 10^{-5} after 40 passes.

In the right image of Figure 4.2, we repeat the same simulation but with larger condition number. In this case, SAG uses stepsize $\eta = 2 \times 10^{-4}$ while others remain the same. Observe that while the performance of IQN does not degrade with larger condition number, the first order methods all suffer large degradation. SAG, SAGA, and IAG reach after 40 passes a normalized error of 6.5×10^{-3}, 5.5×10^{-2}, and 9.6×10^{-1}, respectively. It can be seen that IQN significantly outperforms the first order method for both condition number sizes, with the outperformance increasing for larger condition number. This is an expected result, as first order methods often do not perform well for ill conditioned problems.

4.5.1 Logistic regression

We proceed to numerically evaluate the performance of IQN relative to existing methods on the classification of handwritten digits in the MNIST database [50]. In particular, we solve the binary logistic regression problem. A logistic regression takes as inputs N training

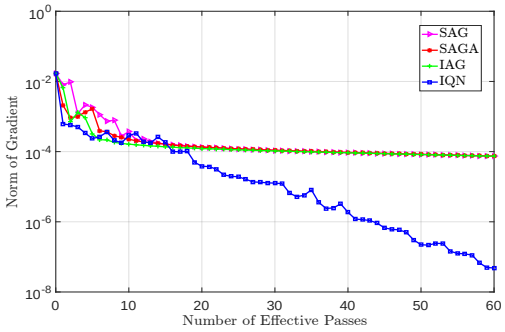


Figure 4.3: Convergence results for a sample convergence path for the logistic regression problem on classifying handwritten digits. IQN substantially outperforms the first order methods.

feature vectors $\mathbf{u}_i \in \mathbb{R}^p$ with associated labels $v_i \in \{-1, 1\}$ and outputs a linear classifier $\mathbf{w}$ to predict the label of unknown feature vectors. For the digit classification problem, each feature vector $\mathbf{u}_i$ represents a vectorized image and label v_i its label as one of two digits. We evaluate for any training sample i the probability of a label $v_i = 1$ given image $\mathbf{u}_i$ as $P(v = 1|\mathbf{u}) = 1/(1 + \exp(-\mathbf{u}^T\mathbf{w}))$. The classifier $\mathbf{w}$ is chosen to be the vector which maximizes the log likelihood across all N samples. Given N images $\mathbf{u}_i$ with associated labels v_i, the optimization problem for logistic regression is written as

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}} f(\mathbf{w}) := \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} \log[1 + \exp(-v_i\mathbf{u}_i^T\mathbf{w})], \tag{4.98}$$

where the first term is a regularization term parametrized by $\lambda \geq 0$.

For our simulations we select from the MNIST dataset $N = 1000$ images with dimension $p = 784$ labelled as one of the digits "0" or "8' and fix the regularization parameter as $\lambda = 1/N$ and stepsize $\eta = 0.01$ for all first order methods. In Figure 4.3 we present the convergence path of IQN relative to existing methods in terms of the norm of the gradient. As in the case of the quadratic program, the IQN performs all gradient-based methods. IQN reaches a gradient magnitude of 4.8×10^{-8} after 60 passes through the data while the SAGA reaches only a magnitude of 7.4×10^{-5} (all other methods perform even worse). Further note that while the first order methods begin to level out after 60 passes, the IQN method continues to descend. These results demonstrate the effectiveness of IQN on a practical machine learning problem with real world data.

Part II

Decentralized Methods

Chapter 5

Network Newton methods

5.1 Context and background

Distributed optimization algorithms are used to solve the problem of minimizing a global cost function over a set of nodes in situations where the objective function is defined as a sum of local functions. To be more precise, consider a variable $\mathbf{w} \in \mathbb{R}^p$ and a connected network containing V agents each of which has access to a local function $f_v : \mathbb{R}^p \to \mathbb{R}$. The agents cooperate in minimizing the aggregate cost function $f : \mathbb{R}^p \to \mathbb{R}$ taking values $f(\mathbf{w}) := \sum_{v=1}^{V} f_v(\mathbf{w})$. I.e., agents cooperate in solving the problem

$$\mathbf{w}^* \;:=\; \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}}\, f(\mathbf{w}) \;=\; \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{v=1}^{V} f_v(\mathbf{w}). \tag{5.1}$$

Problems of this form arise often in, e.g., decentralized control systems [25, 56], wireless systems [94, 96], sensor networks [46, 93, 103], and large scale machine learning [6, 57, 118].

As explained in Chapter 1, the empirical risk minimization (ERM) problem can be solved using distributed optimization methods by splitting samples among nodes (processors) in the network. This observation implies that ERM can be written in the form of the problem formulation in (5.1) if we consider $f_v(\mathbf{w})$ as the loss associated to the samples of node v. In the second part of the thesis, we focus on developing algorithms for solving the distributed optimization problem in (5.1) which can be exploited to solve large-scale ERM problems.

There are different algorithms to solve (5.1) in a distributed manner. The most popular choices are decentralized gradient descent (DGD) [44, 80, 111, 126], distributed implementations of the alternating direction method of multipliers [19,27,77,103,112], and decentralized dual averaging [33, 119]. Although there are substantial differences between them, these methods can be generically abstracted as combinations of local descent steps followed by variable exchanges and averaging of information among neighbors. A feature common to all

of these algorithms is the slow convergence rate in ill-conditioned problems since they operate on first order information only. This is not surprising because gradient descent methods in centralized settings where the aggregate function gradient is available at a single server have the same difficulties in problems with skewed curvature.

This issue is addressed in centralized optimization by Newton's method that uses second order information to determine a descent direction adapted to the objective's curvature. In general, second order methods are not available in distributed settings because distributed approximations of Newton steps are difficult to devise. In the particular case of flow optimization problems, these approximations are possible when operating in the dual domain and have led to the development of the accelerated dual descent methods [123, 128]. As would be expected, these methods result in large reductions of convergence times.

Our goal is to develop approximate Newton's methods to solve (5.1) in distributed settings where agents have access to their local functions only and exchange variables with neighboring agents. We do so by introducing Network Newton (NN), a method that relies on distributed approximations of Newton steps for the global cost function f to accelerate convergence of DGD. We begin the chapter with an alternative formulation of (5.1) and a brief discussion of DGD (Section 5.2). We then introduce a reinterpretation of DGD as an algorithm that utilizes gradient descent to solve a penalized version of (5.1) in lieu of the original optimization problem (Section 5.2.1). This reinterpretation explains convergence of DGD to a neighborhood of $\mathbf{w}^*$. The volume of this neighborhood is given by the relative weight of the penalty function and the original objective which is controlled by a penalty coefficient.

If gradient descent on the penalized function finds an approximate solution to the original problem, the same solution can be found with a much smaller number of iterations by using Newton's method. Alas, distributed computation of Newton steps requires global communication between all nodes in the network and is therefore impractical (Section 5.3). To resolve this issue we approximate the Newton step of the penalized objective function by truncating the Taylor series of the exact Newton step (Section 5.3.1). This approximation results in a family of methods indexed by the number of terms of the Taylor expansion that are kept in the approximation. The method that results from keeping K of these terms is termed NN-K. A fundamental observation here is that the Hessian of the penalized function has a sparsity structure that is the same sparsity pattern of the graph. Thus, when computing terms in the Hessian inverse expansion, the first order term is as sparse as the graph, the second term is as sparse as the two hop neighborhood, and, in general, the k-th term is as sparse as the k-hop neighborhood of the graph. Thus, implementation of the NN-K method requires aggregating information from K hops away. Increasing K makes NN-K arbitrarily close to Newton's method at the cost of increasing the communication overhead

of each iteration. We point out that the same Taylor series is used in the development of the ADD algorithms, but this is done to solve a network utility maximization problem in the dual domain [128]. The Taylor expansion is utilized here to solve a consensus optimization problem in the primal domain.

Convergence of NN-K to the optimal argument of the penalized objective is established (Section 5.4). We do so by establishing several auxiliary bounds on the eigenvalues of the matrices involved in the definition of the method (Propositions 4-6 and Lemma 16). We show that a measure of the error between the Hessian inverse approximation utilized by NN-K and the actual inverse Hessian decays exponentially with the method index K. This exponential decrease hints that using a small value of K should suffice in practice. Convergence is formally claimed in Theorem 7 that shows the convergence rate is at least linear. It follows from this convergence analysis that larger penalty coefficients result in faster convergence that comes at the cost of increasing the distance between the optimal solutions of the original and penalized objectives.

We also study the convergence rate of the NN method as an approximation of Newton's method (Section 5.4.1). We show that for all iterations except the first few, a weighted gradient norm associated with NN-K iterates follows a decreasing path akin to the path that would be followed by Newton iterates (Lemma 17). The only difference between these residual paths is that the NN-K path contains a term that captures the error of the Hessian inverse approximation. Leveraging this similarity, it is possible to show that the rate of convergence is quadratic in a specific interval whose length depends on the order K of the selected network Newton method (Theorem 8). Existence of this quadratic convergence phase explains why NN-K methods converge faster than DGD – as we observe in experiments. It is also worth remarking that the error in the Hessian inverse approximation can be made arbitrarily small by increasing the method's order K and, as a consequence, the quadratic phase can be made arbitrarily large.

We wrap up the chapter with numerical analyses (Section 5.6). We first demonstrate the advantages of NN-K relative to alternative primal and dual methods for the minimization of a family of quadratic objective functions (Section 5.6.1). Then, we study the effect of objective function condition number and show that the NN method outperforms first-order alternatives significantly in ill-conditioned problems (Section 5.6.2). Further, we study the effect of network topology on the performance of NN (Section 5.6.3). Moreover, we compare the convergence rate of NN in theory and practice to show the tightness of the bounds in this chapter (Section 5.6.4).

Notation. Vectors are written as $\mathbf{w} \in \mathbb{R}^p$ and matrices as $\mathbf{A} \in \mathbb{R}^{p \times p}$. The null space of matrix $\mathbf{A}$ is denoted by null($\mathbf{A}$) and the span of a vector by span($\mathbf{w}$). We use $\|\mathbf{w}\|$ and $\|\mathbf{A}\|$ to denote the Euclidean norm of vector $\mathbf{w}$ and matrix $\mathbf{A}$, respectively. The gradient of

a function $f(\mathbf{w})$ is denoted as $\nabla f(\mathbf{w})$ and the Hessian matrix is denoted as $\nabla^2 f(\mathbf{w})$. The v-th largest eigenvalue of matrix $\mathbf{A}$ is denoted by $\mu_v(\mathbf{A})$.

5.2 Distributed gradient descent

The network that connects the V agents is assumed connected, symmetric, and specified by the neighborhoods $\mathcal{N}_v$ that contain the list of nodes that can communicate with v for $v = 1, \ldots, V$. In problem (5.1) agent v has access to the local cost $f_v(\mathbf{w})$ and agents cooperate to minimize the global cost $f(\mathbf{w})$. This specification is more naturally formulated by an alternative representation of (5.1) in which node v selects a local decision vector $\mathbf{w}_v \in \mathbb{R}^p$. Nodes then try to achieve the minimum of their local objective functions $f_v(\mathbf{w}_v)$, while keeping their variables equal to the variables $\mathbf{w}_u$ of neighbors $u \in \mathcal{N}_v$. This alternative formulation can be written as

$$\begin{aligned} \{\mathbf{w}_v^*\}_{v=1}^V \;:=\; & \underset{\{\mathbf{w}_v\}_{i=1}^V}{\operatorname{argmin}} \sum_{v=1}^V f_v(\mathbf{w}_v), \\ & \text{s.t. } \mathbf{w}_v = \mathbf{w}_u, \quad \text{for all } v, u \in \mathcal{N}_v. \end{aligned} \tag{5.2}$$

Since the network is connected, the constraints $\mathbf{w}_v = \mathbf{w}_u$ for all v and $u \in \mathcal{N}_v$ imply that (5.1) and (5.2) are equivalent and we have $\mathbf{w}_v^* = \mathbf{w}^*$ for all v. This must be the case because for a connected network the constraints $\mathbf{w}_v = \mathbf{w}_u$ for all v and $u \in \mathcal{N}_v$ collapse the feasible space of (5.2) to a hyperplane in which all local variables are equal. When all variables are equal, the objectives in (5.1) and (5.2) coincide and so do their optima.

DGD is an established distributed method to solve (5.2) which relies on the introduction of nonnegative weights $w_{vu} \geq 0$ that are null if and only if $u \notin \mathcal{N}_v \cup \{v\}$ – the use of time varying weights w_{vu} is common in DGD implementations but not done here; see, e.g., [80]. Letting $t \in \mathbb{N}$ be a discrete time index and α a given stepsize, DGD is defined by the recursion

$$\mathbf{w}_{v,t+1} = \sum_{u=1}^V w_{vu} \mathbf{w}_{u,t} - \alpha \nabla f_v(\mathbf{w}_{v,t}), \qquad v = 1, \ldots, V. \tag{5.3}$$

Since $w_{vu} = 0$ when $u \neq v$ and $u \notin \mathcal{N}_v$, it follows from (5.3) that each agent v updates its variable $\mathbf{w}_v$ by performing an average over the estimates $\mathbf{w}_{u,t}$ of its neighbors $u \in \mathcal{N}_v$ and its own estimate $\mathbf{w}_{v,t}$, and descending through the negative local gradient $-\nabla f_v(\mathbf{w}_{v,t})$.

The weights in (5.3) cannot be arbitrary. To express conditions on the set of allowable weights define the matrix $\mathbf{W} \in \mathbb{R}^{V \times V}$ with entries w_{uv}. We require the weights to be symmetric, i.e., $w_{vu} = w_{uv}$ for all v, u, and such that the weights of a given node sum up to 1, i.e., $\sum_{u=1}^V w_{vu} = 1$ for all v. If the weights sum up to 1 we must have $\mathbf{W}\mathbf{1} = \mathbf{1}$ which implies that $\mathbf{I} - \mathbf{W}$ is rank deficient. It is also customary to require the rank of $\mathbf{I} - \mathbf{W}$ to

be exactly equal to $n-1$ so that the null space of $\mathbf{I}-\mathbf{W}$ is $\text{null}(\mathbf{I}-\mathbf{W}) = \text{span}(\mathbf{1})$. We therefore have the following three restrictions on the matrix $\mathbf{W}$,

$$\mathbf{W}^T = \mathbf{W}, \quad \mathbf{W}\mathbf{1} = \mathbf{1}, \quad \text{null}(\mathbf{I}-\mathbf{W}) = \text{span}(\mathbf{1}). \tag{5.4}$$

If the conditions in (5.4) are true, it is possible to show that (5.3) approaches the solution of (5.1) in the sense that $\mathbf{w}_{v,t} \approx \mathbf{w}^*$ for all v and large t, [80]. The accepted interpretation of why (5.3) converges is that nodes are gradient descending towards their local minima because of the term $-\alpha\nabla f_v(\mathbf{w}_{v,t})$ but also perform an average of neighboring variables $\sum_{j=1}^n w_{uv}\mathbf{w}_{u,t}$. This latter consensus operation drives the agents to agreement. In the following section we show that (5.3) can be alternatively interpreted as a penalty method.

5.2.1 Penalty method interpretation

It is illuminating to define matrices and vectors so as to rewrite (5.3) as a single equation. To do so define the vectors $\mathbf{y} := [\mathbf{w}_1; \ldots; \mathbf{w}_V]$ and $\mathbf{h}(\mathbf{y}) := [\nabla f_1(\mathbf{w}_1); \ldots; \nabla f_V(\mathbf{w}_V)]$. Vector $\mathbf{y} \in \mathbb{R}^{Vp}$ concatenates the local vectors $\mathbf{w}_v$, and the vector $\mathbf{h}(\mathbf{y}) \in \mathbb{R}^{Vp}$ concatenates the gradients of the local functions f_v taken with respect to the local variable $\mathbf{w}_v$. Notice that $\mathbf{h}(\mathbf{y})$ is *not* the gradient of $f(\mathbf{w})$ and that a vector $\mathbf{y}$ with $\mathbf{h}(\mathbf{y}) = \mathbf{0}$ does *not* necessarily solve (5.1). To solve (5.1) we need to have $\mathbf{w}_v = \mathbf{w}_u$ for all v and u with $\sum_{v=1}^V \nabla f_v(\mathbf{w}_v) = \mathbf{0}$. In any event, to rewrite (5.3) we also define the matrix $\mathbf{Z} := \mathbf{W} \otimes \mathbf{I} \in \mathbb{R}^{Vp \times Vp}$ as the Kronecker product of the weight matrix $\mathbf{W} \in \mathbb{R}^{V \times V}$ and the identity matrix $\mathbf{I} \in \mathbb{R}^{p \times p}$. It is then ready to see that (5.3) is equivalent to

$$\mathbf{y}_{t+1} = \mathbf{Z}\mathbf{y}_t - \alpha\mathbf{h}(\mathbf{y}_t) = \mathbf{y}_t - \big[(\mathbf{I}-\mathbf{Z})\mathbf{y}_t + \alpha\mathbf{h}(\mathbf{y}_t)\big], \tag{5.5}$$

where in the second equality we added and subtracted $\mathbf{y}_t$ and regrouped terms. Inspection of (5.5) reveals that the DGD update formula at step t is equivalent to a (regular) gradient descent algorithm being used to solve the program

$$\mathbf{y}^* := \text{argmin}\, F(\mathbf{y}) := \min \frac{1}{2}\mathbf{y}^T(\mathbf{I}-\mathbf{Z})\mathbf{y} + \alpha\sum_{v=1}^V f_v(\mathbf{w}_v). \tag{5.6}$$

This interpretation has been previously used in [42, 44] to design a Nesterov type acceleration of DGD. Indeed, given the definition of the function $F(\mathbf{y}) := (1/2)\mathbf{y}^T(\mathbf{I}-\mathbf{Z})\ \mathbf{y} + \alpha\sum_{v=1}^V f_v(\mathbf{w}_v)$ it follows that the gradient $\nabla F(\mathbf{y}_t)$ is given by

$$\mathbf{g}_t \ := \ \nabla F(\mathbf{y}_t) \ = \ (\mathbf{I}-\mathbf{Z})\mathbf{y}_t + \alpha\mathbf{h}(\mathbf{y}_t). \tag{5.7}$$

Using (5.7) we rewrite (5.5) as $\mathbf{y}_{t+1} = \mathbf{y}_t - \mathbf{g}_t$ and conclude that DGD descends along the negative gradient of $F(\mathbf{y})$ with unit stepsize. The expression in (5.3) is just a distributed implementation of gradient descent that uses the gradient in (5.7). To confirm that this is true, observe that the vth element of the gradient $\mathbf{g}_t = [\mathbf{g}_{1,t}; \ldots; \mathbf{g}_{V,t}]$ is given by

$$\mathbf{g}_{v,t} = (1 - w_{vv})\mathbf{w}_{v,t} - \sum_{u \in \mathcal{N}_v} w_{vu}\mathbf{w}_{u,t} + \alpha \nabla f_v(\mathbf{w}_{v,t}). \tag{5.8}$$

The gradient descent iteration $\mathbf{y}_{t+1} = \mathbf{y}_t - \mathbf{g}_t$ is then equivalent to (5.3) if we entrust node v with the implementation of the descent $\mathbf{w}_{v,t+1} = \mathbf{w}_{v,t} - \mathbf{g}_{v,t}$, where, we recall, $\mathbf{w}_{v,t}$ and $\mathbf{w}_{v,t+1}$ are the vth components of the vectors $\mathbf{y}_t$ and $\mathbf{y}_{t+1}$. Observe that the local gradient component $\mathbf{g}_{v,t}$ can be computed using local information and the $\mathbf{w}_{u,t}$ iterates of its neighbors $u \in \mathcal{N}_v$. This is as it should be, because the descent $\mathbf{w}_{v,t+1} = \mathbf{w}_{v,t} - \mathbf{g}_{v,t}$ is equivalent to (5.3).

Is it a good idea to descend on $F(\mathbf{y})$ to solve (5.1)? To some extent. Since we know that the null space of $\mathbf{I} - \mathbf{W}$ is $\text{null}(\mathbf{I} - \mathbf{W}) = \text{span}(\mathbf{1})$ and that $\mathbf{Z} = \mathbf{W} \otimes \mathbf{I}$ we know that the null space of $\mathbf{I} - \mathbf{Z}$ is the set of consensus vectors, i.e., $\text{null}(\mathbf{I} - \mathbf{Z}) = \left\{\mathbf{y} = [\mathbf{w}_1; \ldots; \mathbf{w}_V] \,\middle|\, \mathbf{w}_1 = \cdots = \mathbf{w}_V\right\}$. Thus, $(\mathbf{I} - \mathbf{Z})\mathbf{y} = \mathbf{0}$ holds if and only if $\mathbf{w}_1 = \cdots = \mathbf{w}_V$. Since the matrix $\mathbf{I} - \mathbf{Z}$ is positive semidefinite and symmetric, the same is true of the square root matrix $(\mathbf{I} - \mathbf{Z})^{1/2}$. Therefore, the optimization problem in (5.2) is equivalent to the optimization problem

$$\tilde{\mathbf{y}}^* := \underset{\mathbf{w}}{\text{argmin}} \sum_{v=1}^{V} f_v(\mathbf{w}_v), \qquad \text{s.t. } (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{y} = \mathbf{0}. \tag{5.9}$$

Indeed, for $\mathbf{y} = [\mathbf{w}_1; \ldots; \mathbf{w}_V]$ to be feasible in (5.9) we must have $\mathbf{w}_1 = \cdots = \mathbf{w}_V$. This is the same constraint imposed in (5.2) from where it follows that we must have $\tilde{\mathbf{y}}^* = [\mathbf{w}_1^*; \ldots; \mathbf{w}_V^*]$ with $\mathbf{w}_v^* = \mathbf{w}^*$ for all v. The unconstrained minimization in (5.6) is a penalty version of (5.9). The penalty function associated with the constraint $(\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{y} = \mathbf{0}$ is the squared norm $(1/2)\|(\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{y}\|^2$ and the corresponding penalty coefficient is $1/\alpha$. Inasmuch as the penalty coefficient $1/\alpha$ is sufficiently large, the optimal arguments $\mathbf{y}^*$ and $\tilde{\mathbf{y}}^*$ are not too far apart.

The reinterpretation of (5.3) as a penalty method demonstrates that DGD is an algorithm that finds the optimal solution of (5.6), not (5.9) or its equivalent original formulations in (5.1) and (5.2). Using a fixed α the distance between $\mathbf{y}^*$ and $\tilde{\mathbf{y}}^*$ is of order $O(\alpha)$, [126]. To solve (5.9) we need to introduce a rule to progressively decrease α. In the following section we exploit the reinterpretation of (5.5) as a method to minimize (5.6) to propose an approximate Newton algorithm that can be implemented in a distributed manner.

5.3 Network Newton

Instead of solving (5.6) with a gradient descent method as in DGD, we can solve (5.6) using Newton's method. To implement Newton's method we need to compute the Hessian $\mathbf{H}_t := \nabla^2 F(\mathbf{y}_t)$ of F evaluated at $\mathbf{y}_t$ so as to determine the Newton step $\mathbf{d}_t := -\mathbf{H}_t^{-1}\mathbf{g}_t$. Start by differentiating twice in (5.6) in order to write $\mathbf{H}_t$ as

$$\mathbf{H}_t := \nabla^2 F(\mathbf{y}_t) = \mathbf{I} - \mathbf{Z} + \alpha \mathbf{G}_t, \tag{5.10}$$

where $\mathbf{G}_t \in \mathbb{R}^{Vp \times Vp}$ is a block diagonal matrix formed by blocks $\mathbf{G}_{vv,t} \in \mathbb{R}^{p \times p}$ defined as

$$\mathbf{G}_{vv,t} = \nabla^2 f_v(\mathbf{w}_{v,t}). \tag{5.11}$$

It follows from (5.10) and (5.11) that the Hessian $\mathbf{H}_t$ is block sparse with blocks $\mathbf{H}_{vu,t} \in \mathbb{R}^{p \times p}$ having the sparsity pattern of $\mathbf{Z}$, which is the sparsity pattern of the graph. The diagonal blocks are of the form $\mathbf{H}_{vv,t} = (1 - w_{vv})\mathbf{I} + \alpha \nabla^2 f_v(\mathbf{w}_{v,t})$ and the off diagonal blocks are not null only when $u \in \mathcal{N}_v$ in which case $\mathbf{H}_{vu,t} = w_{uv}\mathbf{I}$.

While the Hessian $\mathbf{H}_t$ is sparse, the inverse $\mathbf{H}_t$ is not. It is the latter that we need to compute the Newton step $\mathbf{d}_t := \mathbf{H}_t^{-1}\mathbf{g}_t$. To overcome this problem we split the diagonal and off diagonal blocks of $\mathbf{H}_t$ and rely on a Taylor's expansion of the inverse – This splitting technique is inspired from the Taylor's expansion used in [128]. To be precise, write $\mathbf{H}_t = \mathbf{D}_t - \mathbf{B}$ where the matrix $\mathbf{D}_t$ is defined as

$$\mathbf{D}_t := \alpha \mathbf{G}_t + 2\ (\mathbf{I} - \text{diag}(\mathbf{Z})) := \alpha \mathbf{G}_t + 2\ (\mathbf{I} - \mathbf{Z}_d), \tag{5.12}$$

where in the second equality we defined $\mathbf{Z}_d := \text{diag}(\mathbf{Z})$ for future reference. Since the diagonal weights must be $w_{vv} < 1$, the matrix $\mathbf{I} - \mathbf{Z}_d$ is positive definite. The same is true of the block diagonal matrix $\mathbf{G}_t$ because the local functions are assumed strongly convex. Therefore, the matrix $\mathbf{D}_t$ is block diagonal and positive definite. The vth diagonal block $\mathbf{D}_{vv,t} \in \mathbb{R}^p$ of $\mathbf{D}_t$ can be computed and stored by node v as $\mathbf{D}_{vv,t} = \alpha \nabla^2 f_v(\mathbf{w}_{v,t}) + 2(1 - w_{vv})\mathbf{I}$. To have $\mathbf{H}_t = \mathbf{D}_t - \mathbf{B}$ we must define $\mathbf{B} := \mathbf{D}_t - \mathbf{H}_t$. Considering the definitions of $\mathbf{H}_t$ and $\mathbf{D}_t$ in (5.10) and (5.12), it follows that

$$\mathbf{B} = \mathbf{I} - 2\mathbf{Z}_d + \mathbf{Z}. \tag{5.13}$$

Note that $\mathbf{B}$ is time-invariant and depends on the weight matrix $\mathbf{Z}$ only. As in the case of the Hessian $\mathbf{H}_t$, the matrix $\mathbf{B}$ is block sparse with blocks $\mathbf{B}_{vu} \in \mathbb{R}^{p \times p}$ having the sparsity pattern of $\mathbf{Z}$, which is the sparsity pattern of the graph. Node v can compute the diagonal blocks $\mathbf{B}_{vv} = (1 - w_{vv})\mathbf{I}$ and the off diagonal blocks $\mathbf{B}_{vu} = w_{vu}\mathbf{I}$ using information about

its own and neighbors' weights.

Proceed now to factor $\mathbf{D}_t^{1/2}$ from both sides of the splitting relationship to write $\mathbf{H}_t = \mathbf{D}_t^{1/2}(\mathbf{I} - \mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2})\mathbf{D}_t^{1/2}$. When we consider the Hessian inverse $\mathbf{H}^{-1}$, we can use the Taylor series $(\mathbf{I} - \mathbf{X})^{-1} = \sum_{j=0}^{\infty} \mathbf{X}^j$ with $\mathbf{X} = \mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ to write

$$\mathbf{H}_t^{-1} = \mathbf{D}_t^{-1/2} \sum_{k=0}^{\infty} \left(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\right)^k \mathbf{D}_t^{-1/2}. \tag{5.14}$$

The sum in (5.14) converges if the absolute value of all the eigenvalues of the matrix $\mathbf{D}^{-1/2}\mathbf{B}\mathbf{D}^{-1/2}$ are strictly less than 1. For the time being we assume this to be the case but we will prove that this is true in Section 5.4. When the series converge, we can use truncations of this series to define approximations to the Newton step as we explain in the following section.

Remark 4 The Hessian decomposition $\mathbf{H}_t = \mathbf{D}_t - \mathbf{B}$ with the matrices $\mathbf{D}_t$ and $\mathbf{B}$ in (5.12) and (5.13), respectively, is not the only valid decomposition that we can use for Network Newton. Any decomposition of the form $\mathbf{H}_t = \mathbf{D}_t \pm \mathbf{B}_t$ is valid if $\mathbf{D}_t$ is positive definite and the eigenvalues of the matrix $\mathbf{D}_t^{-1/2}\mathbf{B}_t\mathbf{D}_t^{-1/2}$ are in the interval $(-1, 1)$. An example alternative decomposition is given by the matrices $\mathbf{D}_t = \alpha\mathbf{G}_t$ and $\mathbf{B} = \mathbf{I} - \mathbf{Z}$. This decomposition has the advantage of separating the effects of the function in $\mathbf{D}_t$ and the effects of the network in $\mathbf{B}$. The decomposition in (5.12) and (5.13) exhibits faster convergence of the series in (5.14) because the matrix $\mathbf{D}_t$ in (5.12) accumulates more weight in the diagonal than the matrix $\mathbf{D}_t = \alpha\mathbf{G}_t$. The study of alternative decompositions is beyond the scope of this chapter.

5.3.1 Distributed approximations of the Newton step

Network Newton (NN) is defined as a family of algorithms that rely on truncations of the series in (5.14). The Kth member of this family, NN-K, considers the first $K+1$ terms of the series to define the approximate Hessian inverse

$$\hat{\mathbf{H}}_t^{(K)^{-1}} := \mathbf{D}_t^{-1/2} \sum_{k=0}^{K} \left(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\right)^k \mathbf{D}_t^{-1/2}. \tag{5.15}$$

NN-K uses the approximate Hessian $\hat{\mathbf{H}}_t^{(K)^{-1}}$ as a curvature correction matrix that is used in lieu of the exact Hessian inverse $\mathbf{H}^{-1}$ to estimate the Newton step. I.e., instead of descending along the Newton step $\mathbf{d}_t := -\mathbf{H}_t^{-1}\mathbf{g}_t$ we descend along the NN-K step $\mathbf{d}_t^{(K)} := -\hat{\mathbf{H}}_t^{(K)^{-1}}\mathbf{g}_t$ as an approximation of $\mathbf{d}_t$. Using the explicit expression for $\hat{\mathbf{H}}_t^{(K)^{-1}}$ in (5.15) we write the

NN-K step as

$$\mathbf{d}_t^{(K)} = -\ \mathbf{D}_t^{-1/2}\ \sum_{k=0}^{K}\left(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\right)^k\ \mathbf{D}_t^{-1/2}\ \mathbf{g}_t, \tag{5.16}$$

where, we recall, $\mathbf{g}_t$ as the gradient of the function $F(\mathbf{y})$ defined in (5.7). The NN-K update can then be written as

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \epsilon\ \mathbf{d}_t^{(K)}, \tag{5.17}$$

where ϵ is a properly selected stepsize – see Theorem 7 for specific conditions. The algorithm defined by recursive application of (5.17) can be implemented in a distributed manner because the truncated series in (5.15) has a local structure controlled by the parameter K. To explain this statement better define the components $\mathbf{d}_{v,t}^{(K)} \in \mathbb{R}^p$ of the NN-K step $\mathbf{d}_t^{(K)} = [\mathbf{d}_{1,t}^{(K)}; \ldots; \mathbf{d}_{V,t}^{(K)}]$. A distributed implementation of (5.17) requires that node v computes $\mathbf{d}_{v,t}^{(K)}$ so as to implement the local descent $\mathbf{w}_{v,t+1} = \mathbf{w}_{v,t} + \epsilon\mathbf{d}_{v,t}^{(K)}$. The key observation here is that the step component $\mathbf{d}_{v,t}^{(K)}$ can indeed be computed through local operations. Specifically, begin by noting that as per the definition of the NN-K descent direction in (5.16) the sequence of NN descent directions satisfies

$$\mathbf{d}_t^{(k+1)} = \mathbf{D}_t^{-1}\mathbf{B}\mathbf{d}_t^{(k)} - \mathbf{D}_t^{-1}\mathbf{g}_t = \mathbf{D}_t^{-1}\left(\mathbf{B}\mathbf{d}_t^{(k)} - \mathbf{g}_t\right). \tag{5.18}$$

Since the matrix $\mathbf{B}$ has the sparsity pattern of the graph, this recursion can be decomposed into local components

$$\mathbf{d}_{v,t}^{(k+1)} = \mathbf{D}_{vv,t}^{-1}\Bigg(\sum_{u\in\mathcal{N}_v\cup\{v\}} \mathbf{B}_{vu}\mathbf{d}_{u,t}^{(k)} - \mathbf{g}_{v,t}\Bigg), \tag{5.19}$$

The matrix $\mathbf{D}_{vv,t} = \alpha\nabla^2 f_v(\mathbf{w}_{v,t}) + 2(1-w_{vv})\mathbf{I}$ is stored and computed at node v. The gradient component $\mathbf{g}_{v,t} = (1-w_{vv})\mathbf{w}_{v,t} - \sum_{u\in\mathcal{N}_v} w_{vu}\mathbf{w}_{u,t} + \alpha\nabla f_v(\mathbf{w}_{v,t})$ is also stored and computed at v. Node v can also evaluate the values of the matrix blocks $\mathbf{B}_{vv} = (1-w_{vv})\mathbf{I}$ and $\mathbf{B}_{vu} = w_{vu}\mathbf{I}$. Thus, if the NN-$k$ step components $\mathbf{d}_{u,t}^{(k)}$ are available at neighbors u, node v can determine the NN-$(k+1)$ step component $\mathbf{d}_{v,t}^{(k+1)}$ upon being communicated that information.

The expression in (5.19) represents an iterative computation embedded inside the NN-K recursion in (5.17). At time index t, we compute the local component of the NN-0 step $\mathbf{d}_{v,t}^{(0)} = -\mathbf{D}_{vv,t}^{-1}\mathbf{g}_{v,t}$. Upon exchanging this information with neighbors we use (5.19) to determine the NN-1 step $\mathbf{d}_{v,t}^{(1)}$. These can be exchanged to compuer $\mathbf{d}_{v,t}^{(2)}$ as in (5.19). Repeating this procedure K times, nodes ends up having determined their NN-K step component $\mathbf{d}_{v,t}^{(K)}$.

The resulting NN-K method is summarized in Algorithm 5. The descent iteration in (5.17) is implemented in Step 11. Implementation of this descent requires access to the

Algorithm 5 Network Newton-K method at node v

Require: Initial iterate $\mathbf{w}_{v,0}$. Weights w_{vu}. Penalty coefficient α.

1: $\mathbf{B}$ matrix blocks: $\mathbf{B}_{vv} = (1 - w_{vv})\mathbf{I}$ and $\mathbf{B}_{vu} = w_{vu}\mathbf{I}$
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: $\quad$ $\mathbf{D}$ matrix block: $\mathbf{D}_{vv,t} = \alpha\nabla^2 f_v(\mathbf{w}_{v,t}) + 2(1 - w_{vv})\mathbf{I}$
4: $\quad$ Exchange iterates $\mathbf{w}_{v,t}$ with neighbors $u \in \mathcal{N}_v$.
5: $\quad$ Gradient: $\mathbf{g}_{v,t} = (1 - w_{vv})\mathbf{w}_{v,t} - \sum_{u\in\mathcal{N}_v} w_{vu}\mathbf{w}_{u,t} + \alpha\nabla f_v(\mathbf{w}_{v,t})$
6: $\quad$ Compute NN-0 descent direction $\mathbf{d}_{v,t}^{(0)} = -\mathbf{D}_{vv,t}^{-1}\mathbf{g}_{v,t}$
7: $\quad$ **for** $k = 0, \ldots, K-1$ **do**
8: $\quad\quad$ Exchange elements $\mathbf{d}_{v,t}^{(k)}$ of the NN-k step with neighbors
9: $\quad\quad$ NN-$(k+1)$ step: $\mathbf{d}_{v,t}^{(k+1)} = \mathbf{D}_{vv,t}^{-1}\Big[\sum_{u\in\mathcal{N}_v, u=v} \mathbf{B}_{vu}\mathbf{d}_{u,t}^{(k)} - \mathbf{g}_{v,t}\Big]$
10: $\quad$ **end for**
11: $\quad$ Update local iterate: $\mathbf{w}_{v,t+1} = \mathbf{w}_{v,t} + \epsilon\, \mathbf{d}_{v,t}^{(K)}$.
12: **end for**

NN-K descent direction $\mathbf{d}_{v,t}^{(K)}$ which is computed by the loop in steps 8-10. Step 6 initializes the loop by computing the NN-0 step $\mathbf{d}_{v,t}^{(0)} = -\mathbf{D}_{vv,t}^{-1}\mathbf{g}_{v,t}$. The core of the loop is in Step 9 which corresponds to the recursion in (5.19). Step 8 stands for the variable exchange that is required to implement Step 9. After K iterations through this loop, the NN-K descent direction $\mathbf{d}_{v,t}^{(K)}$ is computed and can be used in Step 11. Both, Steps 6 and 9, require access to the local gradient component $\mathbf{g}_{v,t}$. This is evaluated in Step 5 after receiving the prerequisite information from neighbors in Step 4. Steps 1 and 3 compute the blocks $\mathbf{B}_{ii,t}$, $\mathbf{B}_{ij,t}$, and $\mathbf{D}_{vv,t}$ required in steps 6 and 9.

Remark 5 By trying to approximate the Newton step, NN-K ends up reducing the number of iterations required for convergence. Furthermore, the larger K is, the closer that the NN-K step gets to the Newton step, and the faster NN-K converges. We will justify these assertions both, analytically in Section 5.4, and numerically in Section 5.6. It is important to observe, however, that reducing the number of iterations reduces the computational cost but not necessarily the communication cost. In DGD, each node v shares its vector $\mathbf{w}_{v,t} \in \mathbb{R}^p$ with each of its neighbors $j \in \mathcal{N}_v$. In NN-K, node v exchanges not only the vector $\mathbf{w}_{v,t} \in \mathbb{R}^p$ with its neighboring nodes, but it also communicates iteratively the local components of the descent directions $\{\mathbf{d}_{v,t}^{(k)}\}_{k=0}^{K-1} \in \mathbb{R}^p$ so as to compute the descent direction $\mathbf{d}_{v,t}^{(K)}$. Hence, at each iteration, node v sends $|\mathcal{N}_v|$ vectors of size p to its neighbors in DGD, while in NN-K it sends $(K+1)|\mathcal{N}_v|$ vectors of the same size. Unless the original problem is well conditioned, NN-K also reduces total communication cost until convergence, even though the cost of each individual iteration is larger. However, the use of large K is unwarranted because the added benefit of better approximating the Newton step does not compensate the increase in communication cost.

5.4 Convergence analysis

In this section we show that as time progresses the sequence of objective function values $F(\mathbf{y}_t)$ [cf. (5.6)] approaches the optimal objective function value $F(\mathbf{y}^*)$. In proving this claim we make the following assumptions.

Assumption 9 *There exist constants* $0 \leq \delta \leq \Delta < 1$ *that lower and upper bound the diagonal weights for all* v*,*

$$0 < \delta \leq w_{vv} \leq \Delta < 1, \qquad v = 1, \ldots, V. \tag{5.20}$$

Assumption 10 *The local objective functions* $f_v(\mathbf{w})$ *are twice differentiable and the eigenvalues of the local Hessians are bounded with positive constants* $0 < m \leq M < \infty$*, i.e.*

$$m\mathbf{I} \preceq \nabla^2 f_v(\mathbf{w}) \preceq M\mathbf{I}. \tag{5.21}$$

Assumption 11 *The local objective function Hessians* $\nabla^2 f_v(\mathbf{w})$ *are Lipschitz continuous with respect to the Euclidian norm with parameter* L*. I.e., for all* $\mathbf{w}, \hat{\mathbf{w}} \in \mathbb{R}^p$*, it holds*

$$\|\nabla^2 f_v(\mathbf{w}) - \nabla^2 f_v(\hat{\mathbf{w}})\| \;\leq\; L\, \|\mathbf{w} - \hat{\mathbf{w}}\|. \tag{5.22}$$

The lower bound in Assumption 9 is more a definition than a constraint. To be more precise, the weights w_{uv} are positive if and only if $u \in \mathcal{N}_v$ or $u = v$. This observation verifies existence of a lower bound for the local weights w_{vv} that is defined as $\delta > 0$ in Assumption 9. The upper bound $\Delta < 1$ on the weights w_{vv} is true for all connected networks as long as neighbors $u \in \mathcal{N}_v$ are assigned nonzero weights $w_{uv} > 0$. This is because the matrix $\mathbf{W}$ is doubly stochastic [cf. (5.4)], which implies that $w_{vv} = 1 - \sum_{u \in \mathcal{N}_v} w_{vu} < 1$ as long as $w_{vu} > 0$.

The lower bound m for the eigenvalues of local objective function Hessians $\nabla^2 f_v(\mathbf{w})$ is equivalent to the strong convexity of local objective functions $f_v(\mathbf{w})$ with parameter m. The strong convexity assumption for the local objective functions $f_v(\mathbf{w})$ stated in Assumption 10 is customary in Newton-based methods, since the Hessian of objective function should be invertible to establish Newton's method [Chapter 9 of [20]]. The upper bound M for the eigenvalues of local objective function Hessians $\nabla^2 f_v(\mathbf{w})$ is similar to the condition that gradients $\nabla f_v(\mathbf{w})$ are Lipschitz continuous with parameter M for the case that functions are twice differentiable.

The restriction imposed by Assumption 11 is customary in the analysis of second order methods, see Section 9.5.3 of [20], which guarantees that the Hessians $\nabla^2 F(\mathbf{y})$ are also Lipschitz continuous as we show in the following lemma.

Lemma 14 *Consider the definition of objective function $F(\mathbf{y})$ in (5.6). If Assumption 11 holds then the objective function Hessian $\mathbf{H}(\mathbf{y}) =: \nabla^2 F(\mathbf{y})$ is Lipschitz continuous with parameter αL, i.e., for all $\mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^{Vp}$ we have*

$$\|\mathbf{H}(\mathbf{y}) - \mathbf{H}(\hat{\mathbf{y}})\| \leq \alpha L \|\mathbf{y} - \hat{\mathbf{y}}\|. \tag{5.23}$$

Proof: Consider two vectors $\mathbf{y} := [\mathbf{w}_1; \ldots; \mathbf{w}_V] \in \mathbb{R}^{Vp}$ and $\hat{\mathbf{y}} := [\hat{\mathbf{w}}_1; \ldots; \hat{\mathbf{w}}_V] \in \mathbb{R}^{Vp}$. Based on the Hessian expression in (5.10), we simplify the Euclidean norm $\|\mathbf{H}(\mathbf{y}) - \mathbf{H}(\hat{\mathbf{y}})\|$ as

$$\begin{aligned} \|\mathbf{H}(\mathbf{y}) - \mathbf{H}(\hat{\mathbf{y}})\| &= \alpha \|\mathbf{G}(\mathbf{y}) - \mathbf{G}(\hat{\mathbf{y}})\| \\ &= \alpha \max_{v=1,\ldots,V} \left\|\nabla^2 f_v(\mathbf{w}_v) - \nabla^2 f_v(\hat{\mathbf{w}}_v)\right\|. \end{aligned} \tag{5.24}$$

By using 11 and (5.24) we obtain that

$$\|\mathbf{H}(\mathbf{y}) - \mathbf{H}(\hat{\mathbf{y}})\| \leq \alpha L \max_v \|\mathbf{w}_v - \hat{\mathbf{w}}_v\| \leq \alpha L \|\mathbf{y} - \hat{\mathbf{y}}\|. \tag{5.25}$$

Therefore, the claim in (5.23) follows. ■

Lemma 14 states that the penalty objective function introduced in (5.6) has the property that the Hessians are Lipschitz continuous, while the Lipschitz constant is a function of the penalty coefficient $1/\alpha$. Thus, if we increase the penalty coefficient $1/\alpha$, or, equivalently, decrease α, the objective function $F(\mathbf{y})$ approaches a quadratic form because the curvature becomes constant.

To prove convergence properties of NN we need bounds for the eigenvalues of the block diagonal matrix $\mathbf{D}_t$, the block sparse matrix $\mathbf{B}$, and the Hessian $\mathbf{H}_t$. These eigenvalue bounds are established in the following proposition using the conditions imposed by Assumptions 9 and 10.

Proposition 4 *Consider the definitions of matrices $\mathbf{H}_t$, $\mathbf{D}_t$, and $\mathbf{B}$ in (5.10), (5.12), and (5.13), respectively. If Assumptions 9 and 10 hold true, then the eigenvalues of matrices $\mathbf{H}_t$, $\mathbf{D}_t$, and $\mathbf{B}$ are uniformly bounded as*

$$\alpha m \mathbf{I} \preceq \mathbf{H}_t \preceq (2(1-\delta) + \alpha M)\mathbf{I}, \tag{5.26}$$

$$(2(1-\Delta) + \alpha m)\mathbf{I} \preceq \mathbf{D}_t \preceq (2(1-\delta) + \alpha M)\mathbf{I}, \tag{5.27}$$

$$\mathbf{0} \preceq \mathbf{B} \preceq 2(1-\delta)\mathbf{I}. \tag{5.28}$$

Proof: The Gershgorin circle theorem states that each eigenvalue of a matrix $\mathbf{A}$ lies within at least one of the Gershgorin discs $D(a_{vv}, R_{vv})$ where the center a_{vv} is the vth diagonal

element of A and the radius $R_{vv} := \sum_{u \neq v} |a_{vu}|$ is the sum of the absolute values of all the non-diagonal elements of the vth row. Hence, Gershgorin discs can be considered as intervals of width $[a_{vv} - R_{vv}, a_{vv} + R_{vv}]$ for $\mathbf{I} - \mathbf{W}$, where $a_{vv} = 1 - w_{vv}$ and $R_{vv} = \sum_{u \neq v} |w_{vu}| = \sum_{u \neq v} w_{vu}$. Therefore, all the eigenvalues of $\mathbf{I} - \mathbf{W}$ are in at least one of the intervals $[1 - w_{vv} - \sum_{u \neq v} w_{vu}, 1 - w_{vv} + \sum_{u \neq v} w_{vu}]$. Since $\sum_u w_{vu} = 1$, it can be derived that $1 - w_{vv} = \sum_{u \neq v}^{V} w_{vu}$. Thus, the Gershgorin intervals can be simplified as $[0, 2(1 - w_{vv})]$ for $v = 1, \ldots, V$. This observation in association with the fact that $2(1 - w_{vv}) \leq 2(1 - \delta)$ implies that the eigenvalues of $\mathbf{I} - \mathbf{W}$ are in the interval $[0, 2(1 - \delta)]$ and consequently the eigenvalues of $\mathbf{I} - \mathbf{Z}$ are bounded as

$$\mathbf{0} \preceq \mathbf{I} - \mathbf{Z} \preceq 2(1 - \delta)\mathbf{I}. \tag{5.29}$$

Since matrix $\mathbf{G}_t$ is block diagonal and the eigenvalues of each diagonal block $\mathbf{G}_{vv,t} = \nabla^2 f_v(\mathbf{w}_{v,t})$ are bounded by constants $0 < m \leq M < \infty$ as mentioned in (5.21), we obtain

$$m\mathbf{I} \preceq \mathbf{G}_t \preceq M\mathbf{I}. \tag{5.30}$$

Considering the definition of the Hessian $\mathbf{H}_t := \mathbf{I} - \mathbf{Z} + \alpha\mathbf{G}_t$ and the bounds in (5.29) and (5.30), the first claim follows.

The definition of the matrix $\mathbf{D}_t$ in (5.12) yields

$$\mathbf{D}_t = \alpha\mathbf{G}_t + (\mathbf{I}_n - \mathbf{W}_d) \otimes \mathbf{I}_p \ , \tag{5.31}$$

where $\mathbf{W}_d$ is defined as $\mathbf{W}_d := \text{diag}(\mathbf{W})$. Note that matrix $\mathbf{I}_n - \mathbf{W}_d$ is diagonal and the v-th diagonal component is $1 - w_{vv}$. Since the local weights satisfy $\delta \leq w_{vv} \leq \Delta$, we obtain that the eigenvalues of $\mathbf{I}_n - \mathbf{W}_d$ are bounded below and above by $1 - \Delta$ and $1 - \delta$, respectively. Since the eigenvalues of $(\mathbf{I}_n - \mathbf{W}_d)$ and $(\mathbf{I}_n - \mathbf{W}_d) \otimes \mathbf{I}_p$ are identical we obtain

$$(1 - \Delta)\mathbf{I}_{np} \preceq (\mathbf{I}_n - \mathbf{W}_d) \otimes \mathbf{I}_p \preceq (1 - \delta)\mathbf{I}_{np} \tag{5.32}$$

Considering the relation in (5.31) and bounds in (5.30) and (5.32), the second claim follows.

Based on the definition of $\mathbf{B}$ in (5.13), we can write

$$\mathbf{B} = (\mathbf{I} - 2\mathbf{W}_d + \mathbf{W}) \otimes \mathbf{I}. \tag{5.33}$$

Note that in the v-th row of matrix $\mathbf{I} - 2\mathbf{W}_d + \mathbf{W}$, the diagonal component is $1 - w_{vv}$ and the uth component is w_{vu} for all $u \neq v$. Using Gershgorin theorem and the same argument that we established for the eigenvalues of $\mathbf{I} - \mathbf{Z}$, we can write

$$\mathbf{0} \preceq \mathbf{I} - 2\mathbf{W}_d + \mathbf{W} \preceq 2(1 - \delta)\mathbf{I}. \tag{5.34}$$

Based on (5.34) and (5.33), the last claim follows. ■

Proposition 4 states that Hessian matrix $\mathbf{H}_t$ and block diagonal matrix $\mathbf{D}_t$ are positive definite, while matrix $\mathbf{B}$ is positive semidefinite.

As we noted in Section 5.3, for the expansion in (5.14) to be valid the eigenvalues of the matrix $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ must be nonnegative and strictly smaller than 1. The following proposition states that this is true for all times t.

Proposition 5 *Consider the definitions of the matrices* $\mathbf{D}_t$ *in* (5.12) *and* $\mathbf{B}$ *in* (5.13). *If Assumptions 9 and 10 hold true, the matrix* $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ *is positive semidefinite and its eigenvalues are bounded above by a constant* $\rho < 1$

$$\mathbf{0} \preceq \mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2} \preceq \rho\mathbf{I}, \tag{5.35}$$

where $\rho := 2(1-\delta)/(2(1-\delta)+\alpha m)$.

Proof: According to the result of Proposition 1, $\mathbf{D}_t$ is positive definite and $\mathbf{B}$ is positive semidefinite which immediately implies that $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ is positive semidefinite.

Recall the definition of $\mathbf{D}_t$ in (5.12) and define the matrix $\hat{\mathbf{D}}$ as a special case of matrix $\mathbf{D}_t$ for $\alpha = 0$. I.e., $\hat{\mathbf{D}} := 2(\mathbf{I} - \mathbf{Z}_d)$. Notice that $\hat{\mathbf{D}}$ is diagonal, time invariant, and only depends on the structure of the network. Since $\hat{\mathbf{D}}$ is diagonal and each diagonal component $1 - w_{vv}$ is strictly larger than 0, $\hat{\mathbf{D}}$ is positive definite and invertible. Hence, we can write

$$\mathbf{D}_t^{-\frac{1}{2}}\mathbf{B}\mathbf{D}_t^{-\frac{1}{2}} = (\mathbf{D}_t^{-\frac{1}{2}}\hat{\mathbf{D}}^{\frac{1}{2}})(\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{B}\hat{\mathbf{D}}^{-\frac{1}{2}})(\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{D}_t^{-\frac{1}{2}}). \tag{5.36}$$

We proceed to find an upper bound for the eigenvalues of the matrix $\hat{\mathbf{D}}^{-1/2}\mathbf{B}\hat{\mathbf{D}}^{-1/2}$ in (5.36). Observing the fact that matrices $\hat{\mathbf{D}}^{-1/2}\mathbf{B}\hat{\mathbf{D}}^{-1/2}$ and $\mathbf{B}\hat{\mathbf{D}}^{-1}$ are *similar*, eigenvalues of these matrices are identical. Hence, we proceed to characterize an upper bound for the eigenvalues of matrix $\mathbf{B}\hat{\mathbf{D}}^{-1}$. Based on the definitions of $\mathbf{B}$ and $\hat{\mathbf{D}}$, the product $\mathbf{B}\hat{\mathbf{D}}^{-1}$ is given by $\mathbf{B}\hat{\mathbf{D}}^{-1} = (\mathbf{I} - 2\mathbf{Z}_d + \mathbf{Z})\,(2(\mathbf{I} - \mathbf{Z}_d))^{-1}$. Therefore, the blocks of the matrix $\mathbf{B}\hat{\mathbf{D}}^{-1}$ are given by

$$[\mathbf{B}\hat{\mathbf{D}}^{-1}]_{vv} = \frac{1}{2}\mathbf{I} \quad \text{and} \quad [\mathbf{B}\hat{\mathbf{D}}^{-1}]_{vu} = \frac{w_{vu}}{2(1-w_{uu})}\mathbf{I}. \tag{5.37}$$

Thus, each diagonal component of the matrix $\mathbf{B}\hat{\mathbf{D}}^{-1}$ is $1/2$ and that the sum of non-diagonal components of column v is

$$\sum_{u=1,u\neq v}^{Vp} \mathbf{B}\hat{\mathbf{D}}^{-1}[uv] = \frac{1}{2}\sum_{u=1,u\neq v}^{Vp} \frac{w_{uv}}{1-w_{vv}} = \frac{1}{2}. \tag{5.38}$$

Consider (5.38) and apply Gershgorin theorem to obtain

$$0 \leq \mu_v(\mathbf{B}\hat{\mathbf{D}}^{-1}) \leq 1 \qquad v = 1, \ldots, V, \tag{5.39}$$

where $\mu_v(\mathbf{B}\hat{\mathbf{D}}^{-1})$ indicates the v-th eigenvalue of the matrix $\mathbf{B}\hat{\mathbf{D}}^{-1}$. The bounds in (5.39) and *similarity* of the matrices $\mathbf{B}\hat{\mathbf{D}}^{-1}$ and $\hat{\mathbf{D}}^{-1/2}\mathbf{B}\hat{\mathbf{D}}^{-1/2}$ show that the eigenvalues of the matrix $\hat{\mathbf{D}}^{-1/2}\mathbf{B}\hat{\mathbf{D}}^{-1/2}$ are uniformly bounded in the interval

$$0 \leq \mu_v(\hat{\mathbf{D}}^{-1/2}\mathbf{B}\hat{\mathbf{D}}^{-1/2}) \leq 1. \tag{5.40}$$

Based on (5.36), to characterize the bounds for the eigenvalues of $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$, the bounds for the eigenvalues of the matrix $\hat{\mathbf{D}}^{1/2}\mathbf{D}_t^{-1/2}$ should be studied as well. Notice that according to the definitions of $\hat{\mathbf{D}}$ and $\mathbf{D}_t$, the product $\hat{\mathbf{D}}^{1/2}\mathbf{D}_t^{-1/2}$ is block diagonal and the v-th diagonal block is

$$\left[\hat{\mathbf{D}}^{1/2}\mathbf{D}_t^{-1/2}\right]_{vv} = \left(\frac{\alpha\nabla^2 f_v(\mathbf{w}_{v,t})}{2(1-w_{vv})} + \mathbf{I}\right)^{-1/2}. \tag{5.41}$$

Observe that according to Assumption 1, the eigenvalues of local Hessian matrices $\nabla^2 f_v(\mathbf{w}_v)$ are bounded by m and M. Further notice that the diagonal elements of weight matrix w_{vv} are bounded by δ and Δ, i.e. $\delta \leq w_{vv} \leq \Delta$. Considering these bounds we can show that the eigenvalues of matrices $(\alpha/2(1-w_{vv}))\nabla^2 f_v(\mathbf{w}_{v,t}) + \mathbf{I}$ are lower and upper bounded as

$$\left[\frac{\alpha m}{2(1-\delta)} + 1\right]\mathbf{I} \preceq \frac{\alpha\nabla^2 f_v(\mathbf{w}_{v,t})}{2(1-w_{vv})} + \mathbf{I} \preceq \left[\frac{\alpha M}{2(1-\Delta)} + 1\right]\mathbf{I}. \tag{5.42}$$

By considering the bounds in (5.42) and the expression in (5.41), the eigenvalues of the matrix $\hat{\mathbf{D}}^{1/2}\mathbf{D}_t^{-1/2}$ are bounded as

$$\left[\frac{2(1-\Delta)}{2(1-\Delta)+\alpha M}\right]^{\frac{1}{2}} \leq \mu_v\left(\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{D}_t^{-\frac{1}{2}}\right) \leq \left[\frac{2(1-\delta)}{2(1-\delta)+\alpha m}\right]^{\frac{1}{2}}, \tag{5.43}$$

for $v = 1, \ldots, V$. Observing the decomposition in (5.36), the norm of the matrix $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ is upper bounded as

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{B}\mathbf{D}_t^{-\frac{1}{2}}\| \leq \|\mathbf{D}_t^{-\frac{1}{2}}\hat{\mathbf{D}}^{1/2}\|^2 \, \|\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{B}\hat{\mathbf{D}}^{-\frac{1}{2}}\|. \tag{5.44}$$

Considering the symmetry of matrices $\hat{\mathbf{D}}^{1/2}\mathbf{D}_t^{-1/2}$ and $\hat{\mathbf{D}}^{-1/2}\mathbf{B}\hat{\mathbf{D}}^{-1/2}$, and the upper bounds for their eigenvalues in (5.40) and (5.43), respectively, we can substitute the norm of these two matrices by the upper bounds of their eigenvalues and simplify the upper bound in

(5.44) to

$$\|\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\| \leq \frac{2(1-\delta)}{2(1-\delta)+\alpha m}. \tag{5.45}$$

Since $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ is positive semidefinite and symmetric, the result in (5.35) follows. ■

The results in Proposition 4 would lead to the trivial upper bound $2(1-\delta)/(\alpha M+2(1-\Delta))$ for the eigenvalues of $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$. The upper bound in Proposition 5 is tighter and follows from the structure of the matrix $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$.

The bounds for the eigenvalues of $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ in (5.35) guarantee convergence of the Taylor series in (5.14). As mentioned in Section 5.3, NN-K truncates the first K summands of the Hessian inverse Taylor series in (5.14) to approximate the Hessian inverse of the objective function in optimization problem (5.6). To evaluate the performance of NN-K we study the error of the Hessian inverse approximation by defining the *error matrix* $\mathbf{E}_t \in \mathbb{R}^{Vp\times Vp}$ as

$$\mathbf{E}_t := \mathbf{I} - \hat{\mathbf{H}}_t^{(K)^{-1/2}}\mathbf{H}_t\hat{\mathbf{H}}_t^{(K)^{-1/2}}. \tag{5.46}$$

The error matrix $\mathbf{E}_t$ measures closeness of the Hessian inverse approximation matrix $\hat{\mathbf{H}}_t^{(K)^{-1}}$ and the exact Hessian inverse $\mathbf{H}_t^{-1}$ at time t. Based on the definition of the error matrix $\mathbf{E}_t$, if the Hessian inverse approximation $\hat{\mathbf{H}}_t^{(K)^{-1}}$ approaches the exact Hessian inverse $\mathbf{H}_t^{-1}$ the error matrix $\mathbf{E}_t$ approaches the zero matrix $\mathbf{0}$. We therefore bound the error of the Hessian inverse approximation by developing a bound for the eigenvalues of $\mathbf{E}_t$. This bound is provided in the following proposition.

Proposition 6 *Consider the NN-K method in* (5.12)-(5.17) *and the definition of error matrix* $\mathbf{E}_t$ *in* (5.46)*. Further, recall the definition of the constant* $\rho := 2(1-\delta)/(\alpha m+2(1-\delta)) < 1$ *in Proposition 5. The error matrix* $\mathbf{E}_t$ *is positive semidefinite and all its eigenvalues are upper bounded by* ρ^{K+1}*,*

$$\mathbf{0} \preceq \mathbf{E}_t \preceq \rho^{K+1}\mathbf{I}. \tag{5.47}$$

Proof: In this proof and the rest of the proofs we denote the Hessian approximation as $\hat{\mathbf{H}}_t^{-1}$ instead of $\hat{\mathbf{H}}_t^{(K)^{-1}}$ for simplification of equations. To prove lower and upper bounds for the eigenvalues of the error matrix $\mathbf{E}_t$ we first develop a simplification for the matrix $\mathbf{I}-\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}$ in the following lemma. For the proof of the following lemma Check Lemma 2 in [128].

Lemma 15 *Consider the NN-K method as defined in* (5.12)-(5.17)*. The matrix* $\mathbf{I}-\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}$ *can be simplified as*

$$\mathbf{I} - \mathbf{H}_t\hat{\mathbf{H}}_t^{-1} = \left(\mathbf{B}\mathbf{D}_t^{-1}\right)^{K+1}. \tag{5.48}$$

Recall the result in Proposition 5. Since the matrices $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ and $\mathbf{B}_t\mathbf{D}_t^{-1}$ are similar (conjugate) the sets of eigenvalues of these two matrices are identical. Thus, the

eigenvalues of $\mathbf{BD}^{-1}$ are bounded as

$$0 \;\leq\; \mu_v(\mathbf{BD}^{-1}) \;\leq\; \rho, \tag{5.49}$$

for $i = 1, 2, \ldots, np$. This result in association with (5.48) yields

$$0 \;\leq\; \mu_v(\mathbf{I} - \mathbf{H}_t\hat{\mathbf{H}}_t^{-1}) \;\leq\; \rho^{K+1}. \tag{5.50}$$

Observe that the error matrix $\mathbf{E}_t = \mathbf{I} - \hat{\mathbf{H}}_t^{-1/2}\mathbf{H}_t\hat{\mathbf{H}}_t^{-1/2}$ is the conjugate of matrix $\mathbf{I} - \mathbf{H}_t\hat{\mathbf{H}}_t^{-1}$. Hence, the bounds for the eigenvalues of matrix $\mathbf{I} - \mathbf{H}_t\hat{\mathbf{H}}_t^{-1}$ also hold for the eigenvalues of error matrix $\mathbf{E}_t$ and the claim in (5.47) follows. ■

Proposition 6 asserts that the error in the approximation of the Hessian inverse, thereby on the approximation of the Newton step, is bounded by ρ^{K+1}. This result corroborates the intuition that the larger K is, the closer that $\mathbf{d}_{v,t}^{(K)}$ approximates the Newton step. This closer approximation comes at the cost of increasing the communication cost of each descent iteration. The decrease of this error being proportional to ρ^{K+1} hints that using a small value of K should suffice in practice. Further to decrease ρ we can increase δ or increase α. Increasing δ calls for assigning substantial weight to w_{vv}. Increasing α comes at the cost of moving the solution of (5.6) away from the solution of (5.9) and its equivalent (5.1).

Bounds on the eigenvalues of the objective function Hessian $\mathbf{H}_t$ are central to the convergence analysis of Newton's method [Chapter 9 of [20]]. Lower bounds for the Hessian eigenvalues guarantee that the matrix is nonsingular. Upper bounds imply that the minimum eigenvalue of the Hessian inverse $\mathbf{H}^{-1}$ is strictly larger than zero, which, in turn, implies a strict decrement in each Newton step. Analogous bounds for the eigenvalues of the NN approximate Hessian inverses $\hat{\mathbf{H}}_t^{(K)^{-1}}$ are required. These bounds are studied in the following lemma.

Lemma 16 *Consider the NN-K method as defined in* (5.12)-(5.17)*. If Assumptions 9 and 10 hold true, we have*

$$\lambda\mathbf{I} \;\preceq\; \hat{\mathbf{H}}_t^{(K)^{-1}} \;\preceq\; \Lambda\mathbf{I}, \tag{5.51}$$

where constants λ *and* Λ *are defined as*

$$\lambda := \frac{1}{2(1-\delta) + \alpha M} \quad \textit{and} \quad \Lambda := \frac{1 - \rho^{K+1}}{(1-\rho)(2(1-\Delta) + \alpha m)}. \tag{5.52}$$

Proof: Based on the Cauchy-Schwarz inequality, the product of the norms is larger than norm of the products. This observation and the definition of $\hat{\mathbf{H}}_t^{-1}$ in (5.15) lead to

$$\|\hat{\mathbf{H}}_t^{-1}\| \leq \|\mathbf{D}_t^{-\frac{1}{2}}\|^2 \|\,\mathbf{I} + \mathbf{D}_t^{-\frac{1}{2}}\mathbf{B}\mathbf{D}_t^{-\frac{1}{2}} + \ldots + [\mathbf{D}_t^{-\frac{1}{2}}\mathbf{B}\mathbf{D}_t^{-\frac{1}{2}}]^K\|. \tag{5.53}$$

As a result of Proposition 4 the eigenvalues of $\mathbf{D}_t$ are bounded below by $2(1-\Delta)+\alpha m$. Thus, the maximum eigenvalue of its inverse $\mathbf{D}_t^{-1}$ is smaller than $1/(2(1-\Delta)+\alpha m)$, and, therefore, the norm of the matrix $\mathbf{D}_t^{-1/2}$ is bounded above as

$$\|\mathbf{D}_t^{-1/2}\| \;\leq\; [2(1-\Delta)+\alpha m]^{-1/2}\,. \tag{5.54}$$

Based on the result in Proposition 5, the eigenvalues of $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ are smaller than ρ. Further, using the symmetry and positive definiteness of $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ we obtain

$$\|\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\| \;\leq\; \rho. \tag{5.55}$$

Using the triangle inequality in (5.53) to claim that the norm of the sum is smaller than the sum of the norms and substituting the bounds in (5.54) and (5.55) into the resulting expression yield

$$\|\hat{\mathbf{H}}_t^{-1}\| \leq \frac{1}{2(1-\Delta)+\alpha m}\;\sum_{k=0}^{K}\rho^k. \tag{5.56}$$

Since $\rho<1$, the sum $\sum_{k=0}^{K}\rho^k$ can be simplified to $(1-\rho^{K+1})/(1-\rho)$. Considering this simplification for the sum in (5.56), the upper bound in (5.51) for the eigenvalues of the approximate Hessian inverse $\hat{\mathbf{H}}_t^{-1}$ follows.

In expression (5.15), all the summands except the first one, $\mathbf{D}_t^{-1}$, are positive semidefinite. Hence, the approximate Hessian inverse $\hat{\mathbf{H}}_t^{-1}$ is the sum of the matrix $\mathbf{D}_t^{-1}$ and K positive semidefinite matrices and as a result we can conclude that

$$\mathbf{D}_t^{-1} \preceq\; \hat{\mathbf{H}}_t^{-1}. \tag{5.57}$$

Proposition 4 shows that the eigenvalues of $\mathbf{D}_t$ are bounded above by $2(1-\delta)+\alpha M$ which leads to the conclusion that there exists a lower bound for the eigenvalues of $\mathbf{D}_t^{-1}$,

$$(2(1-\delta)+\alpha M)^{-1}\;\mathbf{I} \;\preceq\; \mathbf{D}_t^{-1}. \tag{5.58}$$

The claim in (5.51) follows from the results in (5.57) and (5.58). ■

According to the result in Lemma 16, the NN-K approximate Hessian inverses $\hat{\mathbf{H}}_t^{(K)^{-1}}$ are strictly positive definite and have all of their eigenvalues bounded between the positive and finite constants λ and Λ. This is true for all K and uniform across all iteration indexes t. Considering these eigenvalue bounds and the fact that $-\mathbf{g}_t$ is a descent direction, the approximate Newton step $-\hat{\mathbf{H}}_t^{(K)^{-1}}\mathbf{g}_t$ enforces convergence of the iterate $\mathbf{y}_t$ to the optimal argument $\mathbf{y}^*$ of the penalized objective function $F(\mathbf{y})$ in (5.6). In the following theorem we show that if the stepsize ϵ is properly chosen, the sequence of objective function values

$F(\mathbf{y}_t)$ converges at least linearly to the optimal objective function value $F(\mathbf{y}^*)$.

Theorem 7 *Consider the NN-K method as defined in* (5.12)-(5.17) *and the objective function* $F(\mathbf{y})$ *as introduced in* (5.6). *Further, recall the definitions of the lower and upper bounds* λ *and* Λ, *respectively, for the eigenvalues of the approximate Hessian inverse* $\hat{\mathbf{H}}_t^{(K)^{-1}}$ *in* (5.52). *If the stepsize* ϵ *is chosen as*

$$\epsilon \leq \min \left\{ 1 \ , \left[\frac{3m\lambda^{\frac{5}{2}}}{L\Lambda^3 (F(\mathbf{y}_0) - F(\mathbf{y}^*))^{\frac{1}{2}}} \right]^{\frac{1}{2}} \right\}, \tag{5.59}$$

and Assumptions 9-11 hold, the sequence $F(\mathbf{y}_t)$ *converges to the optimal argument* $F(\mathbf{y}^*)$ *at least linearly as*

$$F(\mathbf{y}_t) - F(\mathbf{y}^*) \leq (1-\zeta)^t (F(\mathbf{y}_0) - F(\mathbf{y}^*)), \tag{5.60}$$

where the constant $0 < \zeta < 1$ *is explicitly given by*

$$\zeta := (2-\epsilon)\epsilon\alpha m\lambda - \frac{\alpha\epsilon^3 L\Lambda^3 (F(\mathbf{y}_0) - F(\mathbf{y}^*))^{\frac{1}{2}}}{6\lambda^{\frac{3}{2}}}. \tag{5.61}$$

Proof: See Appendix B.1. ■

Theorem 7 shows that the objective function error sequence $F(\mathbf{y}_t) - F(\mathbf{y}^*)$ asymptoticly converges to zero and that the rate of convergence is at least linear. Note that according to the definition of the convergence parameter ζ in Theorem 7 and the definitions of λ and Λ in (5.52), increasing α leads to faster convergence. This observation verifies existence of a tradeoff between rate and accuracy of convergence. For large values of α the sequence generated by network Newton converges faster to the optimal solution of (5.6). These faster convergence comes at the cost of increasing the distance between the optimal solutions of (5.6) and (5.1). Conversely, smaller α implies smaller gap between the optimal solutions of (5.6) and (5.1), but the convergence rate of NN-K is slower. In the following section, we illustrate the connection between network Newton and the centralized Newton's method.

5.4.1 Analysis of network Newton as a Newton-like method

To connect the proposed NN method with the classic Newton's method, we first study the difference between these methods. In particular, the following lemma shows that the convergence of the norm of the weighted gradient $\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$ in NN-K is akin to the convergence of Newton's method with constant stepsize. The difference is the appearance of a term associated with the error of the Hessian inverse approximation as we formally state next.

Lemma 17 *Consider the NN-K method as defined in* (5.12)-(5.17). *If Assumptions 9-11 hold, the sequence of weighted gradients* $\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}$ *satisfies*

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_{t+1}\| \leq \left(1-\epsilon+\epsilon\rho^{K+1}\right)\left[1+\Gamma_1(1-\zeta)^{\frac{(t-1)}{4}}\right]\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\| + \epsilon^2\Gamma_2\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2, \quad (5.62)$$

where the constants Γ_1 *and* Γ_2 *are defined as*

$$\Gamma_1 := \frac{(\alpha\epsilon L\Lambda)^{\frac{1}{2}}(F(\mathbf{y}_0)-F(\mathbf{y}^*))^{\frac{1}{4}}}{\lambda^{\frac{3}{4}}(2(1-\Delta)+\alpha m)}, \quad \Gamma_2 := \frac{\alpha L\Lambda^2}{2\lambda(2(1-\Delta)+\alpha m)^{\frac{1}{2}}}. \quad (5.63)$$

Proof: To simplify notation we use $\hat{\mathbf{H}}_t^{-1}$ to indicate the approximate Hessian inverse $\hat{\mathbf{H}}_t^{(K)^{-1}}$. Based on Lemma 1.2.3 in [84], the Lipschitz continuity of Hessians with constant αL yields

$$\|\mathbf{g}_{t+1}-\mathbf{g}_t+\epsilon\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\| \leq \frac{\epsilon^2\alpha L}{2}\|\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\|^2, \quad (5.64)$$

where we have used $\mathbf{y}_{t+1}-\mathbf{y}_t = -\epsilon\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t$. Based on the definition of matrix norm, we can write

$$\|\mathbf{D}_t^{-\frac{1}{2}}[\mathbf{g}_{t+1}-\mathbf{g}_t+\epsilon\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t]\| \leq \|\mathbf{D}_t^{-\frac{1}{2}}\|\|\mathbf{g}_{t+1}-\mathbf{g}_t+\epsilon\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\|. \quad (5.65)$$

Substituting $\|\mathbf{g}_{t+1}-\mathbf{g}_t+\epsilon\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\|$ in the right hand side of (5.65) by the upper bound in (5.64) leads to

$$\|\mathbf{D}_t^{-\frac{1}{2}}[\mathbf{g}_{t+1}-\mathbf{g}_t+\epsilon\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t]\| \leq \frac{\epsilon^2\alpha L}{2}\|\mathbf{D}_t^{-\frac{1}{2}}\|\|\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\|^2. \quad (5.66)$$

Based on the triangle inequality, for any vectors $\mathbf{a}$ and $\mathbf{b}$, and a positive constant C, if the relation $\|\mathbf{a}-\mathbf{b}\| \leq C$ holds, then $\|\mathbf{a}\| \leq \|\mathbf{b}\| + C$. Thus, we can use the result in (5.66) to write

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_{t+1}\| \leq \|\mathbf{D}_t^{-\frac{1}{2}}[\mathbf{g}_t-\epsilon\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t]\| + \frac{\epsilon^2\alpha L}{2}\|\mathbf{D}_t^{-\frac{1}{2}}\|\|\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\|^2. \quad (5.67)$$

Write $\mathbf{D}_t^{-1/2}\mathbf{g}_t$ as the sum $(1-\epsilon)(\mathbf{D}_t^{-1/2}\mathbf{g}_t)+\epsilon(\mathbf{D}_t^{-1/2}\mathbf{g}_t)$ and use the triangle inequality to obtain

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_{t+1}\| \leq (1-\epsilon)\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_t\| + \epsilon\|\mathbf{D}_t^{-\frac{1}{2}}[\mathbf{I}-\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}]\mathbf{g}_t\| + \frac{\epsilon^2\alpha L}{2}\|\mathbf{D}_t^{-\frac{1}{2}}\|\|\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\|^2. \quad (5.68)$$

Use the result in Lemma 15 to write

$$\|\mathbf{D}_t^{-\frac{1}{2}}[\mathbf{I}-\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}]\mathbf{g}_t\| = \|[\mathbf{D}_t^{-\frac{1}{2}}\mathbf{B}\mathbf{D}_t^{-\frac{1}{2}}]^{K+1}\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_t\|. \quad (5.69)$$

The result in Proposition 5 implies that $\|[\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}]^{K+1}\| \leq \rho^{K+1}$. Considering this upper bound and the simplification in (5.69) we can write

$$\|\mathbf{D}_t^{-1/2}[\mathbf{I} - \mathbf{H}_t\hat{\mathbf{H}}_t^{-1}]\mathbf{g}_t\| \leq \rho^{K+1}\|\mathbf{D}_t^{-1/2}\mathbf{g}_t\|. \tag{5.70}$$

Substitute the upper bound in (5.70) into (5.68) and use the inequality $\|\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\| \leq \|\hat{\mathbf{H}}_t^{-1}\|\|\mathbf{g}_t\|$ to write

$$\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\| \leq (1 - \epsilon + \epsilon\rho^{K+1})\|\mathbf{D}_t^{-1/2}\mathbf{g}_t\| + \frac{\alpha\epsilon^2 L}{2}\|\mathbf{D}_t^{-1/2}\|\|\hat{\mathbf{H}}_t^{-1}\|^2\|\mathbf{g}_t\|^2. \tag{5.71}$$

Note that $\|\mathbf{D}_t^{-1} - \mathbf{D}_{t-1}^{-1}\|$ is bounded above as

$$\left\|\mathbf{D}_t^{-1} - \mathbf{D}_{t-1}^{-1}\right\| \leq \left\|\mathbf{D}_t^{-1}\right\| \|\mathbf{D}_t - \mathbf{D}_{t-1}\| \left\|\mathbf{D}_{t-1}^{-1}\right\|. \tag{5.72}$$

The eigenvalues of $\mathbf{D}_t$ and $\mathbf{D}_{t-1}$ are bounded below by $\alpha m + 2(1-\Delta)$. Thus, the eigenvalues of $\mathbf{D}_t^{-1}$ and $\mathbf{D}_{t-1}^{-1}$ are bounded above by $1/(\alpha m + 2(1-\Delta))$. Hence,

$$\left\|\mathbf{D}_t^{-1} - \mathbf{D}_{t-1}^{-1}\right\| \leq (2(1-\Delta) + \alpha m)^{-2} \|\mathbf{D}_t - \mathbf{D}_{t-1}\|. \tag{5.73}$$

The difference $\mathbf{D}_t - \mathbf{D}_{t-1}$ can be simplified as $\alpha(\mathbf{G}_t - \mathbf{G}_{t-1})$. Moreover, $\mathbf{H}_t - \mathbf{H}_{t-1} = \alpha(\mathbf{G}_t - \mathbf{G}_{t-1})$. Thus, $\mathbf{D}_t - \mathbf{D}_{t-1} = \mathbf{H}_t - \mathbf{H}_{t-1}$. This observation in conjunction with the Lipschitz continuity of the Hessians with parameter αL implies that

$$\|\mathbf{D}_t - \mathbf{D}_{t-1}\| \leq \alpha L\|\mathbf{y}_t - \mathbf{y}_{t-1}\|. \tag{5.74}$$

Replace $\|\mathbf{D}_t - \mathbf{D}_{t-1}\|$ in (5.73) by the bound in (5.74) to obtain

$$\left\|\mathbf{D}_t^{-1} - \mathbf{D}_{t-1}^{-1}\right\| \leq \frac{\alpha L}{(2(1-\Delta) + \alpha m)^2} \|\mathbf{y}_t - \mathbf{y}_{t-1}\|. \tag{5.75}$$

Note that $|\mathbf{g}_t^T(\mathbf{D}_t^{-1} - \mathbf{D}_{t-1}^{-1})\mathbf{g}_t|$ is bounded above by $\|\mathbf{D}_t^{-1} - \mathbf{D}_{t-1}^{-1}\|\|\mathbf{g}_t\|^2$. Considering the upper bound for $\|\mathbf{D}_t^{-1} - \mathbf{D}_{t-1}^{-1}\|$ in (5.75), the term $|\mathbf{g}_t^T(\mathbf{D}_t^{-1} - \mathbf{D}_{t-1}^{-1})\mathbf{g}_t|$ is bounded above by

$$\left|\mathbf{g}_t^T(\mathbf{D}_t^{-1} - \mathbf{D}_{t-1}^{-1})\mathbf{g}_t\right| \leq \frac{\alpha L \ \|\mathbf{y}_t - \mathbf{y}_{t-1}\|\|\mathbf{g}_t\|^2}{(2(1-\Delta) + \alpha m)^2}. \tag{5.76}$$

Using the result in (5.76), and simplifactions $|\mathbf{g}_t^T\mathbf{D}_{t-1}^{-1}\mathbf{g}_t| = \|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2$ and $|\mathbf{g}_t^T\mathbf{D}_t^{-1}\mathbf{g}_t| = \|\mathbf{D}_t^{-1/2}\mathbf{g}_t\|^2$, we can write

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_t\|^2 \leq \|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\|^2 + \frac{\alpha L \ \|\mathbf{y}_t - \mathbf{y}_{t-1}\|\|\mathbf{g}_t\|^2}{(2(1-\Delta) + \alpha m)^2}. \tag{5.77}$$

For any constants a, b, and c if $a^2 \leq b^2 + c^2$ holds, then $|a| \leq |b| + |c|$ holds. Using this result and (5.77) we obtain

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_t\| \leq \|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\| + \frac{(\alpha L\|\mathbf{y}_t - \mathbf{y}_{t-1}\|)^{\frac{1}{2}}\,\|\mathbf{g}_t\|}{2(1-\Delta)+\alpha m}. \tag{5.78}$$

Considering the update in (5.17) we can substitute $\mathbf{y}_t - \mathbf{y}_{t-1}$ by $-\epsilon\hat{\mathbf{H}}_{t-1}^{-1}\mathbf{g}_{t-1}$. Applying this substitution into (5.78) yields

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_t\| \leq \|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\| + \frac{[\alpha\epsilon L\|\hat{\mathbf{H}}_{t-1}^{-1}\mathbf{g}_{t-1}\|]^{\frac{1}{2}}\|\mathbf{g}_t\|}{2(1-\Delta)+\alpha m}. \tag{5.79}$$

If we substitute $\|\mathbf{D}_t^{-1/2}\mathbf{g}_t\|$ by the upper bound in (5.79) and substitute $\|\hat{\mathbf{H}}_{t-1}^{-1}\mathbf{g}_{t-1}\|$ by the upper bound $\|\hat{\mathbf{H}}_{t-1}^{-1}\|\|\mathbf{g}_{t-1}\|$, the inequality in (5.71) can be written as

$$\begin{aligned}\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\| \leq \left(1-\epsilon+\epsilon\rho^{K+1}\right)\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\| + \frac{\left(1-\epsilon+\epsilon\rho^{K+1}\right)[\alpha\epsilon L\|\hat{\mathbf{H}}_{t-1}^{-1}\|\,\|\mathbf{g}_{t-1}\|]^{1/2}}{2(1-\Delta)+\alpha m}\|\mathbf{g}_t\| \\ + \frac{\alpha\epsilon^2 L}{2}\|\mathbf{D}_t^{-1/2}\|\|\hat{\mathbf{H}}_t^{-1}\|^2\,\|\mathbf{g}_t\|^2\,.\end{aligned} \tag{5.80}$$

Note that $\mu_{min}(\mathbf{D}_{t-1}^{-1/2})\|\mathbf{g}_t\| \leq \|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$. Considering this inequality and the lower bound $(2(1-\delta)+\alpha M)^{-1/2}$ for the eigenvalues of $\mathbf{D}_{t-1}^{-1/2}$ we can write

$$\|\mathbf{g}_t\| \leq (2(1-\delta)+\alpha M)^{1/2}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|. \tag{5.81}$$

Substitute $\|\mathbf{g}_t\|$ by the upper bound in (5.81), use the definition $\lambda := 1/(2(1-\delta)+\alpha M)$, replace the norms the norms $\|\hat{\mathbf{H}}_t^{-1}\|$ and $\|\hat{\mathbf{H}}_{t-1}^{-1}\|$ by their upper bound Λ, and use the fact that $\|\mathbf{D}_t^{-1/2}\|$ is bounded above by $1/(2(1-\Delta)+\alpha m)^{1/2}$ to rewrite the right hand side of (5.80) as

$$\begin{aligned}\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_{t+1}\| \leq (1-\epsilon+\epsilon\rho^{K+1})[1+C_1\|\mathbf{g}_{t-1}\|^{\frac{1}{2}}]\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\| \\ + \frac{\alpha\epsilon^2 L\Lambda^2}{2\lambda(2(1-\Delta)+\alpha m)^{\frac{1}{2}}}\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\|^2,\end{aligned} \tag{5.82}$$

where $C_1 := \left[\alpha\epsilon L\Lambda/\lambda(2(1-\Delta)+\alpha m)^2\right]^{1/2}$.

Since the eigenvalues of the Hessian are upper bounded by $2(1-\delta)+\alpha M$, for any vectors $\hat{\mathbf{y}}$ and $\mathbf{y}$ in $\mathbb{R}^{Vp}$ we can write

$$F(\mathbf{y}) \leq F(\hat{\mathbf{y}}) + \nabla F(\hat{\mathbf{y}})^T(\mathbf{y}-\hat{\mathbf{y}}) + \frac{2(1-\delta)+\alpha M}{2}\|\mathbf{y}-\hat{\mathbf{y}}\|^2. \tag{5.83}$$

According to (5.52), we can substitute $1/(2(1-\delta)+\alpha M)$ by λ. Applying this substitution into (5.83) and minimizing the both sides of (5.83) with respect to $\mathbf{y}$ yields

$$F(\mathbf{y}^*) \leq \; F(\hat{\mathbf{y}}) - \lambda \|\nabla F(\hat{\mathbf{y}})\|^2. \tag{5.84}$$

Since (5.84) holds for any $\hat{\mathbf{y}}$, we set $\hat{\mathbf{y}} := \mathbf{y}_{t-1}$. By rearranging the terms and taking their square roots, we obtain an upper bound for the gradient norm $\|\nabla F(\mathbf{y}_{t-1})\| = \|\mathbf{g}_{t-1}\|$ as

$$\|\mathbf{g}_{t-1}\| \leq \left[\lambda^{-1}[F(\mathbf{y}_{t-1}) - F(\mathbf{y}^*)]\right]^{\frac{1}{2}}. \tag{5.85}$$

The result in Theorem 7 implies that $\|\mathbf{g}_{t-1}\|^{1/2}$ is upper bounded by

$$\|\mathbf{g}_{t-1}\|^{\frac{1}{2}} \leq \left[\lambda^{-1}(1-\zeta)^{t-1}(F(\mathbf{y}_0) - F(\mathbf{y}^*))\right]^{\frac{1}{4}}. \tag{5.86}$$

Consider the definition of Γ_2 in (5.63) and substitute the upper bound in (5.86) for $\|\mathbf{g}_{t-1}\|^{1/2}$ to update (5.82) as

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_{t+1}\| \leq \left(1-\epsilon+\epsilon\rho^{K+1}\right)\left[1 + C_2(1-\zeta)^{\frac{t-1}{4}}\right]\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\| + \epsilon^2\Gamma_2\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\|^2, \tag{5.87}$$

where $C_2 := C_1[(F(\mathbf{y}_0) - F(\mathbf{y}^*))/\lambda]^{1/4}$. Based on the definitions of C_2 and Γ_1 we obtain that $C_2 = \Gamma_1$. This observation in association with (5.87) leads to the claim in (5.62). ■

As per Lemma 17 the weighted gradient norm $\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\|$ is upper bounded by terms that are linear and quadratic on the weighted norm $\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$ associated with the previous iterate. This is akin to the gradient norm decrease of Newton's method with constant stepsize. Note that if the error of Hessian inverse approximation which is characterized by ρ^{K+1} becomes zero, by setting $\epsilon = 1$ we can simplify (5.62) as $\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\| \leq \Gamma_2\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2$. This result shows quadratic convergence when $\Gamma_2\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\| < 1$. However, the term ρ^{K+1} is not zero in general. Although, the error of Hessian inverse approximation is not zero, the result in (5.62) is very similar to the one for the classic Newton's method. To make this connection clearer, further note that for all except the first few iterations the term $\Gamma_1(1-\zeta)^{(t-1)/4} \approx 0$ is close to 0 and the relation in (5.62) can be simplified to

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{g}_{t+1}\| \lesssim (1-\epsilon+\epsilon\rho^{K+1})\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\| + \epsilon^2\Gamma_2\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{g}_t\|^2. \tag{5.88}$$

In (5.88), the coefficient in the linear term is reduced to $(1-\epsilon+\epsilon\rho^{K+1})$ and the coefficient in the quadratic term stays at $\epsilon^2\Gamma_2$. If, for discussion purposes, we set $\epsilon = 1$ as in Newton's quadratic phase, the upper bound in (5.88) is further reduced to

$$\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\| \; \lesssim \rho^{K+1}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\| + \Gamma_2\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2. \tag{5.89}$$

The equation in (5.89) makes the connection between NN and Newton's clear, because the exact same result would hold for Newton's method if we set $\rho = 0$. The NN method can not have a quadratic convergence phase for the rest of the iterations – like the one for Newton's method – because of the term $\rho^{K+1}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$. However, since the constant ρ (cf. Proposition 5) is smaller than 1 the term ρ^{K+1} can be made arbitrarily small by increasing the approximation order K. Equivalently, this means that by selecting K to be large enough, we can make the quadratic term in (5.89) dominant and observe a quadratic convergence phase. The boundaries of this quadratic convergence phase are formally determined in the following Theorem using the result in (5.62).

Theorem 8 *Consider the NN-K method as defined in* (5.12)-(5.17)*. Define the sequence* $\eta_t := [(1-\epsilon+\epsilon\rho^{K+1})(1+\Gamma_1(1-\zeta)^{(t-1)/4})]$ *and the time* t_0 *as the first time at which sequence* η_t *is smaller than 1, i.e.* $t_0 := \text{argmin}_t\{t \mid \eta_t < 1\}$*. If Assumptions 9-11 hold, then for all* $t \geq t_0$ *when the sequence* $\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$ *satisfies*

$$\frac{\sqrt{\eta_t}(1-\sqrt{\eta_t})}{\epsilon^2\Gamma_2} \leq \|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\| < \frac{1-\sqrt{\eta_t}}{\epsilon^2\Gamma_2}\ , \tag{5.90}$$

the sequence of scaled gradient norms is such that

$$\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\| \leq \frac{\epsilon^2\Gamma_2}{1-\sqrt{\eta_t}}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2. \tag{5.91}$$

Proof: Based on the definition of η_t, we can rewrite (5.62) as

$$\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\| \leq \eta_t\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\| + \epsilon^2\Gamma_2\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2. \tag{5.92}$$

We use this expression to prove the inequality in (5.91). To do so, rearrange terms in the first inequality in (5.90) and write

$$\sqrt{\eta_t} \ \leq \ \frac{\epsilon^2\Gamma_2}{1-\sqrt{\eta_t}}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|. \tag{5.93}$$

Multiplying both sides of (5.93) by $\sqrt{\eta_t}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$ yields

$$\eta_t\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\| \leq \frac{\sqrt{\eta_t}\epsilon^2\Gamma_2}{1-\sqrt{\eta_t}}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2. \tag{5.94}$$

Substituting $\eta_t \|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$ in (5.92) by its upper bound in (5.94) implies that

$$\begin{aligned}\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\| &\leq \frac{\sqrt{\eta_t}\epsilon^2\Gamma_2}{1-\sqrt{\eta_t}}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2 + \epsilon^2\Gamma_2\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2 \\ &= \frac{\epsilon^2\Gamma_2}{1-\sqrt{\eta_t}}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2. \end{aligned} \tag{5.95}$$

To verify quadratic convergence, it is necessary to prove that the sequence $\|\mathbf{D}_{i-1}^{-1/2}\mathbf{g}_v\|$ of weighted gradient norms is decreasing. For this to be true we must have

$$\frac{\epsilon^2\Gamma_2}{1-\sqrt{\eta_t}}\left\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\right\| < 1. \tag{5.96}$$

But (5.96) is true because we are looking at a range of gradients that satisfy the second inequality in (5.90). ■

As per Theorem 7, $\mathbf{y}_t$ converges to $\mathbf{y}^*$ at a rate that is at least linear. Thus, the gradients $\mathbf{g}_t$ will be such that at some point in time they satisfy the rightmost inequality in (5.90). At that point in time, progress towards $\mathbf{y}^*$ proceeds at a quadratic rate as indicated by (5.91). This quadratic rate of progress is maintained until the leftmost inequality in (5.90) is satisfied, at which point the linear term in (5.62) dominates and the convergence rate goes back to linear. Furthermore, making K sufficiently large it is possible to reduce η_t arbitrarily and make the quadratic convergence region last longer. In practice, this calls for making K large enough so that $\sqrt{\eta_t}$ is close to the desired gradient norm accuracy.

Remark 6 Making ρ^{K+1} small reduces the factor in front of the linear term in (5.89) and makes the quadratic phase longer. This factor, as it follows from the definition in Proposition 5, is $\rho^{K+1} = [2(1-\delta)/(2(1-\delta)+\alpha m)]^{K+1}$. Thus, other than increasing K, we can make ρ small by increasing the product αm. That implies making the inverse penalty coefficient α large relative to the smallest Hessian eigenvalue of the local functions f_v [cf. (5.21)]. This is not possible if we want to keep the solution $\mathbf{y}^*$ of (5.6) close to the solution of $\tilde{\mathbf{y}}^*$ of (5.9). This calls for the use of adaptive rules to decrease the inverse penalty coefficient α as we elaborate in Section 5.5. Further observe that ρ is independent of the condition number M/m of the local objectives. Making ρ small is an algorithmic choice which is controlled by the selection of α and K, and not a property of the function being minimized.

Remark 7 For a quadratic function F, the Lipschitz constant for the Hessian is $L = 0$. Then, the optimal choice of stepsize for NN-K is $\epsilon = 1$ as a result of stepsize rule in (5.59). Moreover, the constants for the linear and quadratic terms in (5.62) are $\Gamma_1 = \Gamma_2 = 0$ as it follows from their definitions in (5.63). For quadratic functions we also have that the

Hessian of the objective function $\mathbf{H}_t = \mathbf{H}$ and the block diagonal matrix $\mathbf{D}_t = \mathbf{D}$ are time-invariant. Thus, we can rewrite (5.62) as

$$\|\mathbf{D}^{-1/2}\mathbf{g}_{t+1}\| \leq \rho^{K+1}\|\mathbf{D}^{-1/2}\mathbf{g}_t\|. \tag{5.97}$$

Note that Newton's method converges in a single step in quadratic programming. This property follows from (5.97) because Newton's method is equivalent to NN-K as $K \to \infty$. The expression in (5.97) states that NN-K converges linearly with a constant decrease factor of ρ^{K+1} per iteration. This in contrast with first order methods like DGD that converge with a linear rate that depends on the problem condition number.

5.5 Implementation details

As mentioned in Section 5.2, NN-K does not solve (5.1) or its equivalent (5.9), but the penalty version in (5.6). The optimal solutions of the optimization problems in (5.9) and (5.6) are different and the gap between them is of order $O(\alpha)$, [126]. This observation implies that by setting a decreasing policy for α, or equivalently, an increasing policy for the penalty coefficient $1/\alpha$, the solution of (5.9) approaches the minimizer of (5.6), i.e. $\tilde{\mathbf{y}}^* \to \mathbf{y}^*$ for $\alpha \to 0$. There are various possible alternatives to reduce α. Given the penalty method interpretation in Section 5.2 it is more natural to consider fixed penalty parameters α that are decreased after detecting convergence to the optimum argument of the function $F(\mathbf{y})$ [cf. (5.6)]. This latter idea is summarized under the name of Adaptive Network Newton-K (ANN-K) in Algorithm 7 where α is reduced by a given factor $\eta < 1$.

Specifically, ANN-K relies on Algorithm 6, which receives an initial iterate $\mathbf{w}_v$, a penalty parameter α, and a given tolerance tol (Step 1) and runs the local NN-K iteration in (5.19) for node v until the local gradient norm $\|\mathbf{g}_v\|$ becomes smaller than tol (Step 13). The descent iteration is implemented in Step 12. Implementation of this descent requires access to the NN-K descent direction $\mathbf{d}_{v,t}^{(K)}$ which is computed by the loop in steps 7-11. Step 7 initializes the loop by computing the NN-0 step $\mathbf{d}_{v,t}^{(0)} = -\mathbf{D}_{vv,t}^{-1}\mathbf{g}_{v,t}$. The core of the loop is in Step 10 which corresponds to the recursion in (5.19). Step 8 stands for the variable exchange that is necessary to implement Step 7. After K iterations through this loop, the NN-K descent direction $\mathbf{d}_{v,t}^{(K)}$ is computed and can be used in Step 12. Both, steps 7 and 10, require access to the local gradient component $\mathbf{g}_{v,t}$. This is evaluated in Step 6 after receiving the prerequisite information from neighbors in Step 5. Steps 3 and 4 compute the blocks $\mathbf{B}_{ii,t}$, $\mathbf{B}_{ij,t}$, and $\mathbf{D}_{vv,t}$ that are also necessary in steps 7 and 10. This process is repeated until $\|\mathbf{g}_v\| <$ tol (Step 13).

ANN-K calls Algorithm 6 in Step 2 of Algorithm 7. The factor α is subsequently reduced by the factor $\eta < 1$ as indicated in Step 6 of Algorithm 7 that implements the replacement

Algorithm 6 Network Newton-K method at node v

1: **function** $\mathbf{w}_v = \text{NN-}K(\alpha, \mathbf{w}_v, \text{tol})$
2: **repeat**
3: $\quad$ **B** matrix blocks: $\mathbf{B}_{vv} = (1 - w_{vv})\mathbf{I}$ and $\mathbf{B}_{vu} = w_{uv}\mathbf{I}$
4: $\quad$ **D** matrix block: $\mathbf{D}_{vv} = \alpha\nabla^2 f_v(\mathbf{w}_v) + 2(1 - w_{vv})\mathbf{I}$
5: $\quad$ Exchange iterates $\mathbf{w}_v$ with neighbors $u \in \mathcal{N}_v$.
6: $\quad$ Gradient: $\mathbf{g}_v = (1 - w_{vv})\mathbf{w}_v - \sum_{u\in\mathcal{N}_v} w_{vu}\mathbf{w}_u + \alpha\nabla f_v(\mathbf{w}_v)$.
7: $\quad$ Compute NN-0 descent direction $\mathbf{d}_v^{(0)} = -\mathbf{D}_{vv}^{-1}\mathbf{g}_v$
8: $\quad$ **for** $k = 0, \ldots, K-1$ **do**
9: $\quad\quad$ Exchange elements $\mathbf{d}_i^{(k)}$ of the NN-k step with neighbors
10: $\quad\quad$ NN-$(k+1)$ step: $\mathbf{d}_v^{(k+1)} = \mathbf{D}_{vv}^{-1}\Big[\sum_{u\in\mathcal{N}_v, u=v} \mathbf{B}_{vu}\mathbf{d}_u^{(k)} - \mathbf{g}_v\Big]$.
11: $\quad$ **end for**
12: $\quad$ Update local iterate: $\mathbf{w}_v = \mathbf{w}_v + \epsilon\ \mathbf{d}_v^{(K)}$.
13: **until** $\|\mathbf{g}_v\| < \text{tol}$

Algorithm 7 Adaptive Network Newton-K method at node v

Require: Iterate $\mathbf{w}_v$. Initial parameter α. Flags $s_{vu} = 0$. Factor $\eta < 1$.
1: **for** $t = 0, 1, 2, \ldots$ **do**
2: $\quad$ Call NN-K function: $\mathbf{w}_v = \text{NN-}K(\alpha, \mathbf{w}_v, \text{tol})$
3: $\quad$ Set $s_{vv} = 1$ and broadcast it to all nodes.
4: $\quad$ Set $s_{vu} = 1$ for all nodes u that sent the signal $s_{uu} = 1$.
5: $\quad$ **if** $s_{vu} = 1$ for all $u = 1, \ldots, V$ **then**
6: $\quad\quad$ Update penalty parameter $\alpha = \eta\alpha$.
7: $\quad\quad$ Set $s_{vu} = 0$ for all $u = 1, \ldots, V$.
8: $\quad$ **end if**
9: **end for**

$\alpha = \eta\alpha$. The rest of Algorithm 7 is designed to handle the fact that a small local gradient norm does not necessarily imply a small global gradient norm. To handle this possible mismatch, flag variables s_{vu} are introduced at node v to signal the fact that node u has reached a local gradient $\mathbf{g}_u$ with norm $\|\mathbf{g}_u\| \leq$ tol. Whenever node v completes a run of Algorithm 6 it broadcasts the signal s_{vv} to all other nodes (Step 3) and updates the variables s_{vu} to $s_{vu} = 1$ for all the nodes that sent the signals $s_{uu} = 1$ while Algorithm 6 was executing (Step 4). If all the variables $s_{vu} = 1$ (Step 5) it must be that this is true for all nodes and it is thus safe to modify α (Step 6). The flag variables are reset to $s_{vu} = 0$ and Algorithm 6 is called with the reduced α.

As is typical of penalty methods there are tradeoffs on the selection of the initial value of α and the decrease factor η. Small values of the initial penalty parameter and α and factor η results in sequence of approximate problems having solutions $\tilde{\mathbf{y}}^*$ that are closer to the actual solution $\mathbf{y}^*$. However, problems with small α may take a large number of iterations to converge if initialized far from the optimum value because the constant ρ approaches 1 when

α is small – as we discussed in Remark 6. It is therefore better to initialize Algorithm 7 with values of α that are not too small and to decrease α by a factor η that is not too aggressive. We discuss the details in the numerical examples of Section 5.6.5.

5.6 Numerical analysis

In this section, we study the performance of NN-K in the minimization of a distributed quadratic objective. For each agent v we consider a positive definite diagonal matrix $\mathbf{A}_v \in \mathbb{S}_p^{++}$ and a vector $\mathbf{b}_v \in \mathbb{R}^p$ to define the local objective function $f_v(\mathbf{w}) := (1/2)\mathbf{w}^T\mathbf{A}_v\mathbf{w} + \mathbf{b}_v^T\mathbf{w}$. Therefore, the global cost function $f(\mathbf{w})$ is written as

$$f(\mathbf{w}) := \sum_{v=1}^{V} \frac{1}{2}\mathbf{w}^T\mathbf{A}_v\mathbf{w} + \mathbf{b}_v^T\mathbf{w} \ . \tag{5.98}$$

The difficulty of solving (5.98) is given by the condition number of the matrices $\mathbf{A}_v$. To tune condition numbers we generate diagonal matrices $\mathbf{A}_v$ with random diagonal elements a_{vv}. The first $p/2$ diagonal elements a_{vv} are drawn uniformly at random from the discrete set $\{1, 10^{-1}, \ldots, 10^{-\xi}\}$ and the next $p/2$ are uniformly and randomly chosen from the set $\{1, 10^{1}, \ldots, 10^{\xi}\}$. This choice of coefficients yields local matrices $\mathbf{A}_v$ with eigenvalues in the interval $[10^{-\xi}, 10^{\xi}]$ and global matrices $\sum_{v=1}^{V}\mathbf{A}_v$ with eigenvalues in the interval $[n10^{-\xi}, n10^{\xi}]$. The linear terms $\mathbf{b}_v^T\mathbf{w}$ are added so that the different local functions have different minima. The vectors $\mathbf{b}_v$ are chosen uniformly at random from the box $[0,1]^p$.

The graph is d-regular and generated by creating a cycle and then connecting each node with $d/2$ nodes that are closest in each direction. The diagonal weights in $\mathbf{W}$ are set to $w_{vv} = 1/2 + 1/2(d+1)$ and the off diagonal weights to $w_{uv} = 1/2(d+1)$ when $u \in \mathcal{N}_v$. In our comparison the relative error is defined as the ratio $\|\mathbf{w}^t - \mathbf{w}^*\|^2/\|\mathbf{w}^0 - \mathbf{w}^*\|^2$.

5.6.1 Comparison with existing methods

In this section we compare the performance of the proposed NN method with primal methods such as DGD in [80] and the accelerated version of DGD (Acc. DGD) in [44]. For the Acc. DGD method, we assume that the stepsize parameter and the momentum coefficients are constant as in the case for the centralized accelerated gradient descent. This makes the comparison between Acc. DGD, DGD, and NN fair, since our aim is to compare their performances in solving the penalized objective function. Moreover, we consider the convergence paths of the distributed ADMM (DADMM) in [112] and the exact first order method EXTRA in [111]. Although EXTRA operates in the primal domain, it has been shown that it can be interpreted as a saddle-point method [73]. Thus, we consider EXTRA in the category of dual methods which has a linear convergence rate as DADMM.

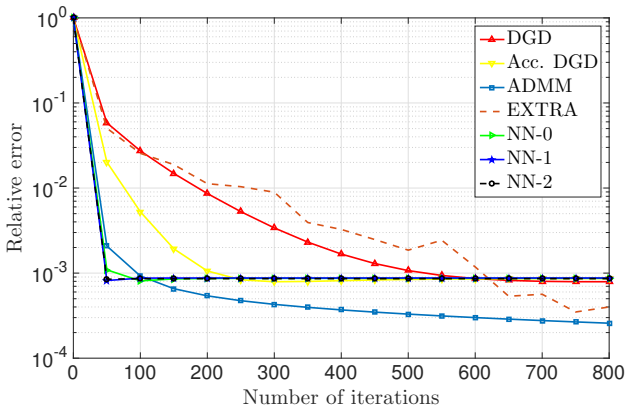


Figure 5.1: Comparison of DGD, Acc. DGD, DADMM, EXTRA, NN-0, NN-1, and NN-2 in terms of number of iterations.

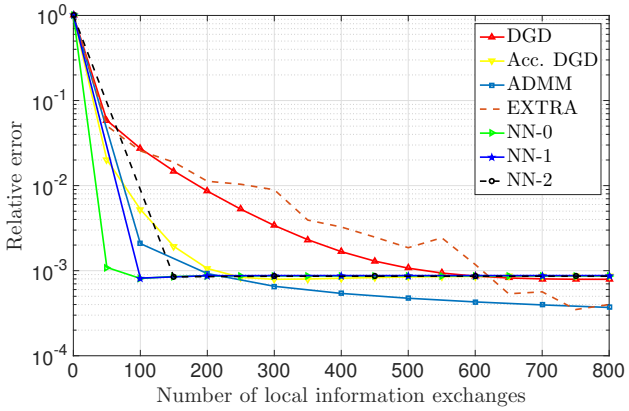


Figure 5.2: Comparison of DGD, Acc. DGD, DADMM, EXTRA, NN-0, NN-1, and NN-2 in terms of rounds of local information exchanges.

We compare these methods in solving (5.98) for the case that there are $n = 100$ nodes in the network and the dimension of the vector $\mathbf{w}$ is $p = 20$. We assume that the graph is 4-regular. Further, we set the condition number parameter to $\xi = 2$ and the penalty parameter to $\alpha = 10^{-3}$. The momentum coefficient for the accelerated DGD is 0.9. Note that among the values $\{0.1, 0.2, \ldots, 0.9, 1\}$, the best performance belongs to the momentum coefficient 0.9 which we use in the experiments.

As the condition number of the problem is relatively large, i.e., 4.3×10^3, the NN method performs better than DGD and Acc. DGD in terms of the number of iterations and total number of local information exchanges as they are illustrated in Fig. 5.1 and Fig. 5.2, respectively. In the case that the condition number of the objective function is not significantly large with respect to the dimension of the problem, the accelerated DGD would be a better choice relative to NN.

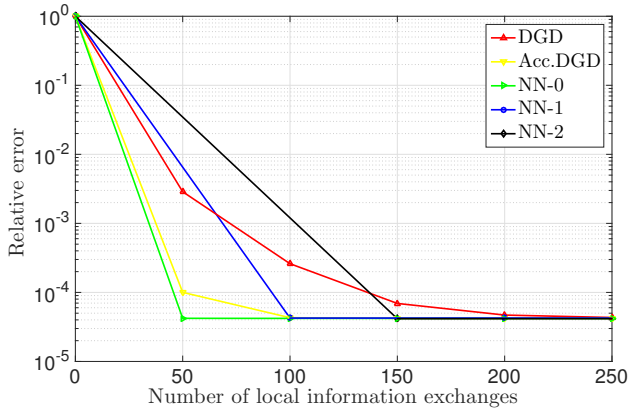


Figure 5.3: Relative error of DGD, Acc. DGD, NN-0, NN-1, and NN-2 vs number of local info. exchanges for a well-conditioned problem.

The comparison with dual methods shows that in terms of iterations and rounds of communications DADMM and different variants of NN perform relatively well and after some point DADMM outperform NN and other primal methods because it converges to the optimal argument of the original problem instead of the penalized function. On the other hand, each step of DADMM requires solving a convex program which can be computationally costly. We observe that EXTRA also has a linear convergence rate to the exact optimal solution, and its accuracy becomes better than all primal methods. However, EXTRA is a first-order method and its convergence at the beginning is relatively slower than NN. This advantage of NN results from incorporation of the curvature information of the objective function. These observations show that by incorporating the idea of NN and EXTRA we should be able to come up with a second-order method that has a linear convergence rate to the exact solution of (5.98) while it can perform well in ill-conditioned problems.

5.6.2 Effect of objective function condition number

We study the effect of condition number on the convergence rate of NN and show that NN is less sensitive to the objective function condition number with respect to primal first-order methods, e.g., DGD in [80] and accelerated DGD in [44]. To do so, we compare the performances of the mentioned methods in solving the problem in (5.98) for small and large condition numbers. The parameters are the same as the parameters in Fig. 5.1 except the choice of the condition number parameter ξ.

We first consider the case that $\xi = 1$ which leads to condition number 1.24×10^1. The convergence paths of DGD, accelerated DGD, NN-0, NN-1, and NN-2 in terms of the number of local information exchanges are shown in Fig. 5.3. The performance of variations of NN are not significantly better than DGD and accelerated DGD. In particular, DGD and

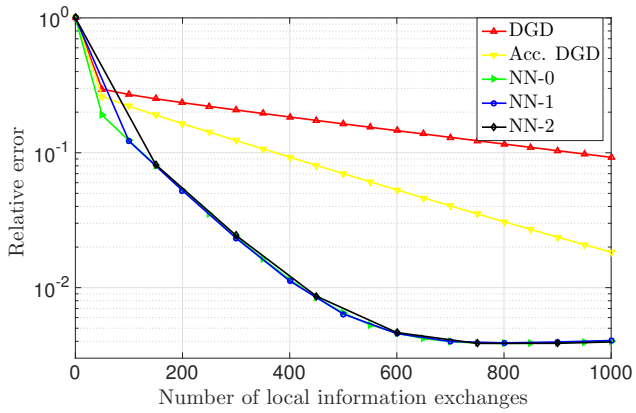


Figure 5.4: Relative error of DGD, Acc. DGD, NN-0, NN-1, and NN-2 vs number of local info. exchanges for an ill-conditioned problem

Acc. DGD both outperform NN-1 and NN-2 in terms of the total communications until convergence. Thus, accelerated DGD is the best option among the primal methods for problems with small condition number.

To explore the performance of these methods for an ill-conditioned problem we set the condition number parameter $\xi = 3$ which leads to the condition number 1.4×10^4 for the considered realization. Fig. 5.4 illustrate the convergence paths of the considered primal methods in terms of the number of local information exchanges. As we observe, the advantage of the network Newton methods is substantial in this setting and they outperform DGD and accelerated DGD in terms of communication cost.

5.6.3 Effect of network topology

We proceed to compare the performance of NN in different network topologies. In particular, we consider five different topologies which are random graphs with connectivity probabilities $p_c = 0.25$ and $p_c = 0.35$, complete graph, cycle, and line. Note that in random graphs, we generate the edges between nodes with probability p_c. The complete graph is a graph that all nodes are connected to each other directly. A cycle graph is a connected graph that each node has degree 2. A line graph is a cycle graph that is missing an edge. The parameters are the same as the parameters in Fig. 5.1 except the network graph and the way that we generate the weight matrix $\mathbf{W}$. We generate the weight matrix $\mathbf{W}$ using the formula $\mathbf{W} = \mathbf{I} - \mathbf{L}/\tau$ where $\mathbf{L}$ is the Laplacian matrix of the graph and $\tau/2$ is the largest eigenvalue of the Laplacian $\mathbf{L}$. We compare the performance of NN-2 for all these networks in terms of the number of iterations and the total number of communications between nodes. Notice that in this section we use total communications between node instead of the number of local information exchanges (rounds of local communications) since the degrees of nodes in

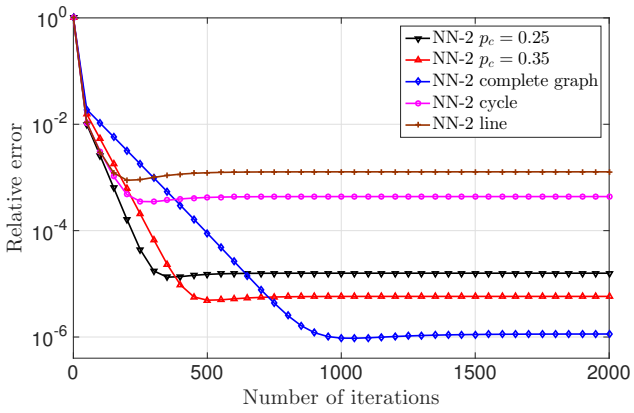


Figure 5.5: Relative error of NN-2 vs num. of iterations for random graphs with $p_c = \{0.25, 0.35\}$, complete graph, cycle graph, and line graph.

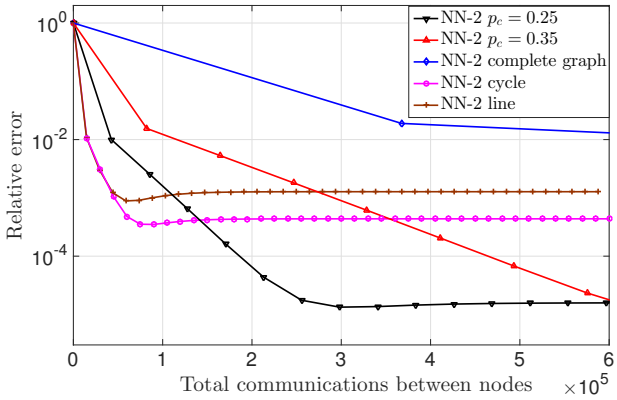


Figure 5.6: Relative error of NN-2 vs num. of communications for random graphs with $p_c = \{0.25, 0.35\}$, complete graph, cycle graph, and line graph.

the different networks are not equal.

The convergence paths of NN-2 for the considered topologies in terms of the number of iterations and the total number of communications are demonstrated in Fig. 5.5 and Fig 5.6, respectively. The first important observation is the accuracy of convergence. According to the results in [126], if we define $\beta < 1$ as the second largest magnitude of the eigenvalues of $\mathbf{W}$, then the accuracy of convergence is proportional to $1/(1-\beta)$. Thus, the graphs with smaller β converge to a smaller neighborhood of the optimal argument. In particular, the parameter β for the complete graph which has the most accurate convergence is $\beta = 0.5$, while for the line graph that has the least accurate convergence path $\beta = 0.99$.

The second important observation is the rate of convergence for NN-2 in these network topologies. It follows from the result in Theorem 7 that for a quadratic objective function the constant of linear convergence becomes $1 - \alpha m \lambda$. Therefore, for larger values of λ we

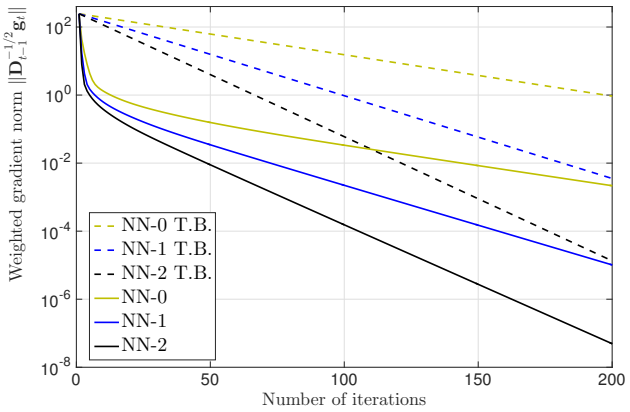


Figure 5.7: Comparison of the theoretical bound (T.B.) in (5.97) with the empirical result for a quadratic programming.

expect faster convergence. Note that λ is large when $\delta = \min_i w_{vv}$ is large and close to 1. These observations imply that for the graphs that δ is larger we expect faster linear convergence. The convergence paths in Fig 5.5 reinforce this claim. Note that δ for the considered graphs are $\delta_{p_c=0.25} = 0.5898$, $\delta_{p_c=0.35} = 0.5585$, $\delta_{com} = 0.51$, $\delta_{cycle} = 0.75$, $\delta_{line} = 0.7498$. These numbers justify the similarity of the convergence paths of line and cycle graphs and the slow convergence rate of the complete graph.

5.6.4 Tightness of the bounds

In this section, we study the tightness of the theoretical bounds in the chapter. To do so, we compare the empirical convergence rates of NN-0, NN-1, and NN-2 with the theoretical result in Lemma 17. As we discussed in Remark 3, for a quadratic objective function the sequence of weighted gradients of NN-K satisfies the inequality $\|\mathbf{D}^{-1/2}\mathbf{g}_{t+1}\| \leq \rho^{K+1}\|\mathbf{D}^{-1/2}\mathbf{g}_t\|$. We refer to this rate as T.B. which stands for theoretical bound. Figure 5.7 illustrates the theoretical bounds and empirical convergence paths of NN-0, NN-1, and NN-2 for the quadratic problem in (5.98). As we observe, the convergence rates of all methods are faster than their theoretical bounds at the beginning, but after almost 10 iterations their convergence rate becomes similar to the theoretical bound in (5.97). To be clearer, the slopes of the actual convergence paths and their corresponding theoretical bounds become equal after almost 10 iterations. This observation shows that the bound in (5.97) is reasonably tight and the sequence of weighted gradients for NN-K diminishes with factor ρ^{K+1}.

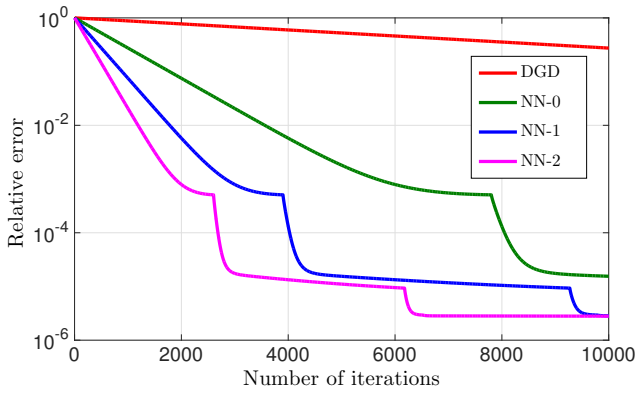


Figure 5.8: Convergence of adaptive DGD, ANN-0, ANN-1, and ANN-2 for $\alpha_0 = 10^{-2}$. Network Newton methods require less iterations than DGD.

5.6.5 Adaptive network Newton

Given that DGD and network Newton are penalty methods it is of interest to consider their behavior when the inverse penalty coefficient α is decreased recursively. The adaptation of α for NN-K is discussed in Section 5.5 where it is termed adaptive (A)NN-K. The same adaptation strategy is considered here for DGD. The parameter α is kept constant until the local gradient components $\mathbf{g}_{v,t}$ become smaller than a given tolerance tol, i.e., until $\|\mathbf{g}_{v,t}\| \leq \text{tol}$ for all v. When this tolerance is achieved, the parameter α is scaled by a factor $\eta < 1$, i.e., α is decreased from its current value to $\eta\alpha$. This requires the use of a signaling method like the one summarized in Algorithm 7 for ANN-K.

We consider the objective in (5.98) and nodes connected by a d-regular cycle. We use the same parameters used to generate Fig. 5.1. The adaptive gradient tolerance is set to $\text{tol} = 10^{-3}$ and the scaling parameter to $\eta = 0.1$. We consider two different scenarios where the initial penalty parameters are $\alpha = \alpha_0 = 10^{-1}$ and $\alpha = \alpha_0 = 10^{-2}$. The respective error trajectories e_t with respect to the number of iterations are shown in figures 5.8 – where $\alpha_0 = 10^{-2}$ – and 5.9 – where $\alpha_0 = 10^{-1}$. In each figure we show e_t for adaptive DGD, ANN-0, ANN-1, and ANN-2. Both figures show that the ANN methods outperform adaptive DGD and that larger K reduces the number of iterations that it takes ANN-K to achieve a target error.

Note that for different ANN methods and adaptive DGD we need an extra cost of updating the parameters s_{vu} which is not very expensive. This is true since the this process requires a *binary* communication that happens every time that nodes update the parameter α. As we observe in the plots, nodes update α only 3 or 4 times and the extra communication cost is not substantial. We have not included this extra cost in the figures, but adding this extra cost doesn't change the conclusion that ANN outperforms adaptive DGD.

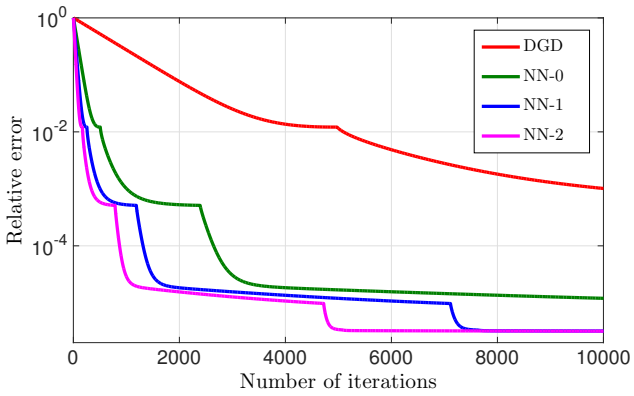


Figure 5.9: Convergence of Adaptive DGD, ANN-0, ANN-1, and ANN-2 for $\alpha_0 = 10^{-1}$. ANN methods require less iterations and convergence of all algorithms are faster relative to the case that $\alpha_0 = 10^{-2}$.

More interesting conclusions follow from a comparison across figures 5.8 and 5.9. We can see that it is better to start with the (larger) value $\alpha = 10^{-1}$ even if the method initially converges to a point farther from the actual optimum. This happens because increasing α decreases ρ in (5.35).

5.6.6 Logistic regression

We consider the application of NN for solving a logistic regression problem. In this problem we are given q training samples that we distribute across V distinct servers. Denote q_v as the number of samples assigned to server v. Each of the training samples at node v contains a feature vector $\mathbf{x}_{vi} \in \mathbb{R}^p$ and a class $y_{vi} \in \{-1, 1\}$. The goal is to predict the probability $\mathrm{P}(y = 1 \mid \mathbf{x})$ of having label $y = 1$ when given a feature vector $\mathbf{x}$ whose class is unknown. The logistic regression model assumes that this probability can be computed as $\mathrm{P}(y = 1 \mid \mathbf{x}) = 1/(1 + \exp(-\mathbf{x}^T\mathbf{w}))$ for a linear classifier $\mathbf{w}$ that is computed based on the training samples. It follows from this model that the regularized maximum log likelihood estimate of the classifier $\mathbf{w}$ given the training samples $(\mathbf{z}_{vi}, y_{vi})$ for $i = 1, \ldots, q_v$ and $v = 1, \ldots, V$ is given by

$$\begin{aligned} \mathbf{w}^* &= \operatorname*{argmin}_{\mathbf{w}} f(\mathbf{w}) \\ &:= \operatorname*{argmin}_{\mathbf{w}} \frac{\zeta}{2}\|\mathbf{w}\|^2 + \sum_{v=1}^{V}\sum_{i=1}^{q_v} \log\left[1 + \exp(-y_{vi}\mathbf{x}_{vi}^T\mathbf{w})\right], \end{aligned} \tag{5.99}$$

where we defined the function $f(\mathbf{w})$ for future reference. The regularization term $(\zeta/2)\|\mathbf{w}\|^2$ is added to reduce overfitting to the training set. The optimization problem in (5.99) can

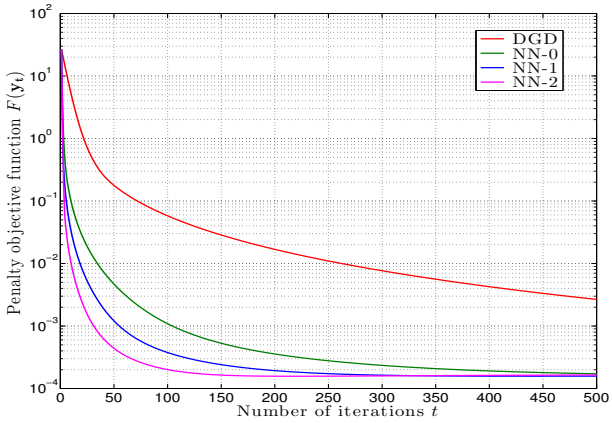


Figure 5.10: Convergence of DGD, NN-0, NN-1, and NN-2. network Newton methods for a linearly separable logistic regression.

be written in the form of the optimization problem in (5.1). To do so simply define the local objective functions f_v as

$$f_v(\mathbf{w}) = \frac{\zeta}{2n}\|\mathbf{w}\|^2 + \sum_{i=1}^{q_v} \log\Big[1 + \exp(-y_{vi}\mathbf{x}_{vi}^T\mathbf{w})\Big], \tag{5.100}$$

and observe that given this definition we can write the objective in (5.99) as $f(\mathbf{w}) = \sum_{v=1}^{V} f_v(\mathbf{w})$. We can then solve (5.99) in a distributed manner using DGD and NN-K methods.

We use a synthetic dataset where each component of the feature vector $\mathbf{x}_{vi}$ with label $y_{vi} = 1$ is generated from a normal distribution with mean μ and standard deviation σ_+, while sample points with label $y_{vi} = -1$ are generated with mean $-\mu$ and standard deviation σ_-. The network is a d-regular cycle with $d = 4$ and has $n = 100$ nodes. The diagonal weights in the matrix $\mathbf{W}$ are set to $w_{vv} = 1/2 + 1/2(d+1)$ and the off diagonal weights to $w_{vu} = 1/2(d+1)$ when $u \in \mathcal{N}_v$. We set the feature vector dimension to $p = 10$, the number of training samples per node at $q_v = 50$, and the regularization parameter to $\zeta = 10^{-4}$.

We consider first a scenario in which the dataset is linearly separable. To generate a linearly separable dataset the mean is set to $\mu = 3$ and the standard deviations to $\sigma_+ = \sigma_- = 1$. Fig. 5.10 illustrates the convergence path of the objective function $F(\mathbf{y})$ [cf. (5.6)] when the penalty parameter is $\alpha = 10^{-2}$ and the network Newton step size is $\epsilon = 1$ – line search methods for distributed optimization methods exist [127], but we found them unnecessary in the numerical experiments presented here. The reduction in the number of iterations required to achieve convergence is a little more marked than in the considered quadratic example. The objective function values $F(\mathbf{y}_t)$ for NN-0, NN-1, and NN-2 after

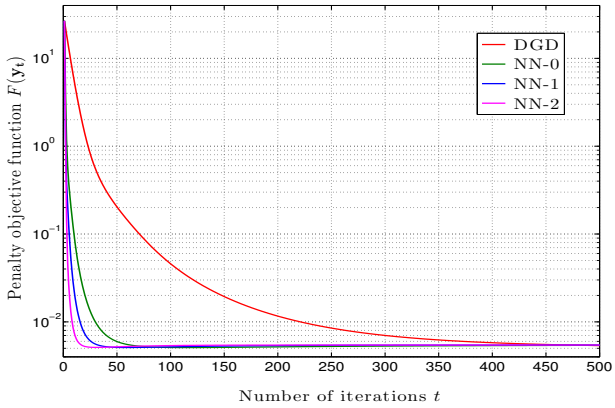


Figure 5.11: Convergence of DGD, NN-0, NN-1, and NN-2. network Newton methods for a non-linearly separable logistic regression.

$t = 500$ iterations are below 1.6×10^{-4}, while for DGD the objective function value after the same number of iterations have passed is $F(\mathbf{y}_t) = 2.6 \times 10^{-3}$. Conversely, achieving accuracy $F(\mathbf{y}_t) = 2.6 \times 10^{-3}$ for NN-0, NN-1, and NN-2 requires 68, 33, and 19 iterations, respectively, while DGD requires 500 iterations. Observe that for this example NN-2 performs better than NN-1 and NN-0 not only in the number of iterations but also in the number of variable exchanges required to achieve a target accuracy. To clarify this advantage, note that each iteration of NN-K requires a total of $K+1$ local information exchanges. Hence, NN-0, NN-1, NN-2, and DGD reach the objective function value $F(\mathbf{y}_t) = 2.6 \times 10^{-3}$ after $68 \times 1 = 68$, $33 \times 2 = 66$, $19 \times 3 = 57$, and $500 \times 1 = 500$ local information exchanges, respectively. It follows that NN-2 performs better than NN-1, NN-0, and DGD in terms of the number of variable exchanges required to achieve a target accuracy.

We also consider a case that the dataset is *not* linearly separable. To generate this dataset we set the mean to $\mu = 2$ and the standard deviations to $\sigma_+ = \sigma_- = 2$. The penalty parameter is $\alpha = 10^{-2}$ and the step size is $\epsilon = 1$. The resulting objective trajectories $F(\mathbf{y}_t)$ of DGD, NN-0, NN-1, and NN-2 are shown in Fig. 5.11. The advantages of the NN methods relative to DGD are less pronounced but still significant.

Chapter 6

Second-order primal-dual method for distributed optimization

6.1 Context and background

In this chapter, we continue studying the idea of solving ERM problems using distributed methods by splitting samples across multiple processors which create a network. As we mentioned in Chapter 5, by assigning q_v samples to each node v and defining f_v as the loss function associated to the samples of node v, the empirical risk minimization problem for a training set of $N = \sum_{v=1}^{V} q_v$ samples boils down to the problem

$$\tilde{\mathbf{w}}^* := \underset{\tilde{\mathbf{w}} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{v=1}^{V} f_v(\tilde{\mathbf{w}}), \tag{6.1}$$

where $\tilde{\mathbf{w}} \in \mathbb{R}^p$ is the optimization variable. Our interest in studying the decentralized problem in (6.1) comes from the idea of solving ERM problem by splitting samples across space; however, this class of problems arises in many application domains such as decentralized control [22, 25, 56], wireless communication [96, 97], and sensor networks [46, 93, 103]. As in Chapter 5, the results in this chapter hold for the general problem formulation in (6.1) irrespective to the domain of application.

Decentralized methods for solving (6.1) can be divided into two classes: primal domain methods and dual domain methods. Decentralized gradient descent (DGD) is a well-established primal method that implements gradient descent on a penalized version of (6.1) whose gradient can be separated into per-node components. Network Newton (NN), presented in Chapter 5, is a more recent alternative that accelerates the convergence of DGD by incorporating second order information of the penalized objective [64, 65]. Both, DGD and NN, solve problem (6.1) in the primal domain, i.e., solve a penalized version of the

original problem, and converge to a neighborhood of the optimal argument $\tilde{\mathbf{w}}^*$ when using a constant stepsize and converge sublinearly to the exact optimal argument if using a diminishing stepsize.

Dual domain methods build on the fact that the dual function of (6.1) has a gradient with separable structure. The use of plain dual gradient descent is possible but generally slow to converge [8, 95, 102]. In centralized optimization, better convergence speeds are attained by the method of multipliers (MM) that adds a quadratic augmentation term to the Lagrangian [7, 40], or the proximal (P)MM that adds an additional term to keep iterates close. In either case, the quadratic term that is added to construct the augmented Lagrangian makes distributed computation of primal gradients impossible. This issue is most often overcome with the use of decentralized (D) versions of the alternating direction method of multipliers (ADMM) [19, 103, 112]. Besides the ADMM, other methods that use different alternatives to approximate the gradients of the dual function have also been proposed [28, 43, 79, 100, 115, 117, 122]. The convergence rates of these methods have not been studied except for the DADMM and its variants that are known to converge linearly to the optimal argument when the local functions are strongly convex and their gradients are Lipschitz continuous [54, 77, 112]. An important observation here is that while all of these methods try to approximate the MM or the PMM, the performance penalty entailed by the approximation has not been studied.

This chapter studies the exact second order method (ESOM) which uses quadratic approximations of the augmented Lagrangians of (6.1) and leads to a set of separable subproblems. Similar to other second order methods, implementation of ESOM requires computation of Hessian inverses. Distributed implementation of this operation is infeasible because while the Hessian of the proximal augmented Lagrangian is neighbor sparse, its inverse is not. ESOM resolves this issue by using the Hessian inverse approximation technique introduced in [64, 65, 128]. This technique consists of truncating the Taylor's series of the Hessian inverse to order K to obtain the family of methods ESOM-K. Implementation of this expansion in terms of local operations is possible. A remarkable property of all ESOM-K methods is that they can be shown to pay a performance penalty relative to (centralized) PMM that vanishes with increasing iterations.

We begin the chapter by reformulating (6.1) in a form more suitable for decentralized implementation (Proposition 7) and proceed to describe the PMM (Section 6.2). ESOM is a variation of PMM that substitutes the proximal augmented Lagrangian with its quadratic approximation (Section 6.3). Implementation of ESOM requires computing the inverse of the Hessian of the proximal augmented Lagrangian. Since this inversion cannot be computed using local and neighboring information, ESOM-K approximates the Hessian inverse with the K-order truncation of the Taylor's series expansion of the Hessian inverse.

This expansion can be carried out using an inner loop of local operations. This and other details required for decentralized implementation of ESOM-K are discussed in Section 6.3.1 along with a discussion of how ESOM can be interpreted as a saddle point generalization of the Network Newton methods proposed in [63] (Remark 2) or a second order version of the EXTRA method in [111] (Remark 3).

Convergence analyses of PMM and ESOM are then presented (Section 6.4). Linear convergence of PMM is established (Section 6.4.1) and linear convergence factors explicitly derived to use as benchmarks (Theorem 9). In the ESOM analysis (Section 6.4.2) we provide an upper bound for the error of the proximal augmented Lagrangian approximation (Lemma 20). We leverage this result to prove linear convergence of ESOM (Theorem 10) and to show that ESOM's linear convergence factor approaches the corresponding PMM factor as time grows (Section 6.4.3). This indicates that the convergence paths of (distributed) ESOM-K and (centralized) PMM are very close. We also study the dependency of the convergence constant with the algorithm's order K.

ESOM tradeoffs and comparisons with other decentralized methods for solving consensus optimization problems are illustrated in numerical experiments (Section 6.5) for a decentralized least squares problem (Section 6.5.1) and a decentralized logistic regression classification problem (Section 6.5.2). Numerical results in both settings verify that larger K leads to faster convergence in terms of number of iterations. However, we observe that all versions of ESOM-K exhibit similar convergence rates in terms of the number of communication exchanges. This implies that ESOM-0 is preferable with respect to the latter metric and that larger K is justified when computational cost is of interest. Faster convergence relative to EXTRA, Network Newton, and DQM is observed.

Notation. Vectors are written as $\mathbf{w} \in \mathbb{R}^p$ and matrices as $\mathbf{A} \in \mathbb{R}^{p\times p}$. Given n vectors $\mathbf{w}_v$, the vector $\mathbf{w} = [\mathbf{w}_1; \dots; \mathbf{w}_V]$ represents a stacking of the elements of each individual $\mathbf{w}_v$. We use $\|\mathbf{w}\|$ and $\|\mathbf{A}\|$ to denote the Euclidean norm of vector $\mathbf{w}$ and matrix $\mathbf{A}$, respectively. The norm of vector $\mathbf{w}$ with respect to positive definite matrix $\mathbf{A}$ is $\|\mathbf{w}\|_{\mathbf{A}} := (\mathbf{w}^T\mathbf{A}\mathbf{w})^{1/2}$. Given a function f its gradient $\mathbf{w}$ is denoted as $\nabla f(\mathbf{w})$ and its Hessian as $\nabla^2 f(\mathbf{w})$.

6.2 Proximal method of multipliers

Let $\mathbf{w}_v \in \mathbb{R}^p$ be a copy of the decision variable $\mathbf{w}$ kept at node v and define $\mathcal{N}_v$ as the neighborhood of node v. Assuming the network is bidirectionally connected, the optimization

problem in (6.1) is equivalent to the program

$$\{\mathbf{w}_v^*\}_{v=1}^V := \underset{\{\mathbf{w}_v\}_{v=1}^V}{\operatorname{argmin}} \sum_{v=1}^V f_v(\mathbf{w}_v), \quad \text{s.t. } \mathbf{w}_v = \mathbf{w}_u, \quad \text{for all } v, u \in \mathcal{N}_v. \tag{6.2}$$

Indeed, the constraint in (6.2) enforces the consensus condition $\mathbf{w}_1 = \cdots = \mathbf{w}_V$ for any feasible point of (6.2). With this condition satisfied, the objective in (6.2) is equal to the objective function in (6.1) from where it follows that the optimal local variables $\mathbf{w}_v^*$ are all equal to the optimal argument $\tilde{\mathbf{w}}^*$ of (6.1), i.e., $\mathbf{w}_1^* = \cdots = \mathbf{w}_V^* = \tilde{\mathbf{w}}^*$.

To derive ESOM define $\mathbf{w} := [\mathbf{w}_1; \ldots; \mathbf{w}_V] \in \mathbb{R}^{Vp}$ as the concatenation of the local decision variables $\mathbf{w}_v$ and the aggregate function $f : \mathbb{R}^{Vp} \to \mathbb{R}$ as $f(\mathbf{w}) = f(\mathbf{w}_1, \ldots, \mathbf{w}_V) := \sum_{v=1}^V f_v(\mathbf{w}_v)$ as the sum of all the local functions $f_v(\mathbf{w}_v)$. Introduce the matrix $\mathbf{W} \in \mathbb{R}^{V \times V}$ with elements $w_{vu} \geq 0$ representing a weight that node v assigns to variables of node u. The weight $w_{vu} = 0$ if and only if $u \notin \mathcal{N}_v \cup \{v\}$. The matrix $\mathbf{W}$ is further required to satisfy

$$\mathbf{W}^T = \mathbf{W}, \quad \mathbf{W}\mathbf{1} = \mathbf{1}, \quad \text{null}(\mathbf{I} - \mathbf{W}) = \text{span}(\mathbf{1}). \tag{6.3}$$

The first condition implies that the weights are symmetric, i.e., $w_{vu} = w_{uv}$. The second condition ensures that the weights of a given node sum up to 1, i.e., $\sum_{u=1}^n w_{vu} = 1$ for all v. Since $\mathbf{W}\mathbf{1} = \mathbf{1}$ we have that $\mathbf{I}-\mathbf{W}$ is rank deficient. The last condition $\text{null}(\mathbf{I}-\mathbf{W}) = \text{span}(\mathbf{1})$ makes the rank of $\mathbf{I} - \mathbf{W}$ exactly equal to $V - 1$ [18].

The matrix $\mathbf{W}$ can be used to reformulate (6.2) as we show in the following proposition.

Proposition 7 *Define the matrix $\mathbf{Z} := \mathbf{W} \otimes \mathbf{I}_p \in \mathbb{R}^{Vp} \times \mathbb{R}^{Vp}$ as the Kronecker product of the weight matrix $\mathbf{W}$ and the identity matrix $\mathbf{I}_p$, and consider the definitions of the global vector $\mathbf{w} := [\mathbf{w}_1; \ldots; \mathbf{w}_V]$ and aggregate function $f(\mathbf{w}) := \sum_{v=1}^V f_v(\mathbf{w}_v)$. The optimization problem in* (6.2) *is equivalent to*

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathbb{R}^{Vp}}{\operatorname{argmin}} \; f(\mathbf{w}) \qquad \textit{s.t. } (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{w} = \mathbf{0}. \tag{6.4}$$

I.e., $\mathbf{w}^* = [\mathbf{w}_1^*; \ldots; \mathbf{w}_V^*]$ with $\{\mathbf{w}_v^*\}_{v=1}^V$ the solution of (6.2).

Proof: We just show that the constraint $((\mathbf{I}_V - \mathbf{W}) \otimes \mathbf{I}_p)\mathbf{w} = (\mathbf{I}_{Vp} - \mathbf{Z})\mathbf{w} = \mathbf{0}$ is also a consensus constraint. To do so begin by noticing that since $\mathbf{I} - \mathbf{W}$ is positive semidefinite, $\mathbf{I}-\mathbf{Z} = (\mathbf{I}-\mathbf{W})\otimes\mathbf{I}_p$ is also positive semidefinite. Therefore, the null space of the square root matrix $(\mathbf{I}-\mathbf{Z})^{1/2}$ is equal to the null space of $\mathbf{I}-\mathbf{Z}$ and we conclude that satisfying the condition $(\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{w}$ is equivalent to the consensus condition $\mathbf{w}_1 = \cdots = \mathbf{w}_V$. This observation in conjunction with the definition of the aggregate function $f(\mathbf{w}) = \sum_{v=1}^V f_v(\mathbf{w}_v)$ shows

that the programs in (6.4) and (6.3) are equivalent. In particular, the optimal solution of (6.4) is $\mathbf{w}^* = [\mathbf{w}_1^*; \ldots; \mathbf{w}_V^*]$ with $\{\mathbf{w}_v^*\}_{v=1}^V$ the solution of (6.2). ■

The formulation in (6.4) is used to define the proximal method of multipliers (PMM) that we consider in this chapter. To do so introduce dual variables $\mathbf{s} \in \mathbb{R}^{Vp}$ to define the augmented Lagrangian $\mathcal{L}(\mathbf{w}, \mathbf{s})$ of (6.4) as

$$\mathcal{L}(\mathbf{w}, \mathbf{s}) = f(\mathbf{w}) + \mathbf{s}^T(\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{w} + \frac{\alpha}{2}\mathbf{w}^T(\mathbf{I} - \mathbf{Z})\mathbf{w}\,, \tag{6.5}$$

where α is a positive constant. Given the properties of the matrix $\mathbf{Z}$, the augmentation term $(\alpha/2)\mathbf{w}^T(\mathbf{I}-\mathbf{Z})\mathbf{w}$ is null when the variable $\mathbf{w}$ is a feasible solution of (6.4). Otherwise, the inner product is positive and behaves as a penalty for the violation of the consensus constraint.

Introduce a time index $t \in \mathbb{N}$ and define $\mathbf{w}_t$ and $\mathbf{s}_t$ as primal and dual iterates at step t. The primal variable $\mathbf{w}_{t+1}$ is updated by minimizing the sum of the augmented Lagrangian in (6.5) and the proximal term $(\epsilon/2)\|\mathbf{w} - \mathbf{w}_t\|^2$. We then have that

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{Vp}} \left\{ \mathcal{L}(\mathbf{w}, \mathbf{s}_t) + \frac{\epsilon}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 \right\}, \tag{6.6}$$

where the proximal coefficient $\epsilon > 0$ is a strictly positive constant. The dual variable $\mathbf{s}_t$ is updated by ascending through the gradient of the augmented Lagrangian with respect to the dual variable $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{w}_{t+1}, \mathbf{s}_t)$ with stepsize α

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \alpha(\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{w}_{t+1}. \tag{6.7}$$

The updates in (6.6) and (6.7) for PMM can be considered as a generalization of the method of multipliers (MM), because setting the proximal coefficient $\epsilon = 0$ recovers the updates of MM. The proximal term $(\epsilon/2)\|\mathbf{w} - \mathbf{w}_t\|^2$ is added to keep the updated variable $\mathbf{w}_{t+1}$ close to the previous iterate $\mathbf{w}_t$. This does not affect convergence guarantees but improves computational stability.

The primal update in (6.6) may be computationally costly – because it requires solving a convex program – and cannot be implemented in a decentralized manner – because the augmentation term $(1/2\alpha)\mathbf{w}^T(\mathbf{I}-\mathbf{Z})\mathbf{w}$ in (6.5) is not separable. In the following section, we propose an approximation of PMM that makes the minimization in (6.6) computationally economic and separable over nodes of the network. This leads to the set of decentralized updates that define the ESOM algorithm.

6.3 ESOM: Exact second-order method

To reduce the computational complexity of (6.6) and obtain a separable update we introduce a second order approximation of the augmented Lagrangian in (6.5). Consider then the second order Taylor's expansion $\mathcal{L}(\mathbf{w}, \mathbf{s}_t) \approx \mathcal{L}(\mathbf{w}_t, \mathbf{s}_t) + \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t, \mathbf{s}_t)^T (\mathbf{w} - \mathbf{w}_t) + (1/2)(\mathbf{w} - \mathbf{w}_t)^T \nabla^2_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t, \mathbf{s}_t)(\mathbf{w} - \mathbf{w}_t)$ of the augmented Lagrangian with respect to $\mathbf{w}$ centered around $(\mathbf{w}_t, \mathbf{s}_t)$. Using this approximation in lieu of $\mathcal{L}(\mathbf{w}, \mathbf{s}_t)$ in (6.6) leads to the primal update

$$\mathbf{w}_{t+1} = \underset{\mathbf{w} \in \mathbb{R}^{Vp}}{\operatorname{argmin}} \Big\{ \mathcal{L}(\mathbf{w}_t, \mathbf{s}_t) + \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t, \mathbf{s}_t)^T (\mathbf{w} - \mathbf{w}_t) + \frac{1}{2} (\mathbf{w} - \mathbf{w}_t)^T \big(\nabla^2_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t, \mathbf{s}_t) + \epsilon \mathbf{I} \big) (\mathbf{w} - \mathbf{w}_t) \Big\}. \tag{6.8}$$

The minimization in the right hand side of (6.8) is of a positive definite quadratic form. Thus, upon defining the Hessian matrix $\mathbf{H}_t \in \mathbb{R}^{np \times np}$ as

$$\mathbf{H}_t := \nabla^2 f(\mathbf{w}_t) + \alpha(\mathbf{I} - \mathbf{Z}) + \epsilon \mathbf{I}, \tag{6.9}$$

and considering the explicit form of the augmented Lagrangian gradient $\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t, \mathbf{s}_t)$ [cf. (6.5)] it follows that the variable $\mathbf{w}_{t+1}$ in (6.8) is given by

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{H}_t^{-1} \Big[\nabla f(\mathbf{w}_t) + (\mathbf{I} - \mathbf{Z})^{1/2} \mathbf{s}_t + \alpha(\mathbf{I} - \mathbf{Z}) \mathbf{w}_t \Big]. \tag{6.10}$$

A fundamental observation here is that the matrix $\mathbf{H}_t$, which is the Hessian of the objective function in (6.8), is block neighbor sparse. By block neighbor sparse we mean that the (v, u)th block is non-zero if and only if $u \in \mathcal{N}_v$ or $u = i$. To confirm this claim, observe that $\nabla^2 f(\mathbf{w}_t) \in \mathbb{R}^{np \times np}$ is a block diagonal matrix where its vth diagonal block is the Hessian of the vth local function, $\nabla^2 f_v(\mathbf{w}_{v,t}) \in \mathbb{R}^{p \times p}$. Additionally, matrix $\epsilon \mathbf{I}_{Vp}$ is a diagonal matrix which implies that the term $\nabla^2 f(\mathbf{w}_t) + \epsilon \mathbf{I}_{Vp}$ is a block diagonal matrix with blocks $\nabla^2 f_v(\mathbf{w}_{v,t}) + \epsilon \mathbf{I}_p$. Further, it follows from the definition of the matrix $\mathbf{Z}$ that the matrix $\mathbf{I} - \mathbf{Z}$ is neighbor sparse. Therefore, the Hessian $\mathbf{H}_t$ is also neighbor sparse. Although the Hessian $\mathbf{H}_t$ is neighbor sparse, its inverse $\mathbf{H}_t^{-1}$ is not. This observation leads to the conclusion that the update in (6.10) is not implementable in a decentralized manner, i.e., nodes cannot implement (6.10) by exchanging information only with their neighbors.

To resolve this issue, we use a Hessian inverse approximation that is built on truncating the Taylor's series of the Hessian inverse $\mathbf{H}_t^{-1}$ as in [64,128]. To do so, we try to decompose the Hessian as $\mathbf{H}_t = \mathbf{D}_t - \mathbf{B}$ where $\mathbf{D}_t$ is a block diagonal positive definite matrix and $\mathbf{B}$ is a neighbor sparse positive semidefinite matrix. In particular, define $\mathbf{D}_t$ as

$$\mathbf{D}_t := \nabla^2 f(\mathbf{w}_t) + \epsilon \mathbf{I} + 2\alpha(\mathbf{I} - \mathbf{Z}_d), \tag{6.11}$$

where $\mathbf{Z}_d := \text{diag}(\mathbf{Z})$. Observing the definitions of the matrices $\mathbf{H}_t$ and $\mathbf{D}_t$ and considering the relation $\mathbf{B} = \mathbf{D}_t - \mathbf{H}_t$ we conclude that $\mathbf{B}$ is given by

$$\mathbf{B} := \alpha \left(\mathbf{I} - 2\mathbf{Z}_d + \mathbf{Z} \right). \tag{6.12}$$

Notice that using the decomposition $\mathbf{H}_t = \mathbf{D}_t - \mathbf{B}$ and by factoring $\mathbf{D}_t^{1/2}$, the Hessian inverse can be written as $\mathbf{H}_t^{-1} = \mathbf{D}_t^{-1/2}(\mathbf{I} - \mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2})^{-1}\mathbf{D}_t^{-1/2}$. Observe that the inverse matrix $(\mathbf{I} - \mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2})^{-1}$ can be substituted by its Taylor's series $\sum_{u=0}^{\infty}(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2})^u$. Note that this is true if the eigenvalues of the matrix $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ are smaller than 1. In the following sections, we prove that this condition is satisfied. However, computation of the series requires global communication which is not affordable in decentralized settings. Thus, we approximate the Hessian inverse $\mathbf{H}_t^{-1}$ by truncating the first $K+1$ terms of its Taylor's series which leads to the Hessian inverse approximation $\tilde{\mathbf{H}}_t^{-1}(K)$,

$$\tilde{\mathbf{H}}_t^{-1}(K) := \mathbf{D}_t^{-1/2} \sum_{u=0}^{K} \left(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2} \right)^u \mathbf{D}_t^{-1/2}. \tag{6.13}$$

Notice that the approximate Hessian inverse $\tilde{\mathbf{H}}_t^{-1}(K)$ is K-hop block neighbor sparse, i.e., the (v,u)th block is nonzero if and only if there is at least one path between nodes v and u with length K or smaller.

We introduce the Exact Second-Order Method (ESOM) as a second order method for solving decentralized optimization problems which substitutes the Hessian inverse in update (6.10) by its K block neighbor sparse approximation $\hat{\mathbf{H}}_k^{-1}(K)$ defined in (6.13). Therefore, the primal update of ESOM is

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \tilde{\mathbf{H}}_t^{-1}(K)\Big[\nabla f(\mathbf{w}_t) + (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t + \alpha(\mathbf{I} - \mathbf{Z})\mathbf{w}_t\Big]. \tag{6.14}$$

The ESOM dual update is identical to the update in (6.7),

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \alpha(\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{w}_{t+1}. \tag{6.15}$$

Notice that ESOM is different from PMM in approximating the augmented Lagrangian in the primal update of PMM by a second order approximation. Further, ESOM approximates the Hessian inverse of the augmented Lagrangian by truncating the Taylor's series of the Hessian inverse which is not necessarily neighbor sparse. In the following subsection we study the implantation details of the updates in (6.14) and (6.15).

Remark 8 The Hessian decomposition $\mathbf{H}_t = \mathbf{D}_t - \mathbf{B}$ with the matrices $\mathbf{D}_t$ and $\mathbf{B}$ in (6.11) and (6.12), respectively, is not the only valid decomposition. All decompositions of the

form $\mathbf{H}_t = \mathbf{D}_t \pm \mathbf{B}_t$ are valid if $\mathbf{D}_t$ is positive definite and the eigenvalues of the matrix $\mathbf{D}_t^{-1/2}\mathbf{B}_t\mathbf{D}_t^{-1/2}$ are in the interval $(-1, 1)$. The suggested framework guarantees that the matrix $\mathbf{B}$ is positive semidefinite which is helpful in the analysis of the proposed ESOM method. A more comprehensive study of alternative decompositions is studied in [3].

6.3.1 Decentralized implementation of ESOM

The updates in (6.14) and (6.15) show that ESOM is a second order approximation of PMM. Although these updates are necessary for understanding the rationale behind ESOM, they are not implementable in a decentralized fashion since the matrix $(\mathbf{I}-\mathbf{Z})^{1/2}$ is not neighbor sparse. To resolve this issue, define the sequence of variables $\mathbf{q}_t$ as $\mathbf{q}_t := (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t$. Considering the definition of $\mathbf{q}_t$, the primal update in (6.14) can be written as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \tilde{\mathbf{H}}_t^{-1}(K)\big(\nabla f(\mathbf{w}_t) + \mathbf{q}_t + \alpha(\mathbf{I} - \mathbf{Z})\mathbf{w}_t\big). \tag{6.16}$$

Multiplying the dual update in (6.15) by $(\mathbf{I} - \mathbf{Z})^{1/2}$ from the left hand side and using the definition $\mathbf{q}_t := (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t$ yields

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \alpha(\mathbf{I} - \mathbf{Z})\mathbf{w}_{t+1}. \tag{6.17}$$

Notice that the system of updates in (6.16) and (6.17) is equivalent to the updates in (6.14) and (6.15), i.e., the sequences of variables $\mathbf{w}_t$ generated by them are identical. Nodes can implement the primal-dual updates in (6.16) and (6.17) in a decentralized manner, since the squared root matrix $(\mathbf{I} - \mathbf{Z})^{1/2}$ is eliminated from the updates and nodes can compute the products $(\mathbf{I} - \mathbf{Z})\mathbf{w}_t$ and $(\mathbf{I} - \mathbf{Z})\mathbf{w}_{t+1}$ by exchanging information with their neighbors.

To characterize the local update of each node for implementing the updates in (6.16) and (6.17), define

$$\mathbf{g}_t := \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}_t, \mathbf{s}_t) = \nabla f(\mathbf{w}_t) + \mathbf{q}_t + \alpha(\mathbf{I} - \mathbf{Z})\mathbf{w}_t, \tag{6.18}$$

as the gradient of the augmented Lagrangian in (6.5). Further, define the primal descent direction $\mathbf{d}_t(K)$ with K levels of approximation as

$$\mathbf{d}_t(K) := -\tilde{\mathbf{H}}_t^{-1}(K)\,\mathbf{g}_t, \tag{6.19}$$

which implies that the update in (6.16) can be written as $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{d}_t(K)$. According to the definitions of the Hessian inverse approximation in (6.13), the explicit expression for the descent direction $\mathbf{d}_t(K)$ is given by $\mathbf{d}_t(K) = \mathbf{D}_t^{-1/2}\sum_{u=0}^{K}\Big(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\Big)^u\mathbf{D}_t^{-1/2}\mathbf{g}_t$. Considering this definition, we can simplify the expression for the descent direction $\mathbf{d}_t(k+1)$

as

$$\mathbf{d}_t(k+1) = -\mathbf{D}_t^{-1/2} \sum_{u=1}^{k+1} \left(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\right)^u \mathbf{D}_t^{-1/2}\mathbf{g}_t - \mathbf{D}_t^{-1}\mathbf{g}_t, \tag{6.20}$$

where we have separated the first term of the sum from the rest. Factorize $\mathbf{D}_t^{-1}\mathbf{B}$ from the summands in (6.20) to obtain

$$\mathbf{d}_t(k+1) = -\mathbf{D}_t^{-1}\mathbf{B}\mathbf{D}_t^{-1/2} \sum_{u=0}^{k} \left(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\right)^u \mathbf{D}_t^{-1/2}\mathbf{g}_t - \mathbf{D}_t^{-1}\mathbf{g}_t. \tag{6.21}$$

Based on the definition of the descent direction $\mathbf{d}_t(k)$, we obtain that the first term in the right hand side of (6.21) can be simplified as $\mathbf{D}_t^{-1}\mathbf{B}\mathbf{d}_t(k)$. Therefore, the descent directions $\mathbf{d}_t(k)$ and $\mathbf{d}_t(k+1)$ satisfy the condition

$$\mathbf{d}_t(k+1) = \mathbf{D}_t^{-1}\mathbf{B}\mathbf{d}_t(k) - \mathbf{D}_t^{-1}\mathbf{g}_t. \tag{6.22}$$

Define $\mathbf{d}_{v,t}(k)$ as the descent direction of node v at step t which is the vth element of the global descent direction $\mathbf{d}_t(k) = [\mathbf{d}_{1,t}(k); \ldots; \mathbf{d}_{V,t}(k)]$. Therefore, the localized version of the relation in (6.22) at node v is given by

$$\mathbf{d}_{v,t}(k+1) = \mathbf{D}_{vv,t}^{-1} \sum_{u=v, u\in\mathcal{N}_v} \mathbf{B}_{vu}\mathbf{d}_{u,t}(k) - \mathbf{D}_{vv,t}^{-1}\mathbf{g}_{v,t}. \tag{6.23}$$

The update in (6.23) shows that node v can compute its $(k+1)$th descent direction $\mathbf{d}_{v,t}(k+1)$ if it has access to the kth descent direction $\mathbf{d}_{v,t}(k)$ of itself and its neighbors $\mathbf{d}_{u,t}(k)$ for $u \in \mathcal{N}_v$. Thus, if nodes initialize with the ESOM-0 descent direction $\mathbf{d}_{v,t}(0) = -\mathbf{D}_{vv,t}^{-1}\mathbf{g}_{v,t}$ and exchange their descent directions with their neighbors for K rounds and use the update in (6.23), they can compute their local ESOM-K descent direction $\mathbf{d}_{v,t}(K)$. Notice that the vth diagonal block $\mathbf{D}_t$ is given by $\mathbf{D}_{vv,t} := \nabla^2 f_v(\mathbf{w}_{v,t}) + (2\alpha(1-w_{vv}) + \epsilon)\mathbf{I}$, where $\mathbf{w}_{v,t}$ is the primal variable of node v at step t. Thus, the block $\mathbf{D}_{vv,t}$ is locally available at node v. Moreover, node v can evaluate the blocks $\mathbf{B}_{vv} = \alpha(1-w_{vv})\mathbf{I}$ and $\mathbf{B}_{vu} = \alpha w_{vu}\mathbf{I}$ without extra communication. In addition, nodes can compute the gradient $\mathbf{g}_t$ by communicating with their neighbors. To confirm this claim observe that the vth element of $\mathbf{g}_t = [\mathbf{g}_{1,t}; \ldots; \mathbf{g}_{V,t}]$ associated with node v is given by

$$\mathbf{g}_{v,t} := \nabla f_v(\mathbf{w}_{v,t}) + \mathbf{q}_{v,t} + \alpha(1-w_{vv})\mathbf{w}_{v,t} - \alpha \sum_{u\in\mathcal{N}_v} w_{vu}\mathbf{w}_{u,t}, \tag{6.24}$$

where $\mathbf{q}_{v,t} \in \mathbb{R}^p$ is the vth element of $\mathbf{q}_t = [\mathbf{q}_{1,t}; \ldots; \mathbf{q}_{V,t}]$ and $\mathbf{w}_{v,t}$ the primal variable of node v at step t and they are both available at node v. Hence, the update in (6.16) can be

Algorithm 8 ESOM-K method at node v

Require: Initial iterates $\mathbf{w}_{v,0} = \mathbf{w}_{u,0} = \mathbf{0}$ for $u \in \mathcal{N}_v$ and $\mathbf{q}_{v,0} = \mathbf{0}$.

1: $\mathbf{B}$ blocks: $\mathbf{B}_{vv} = \alpha(1 - w_{vv})\mathbf{I}$ and $\mathbf{B}_{vu} = \alpha w_{vu}\mathbf{I}$

2: **for** $t = 0, 1, 2, \ldots$ **do**

3: Update $\mathbf{D}$ block: $\mathbf{D}_{vv,t} = \nabla^2 f_v(\mathbf{w}_{v,t}) + (2\alpha(1 - w_{vv}) + \epsilon)\mathbf{I}$

4: Compute gradient $\mathbf{g}_{v,t} = \nabla f_v(\mathbf{w}_{v,t}) + \mathbf{q}_{v,t} + \alpha(1 - w_{vv})\mathbf{w}_{v,t} - \alpha \sum_{u \in \mathcal{N}_i} w_{vu}\mathbf{w}_{u,t}$

5: Compute ESOM-0 descent direction $\mathbf{d}_{v,t}(0) = -\mathbf{D}_{vv,t}^{-1}\mathbf{g}_{v,t}$

6: **for** $k = 0, \ldots, K-1$ **do**

7: Exchange $\mathbf{d}_{v,t}(k)$ with neighbors $u \in \mathcal{N}_v$

8: Compute $\mathbf{d}_{v,t}(k+1) = \mathbf{D}_{vv,t}^{-1}\Big[\sum_{u \in \mathcal{N}_v, u=v} \mathbf{B}_{vu}\mathbf{d}_{u,t}(k) - \mathbf{g}_{v,t}\Big]$

9: **end for**

10: Update primal iterate: $\mathbf{w}_{v,t+1} = \mathbf{w}_{v,t} + \mathbf{d}_{v,t}(K)$.

11: Exchange iterates $\mathbf{w}_{v,t+1}$ with neighbors $u \in \mathcal{N}_i$.

12: Update dual iterate: $\mathbf{q}_{v,t+1} = \mathbf{q}_{v,t} + \alpha(1 - w_{vv})\mathbf{w}_{v,t+1} - \alpha \sum_{u \in \mathcal{N}_v} w_{vu}\mathbf{w}_{u,t+1}$.

13: **end for**

implemented in a decentralized manner. Likewise, nodes can implement the dual update in (6.17) using the local update

$$\mathbf{q}_{v,t+1} = \mathbf{q}_{v,t} + \alpha(1 - w_{vv})\mathbf{w}_{v,t+1} - \alpha \sum_{u \in \mathcal{N}_v} w_{vu}\mathbf{w}_{u,t+1}, \tag{6.25}$$

which requires access to the local primal variable $\mathbf{w}_{u,t+1}$ of the neighboring nodes $u \in \mathcal{N}_v$.

The steps of ESOM-K are summarized in Algorithm 8. The core steps are Steps 5-9 which correspond to computing the ESOM-K primal descent direction $\mathbf{d}_{v,t}(K)$. In Step 5, Each node computes its initial descent direction $\mathbf{d}_{v,t}(0)$ using the block $\mathbf{D}_{vv,t}$ and the local gradient $\mathbf{g}_{v,t}$ computed in Steps 3 and 4, respectively. Steps 7 and 8 correspond to the recursion in (6.23). In step 7, nodes exchange their kth level descent direction $\mathbf{d}_{v,t}(k)$ with their neighboring nodes to compute the $(k+1)$th descent direction $\mathbf{d}_{v,t}(k+1)$ in Step 8. The outcome of this recursion is the Kth level descent direction $\mathbf{d}_{v,t}(K)$ which is required for the update of the primal variable $\mathbf{w}_{v,t}$ in Step 10. Notice that the blocks of the neighbor sparse matrix $\mathbf{B}$, which are required for Step 8, are computed and stored in Step 1. After updating the primal variables in Step 10, nodes exchange their updated variables $\mathbf{w}_{v,t+1}$ with their neighbors $u \in \mathcal{N}_v$ in Step 11. By having access to the decision variable of neighboring nodes, nodes update their local dual variable $\mathbf{q}_{v,t}$ in Step 12.

Remark 9 The proposed ESOM algorithm solves problem (6.4) in the dual domain by defining the proximal augmented Lagrangian. It is also possible to solve problem (6.4) in the primal domain by solving a penalty version of (6.4). In particular, by using the

quadratic penalty function $(1/2)\|.\|^2$ for the constraint $(\mathbf{I}-\mathbf{Z})^{1/2}\mathbf{w}$ with penalty coefficient α, we obtain the penalized version of (6.4)

$$\hat{\mathbf{w}}^* \ := \ \underset{\mathbf{w}\in\mathbb{R}^{Vp}}{\operatorname{argmin}} f(\mathbf{w}) + \frac{\alpha}{2}\mathbf{w}^T(\mathbf{I}-\mathbf{Z})\mathbf{w}, \tag{6.26}$$

where $\hat{\mathbf{w}}^*$ is the optimal argument of the penalized objective function. Notice that $\hat{\mathbf{w}}^*$ is not equal to the optimal argument $\mathbf{w}^*$ and the distance $\|\mathbf{w}^*-\hat{\mathbf{w}}^*\|$ depends on the choice of α. The objective function in (6.26) can be minimized by descending through the gradient descent direction which leads to the update of decentralized gradient descent (DGD) [80]. The convergence of DGD can be improved by using Newton's method. Notice that the Hessian of the objective function in (6.26) is given by

$$\hat{\mathbf{H}} := \nabla^2 f(\mathbf{w}) + \alpha(\mathbf{I}-\mathbf{Z}). \tag{6.27}$$

The Hessian $\hat{\mathbf{H}}$ in (6.27) is identical to the Hessian $\mathbf{H}$ in (6.9) except for the term $\epsilon\mathbf{I}$. Therefore, the same technique for approximating the Hessian inverse $\hat{\mathbf{H}}^{-1}$ can be used to approximate the Newton direction of the penalized objective function in (6.26) which leads to the update of the Network Newton (NN) methods [64, 65]. Thus, ESOM and NN use an approximate decentralized variation of Newton's method for solving two different problems. In other words, ESOM uses the approximate Newton direction for minimizing the augmented Lagrangian of (6.4), while NN solves a penalized version of (6.4) using this approximation. This difference justifies the reason that the sequence of iterates generated by ESOM converges to the optimal argument $\mathbf{w}^*$ (Section 6.4), while NN converges to a neighborhood of $\mathbf{w}^*$.

Remark 10 ESOM approximates the augmented Lagrangian $\mathcal{L}(\mathbf{w},\mathbf{s})$ in (6.6) by its second order approximation. If we substitute the augmented Lagrangian by its first order approximation we can recover the update of EXTRA proposed in [111]. To be more precise, we can substitute $\mathcal{L}(\mathbf{w},\mathbf{s}_t)$ in (6.6) by its first order approximation $\mathcal{L}(\mathbf{w}_t,\mathbf{s}_t)+\nabla\mathcal{L}(\mathbf{w}_t,\mathbf{s}_t)^T(\mathbf{w}-\mathbf{w}_t)$ near the point $(\mathbf{w}_t,\mathbf{s}_t)$ to update the primal variable $\mathbf{w}$. Considering this substitution, the update of $\mathbf{w}_{t+1}$ is given by

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^{Vp}}{\operatorname{argmin}} \left\{ \mathcal{L}(\mathbf{w}_t,\mathbf{s}_t) + \nabla\mathcal{L}(\mathbf{w}_t,\mathbf{s}_t)^T(\mathbf{w}-\mathbf{w}_t) + \frac{\epsilon}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \right\}. \tag{6.28}$$

Thus, considering the definition of the augmented Lagrangian in (6.5) the updated variable $\mathbf{w}_{t+1}$ can be explicitly written as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{\epsilon}\left[\nabla f(\mathbf{w}_t) + (\mathbf{I}-\mathbf{Z})^{1/2}\mathbf{s}_t + \alpha(\mathbf{I}-\mathbf{Z})\mathbf{w}_t\right]. \tag{6.29}$$

By subtracting the update at step $t-1$ from the update at step t and using the dual variables relation that $\mathbf{s}_{t+1} = \mathbf{s}_t + \alpha(\mathbf{I}-\mathbf{Z})^{1/2}\mathbf{w}_{t+1}$ we obtain the update

$$\mathbf{w}_{t+1} = \left(2\mathbf{I} - \frac{2\alpha}{\epsilon}(\mathbf{I}-\mathbf{Z})\right)\mathbf{w}_t - \left(\mathbf{I} - \frac{\alpha}{\epsilon}(\mathbf{I}-\mathbf{Z})\right)\mathbf{w}_{t-1} - \frac{1}{\epsilon}(\nabla f(\mathbf{w}_t) - \nabla f(\mathbf{w}_{t-1})). \quad (6.30)$$

The update in (6.30) shows a first-order approximation of the PMM. It is not hard to show that for specific choices of α and ϵ, the update in (6.30) is equivalent to the update of EXTRA in [111]. Thus, we expect to observe faster convergence for ESOM relative to EXTRA as it incorporates second-order information. This advantage is studied in Section 6.5.

6.4 Convergence analysis

In this section, we study convergence rates of PMM and ESOM. First, we show that the sequence of iterates $\mathbf{w}_t$ generated by PMM converges linearly to the optimal argument $\mathbf{w}^*$. Although, PMM cannot be implemented in a decentralized fashion, its convergence rate can be used as a benchmark for evaluating the performance of ESOM. We then follow the section by analyzing convergence properties ESOM. We show that ESOM exhibits a linear convergence rate and compare its factor of linear convergence with the linear convergence factor of PMM. In proving these results we consider the following assumptions.

Assumption 12 *The local objective functions $f_v(\mathbf{w})$ are twice differentiable and the eigenvalues of the local objective functions Hessian $\nabla^2 f(\mathbf{w})$ are bounded by positive constants $0 < m \leq M < \infty$, i.e.*

$$m\mathbf{I} \preceq \nabla^2 f_v(\mathbf{w}_v) \preceq M\mathbf{I}, \quad (6.31)$$

for all $\mathbf{w}_v \in \mathbb{R}^p$ and $v = 1, \ldots, V$.

The lower bound in (6.31) implies that the local objective functions f_v are strongly convex with constant $m > 0$. The upper bound for the eigenvalues of the Hessians $\nabla^2 f_v$ implies that the gradients of the local objective functions ∇f_v are Lipschitz continuous with constant M. Notice that the global objective function $\nabla^2 f(\mathbf{w})$ is a block diagonal matrix where its vth diagonal block is $\nabla^2 f_v(\mathbf{w}_v)$. Therefore, the bounds on the eigenvalues of the local Hessians $\nabla^2 f_v(\mathbf{w}_v)$ in (6.31) also hold for the global objective function Hessian $\nabla^2 f(\mathbf{w})$. I.e.,

$$m\mathbf{I} \preceq \nabla^2 f(\mathbf{w}) \preceq M\mathbf{I}, \quad (6.32)$$

for all $\mathbf{w} \in \mathbb{R}^{Vp}$. Thus, the global objective function f is also strongly convex with constant m and its gradients ∇f are Lipschitz continuous with constant M.

6.4.1 Convergence of proximal method of multipliers

Convergence rate of PMM can be considered as a benchmark for the convergence rate of ESOM. To establish linear convergence of PMM, We first study the relationship between the primal $\mathbf{w}$ and dual $\mathbf{s}$ iterates generated by PMM and the optimal arguments $\mathbf{w}^*$ and $\mathbf{s}^*$ in the following lemma.

Lemma 18 *Consider the updates for the proximal method of multipliers in* (6.6) *and* (6.7)*. The sequences of primal and dual iterates generated by PMM satisfy*

$$\mathbf{s}_{t+1} - \mathbf{s}_t - \alpha(\mathbf{I} - \mathbf{Z})^{1/2}(\mathbf{w}_{t+1} - \mathbf{w}^*) = \mathbf{0}, \tag{6.33}$$

and

$$\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*) + (\mathbf{I} - \mathbf{Z})^{1/2}(\mathbf{s}_{t+1} - \mathbf{s}^*) + \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t) = \mathbf{0}. \tag{6.34}$$

Proof: Consider the updates of PMM in (6.6) and (6.7). According to (6.4), the optimal argument $\mathbf{w}^*$ satisfies the condition $(\mathbf{I}-\mathbf{Z})^{1/2}\mathbf{w}^* = \mathbf{0}$. This observation in conjunction with the dual variable update in (6.7) yields the claim in (6.33).

To prove the claim in (6.34), note that the optimality condition of (6.6) implies that $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}_{t+1}, \mathbf{s}_t) + \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t) = \mathbf{0}$. Based on the definition of the Lagrangian $\mathcal{L}(\mathbf{w}, \mathbf{s})$ in (6.5), the optimality condition for the primal update of PMM can be written as

$$\nabla f(\mathbf{w}_{t+1}) + (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t + \alpha(\mathbf{I} - \mathbf{Z})\mathbf{w}_{t+1} + \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t) = \mathbf{0}. \tag{6.35}$$

Further, notice that one of the KKT conditions of the optimization problem in (6.4) is

$$\nabla f(\mathbf{w}^*) + (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}^* = \mathbf{0}. \tag{6.36}$$

Moreover, the optimal solution $\mathbf{w}^* = [\tilde{\mathbf{w}}^*; \ldots; \tilde{\mathbf{w}}^*]$ of (6.4) lies in null$\{\mathbf{I} - \mathbf{Z}\}$. Therefore, we obtain

$$\alpha(\mathbf{I} - \mathbf{Z})\mathbf{w}^* = \mathbf{0}. \tag{6.37}$$

Subtracting the equalities in (6.36) and (6.37) from (6.35) yields

$$\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*) + (\mathbf{I} - \mathbf{Z})^{1/2}(\mathbf{s}_t - \mathbf{s}^*) + \alpha(\mathbf{I} - \mathbf{Z})(\mathbf{w}_{t+1} - \mathbf{w}^*) + \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t) = \mathbf{0}. \tag{6.38}$$

Regrouping the terms in (6.33) implies that $\mathbf{s}_t$ is equivalent to

$$\mathbf{s}_t = \mathbf{s}_{t+1} - \alpha(\mathbf{I} - \mathbf{Z})^{1/2}(\mathbf{w}_{t+1} - \mathbf{w}^*). \tag{6.39}$$

Substituting $\mathbf{s}_t$ in (6.38) by the expression in the right hand side of (6.39) leads to the claim in (6.34). ■

Considering the preliminary results in (6.33) and (6.34), we can state convergence results of PMM. To do so, we prove linear convergence of a Lyapunov function of the primal $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ and dual $\|\mathbf{s}_t - \mathbf{s}^*\|^2$ errors. To be more precise, we define the vector $\mathbf{u} \in \mathbb{R}^{2np}$ and matrix $\mathbf{G} \in \mathbb{R}^{np \times np}$ as

$$\mathbf{u} = \begin{bmatrix} \mathbf{s} \\ \mathbf{w} \end{bmatrix}, \ \mathbf{G} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \alpha\epsilon\mathbf{I} \end{bmatrix}. \tag{6.40}$$

Notice that the sequence $\mathbf{u}_t$ is the concatenation of the dual variable $\mathbf{s}_t$ and primal variable $\mathbf{w}_t$. Likewise, we can define $\mathbf{u}^*$ as the concatenation of the optimal arguments $\mathbf{s}^*$ and $\mathbf{w}^*$. We proceed to prove that the sequence $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}}$ converges linearly to null. Observe that $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}}$ can be simplified as $\|\mathbf{s}_t - \mathbf{s}^*\|^2 + \alpha\epsilon\|\mathbf{w}_t - \mathbf{w}^*\|^2$. This observation shows that $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}}$ is a Lyapunov function of the primal $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ and dual $\|\mathbf{s}_t - \mathbf{s}^*\|^2$ errors. Therefore, linear convergence of the sequence $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}}$ implies linear convergence of the sequence $\|\mathbf{w}_t - \mathbf{w}^*\|^2$. In the following theorem, we show that the sequence $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}}$ converges to zero at a linear rate.

Theorem 9 *Consider the proximal method of multipliers as introduced in* (6.6) *and* (6.7). *Consider* $\beta > 1$ *as an arbitrary constant strictly larger than* 1 *and define* $\hat{\lambda}_{\min}(\mathbf{I} - \mathbf{Z})$ *as the smallest non-zero eigenvalue of the matrix* $\mathbf{I} - \mathbf{Z}$. *Further, recall the definitions of the vector* $\mathbf{u}$ *and matrix* $\mathbf{G}$ *in* (6.40). *If Assumption 12 holds, then the sequence of Lyapunov functions* $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}}$ *generated by PMM satisfies*

$$\|\mathbf{u}_{t+1} - \mathbf{u}^*\|^2_{\mathbf{G}} \ \leq \ \frac{1}{1+\delta} \ \|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}}, \tag{6.41}$$

where the constant δ *is given by*

$$\delta = \min\left\{ \frac{2\alpha\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}{\beta(m+M)}, \frac{2mM}{\epsilon(m+M)}, \frac{(\beta-1)\alpha\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}{\beta\epsilon} \right\}. \tag{6.42}$$

Proof: According to Assumption 12, the global objective function f is strongly convex with constant m and its gradients ∇f are Lipschitz continuous with constant M. Considering these assumptions, we obtain that the inner product $(\mathbf{w}_{t+1} - \mathbf{w}^*)^T(\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*))$

is lower bounded by

$$\frac{mM}{m+M}\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2+\frac{1}{m+M}\|\nabla f(\mathbf{w}_{t+1})-\nabla f(\mathbf{w}^*)\|^2 \\ \leq (\mathbf{w}_{t+1}-\mathbf{w}^*)^T(\nabla f(\mathbf{w}_{t+1})-\nabla f(\mathbf{w}^*)). \tag{6.43}$$

The result in (6.34) shows that the difference $\nabla f(\mathbf{w}_{t+1})-\nabla f(\mathbf{w}^*)$ is equal to $-(\mathbf{I}-\mathbf{Z})^{1/2}(\mathbf{s}_{t+1}-\mathbf{s}^*)-\epsilon(\mathbf{w}_{t+1}-\mathbf{w}_t)$. Apply this substitution into (6.43) and multiply both sides of the resulted inequality by 2 to obtain

$$\frac{2mM}{m+M}\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2+\frac{2}{m+M}\|\nabla f(\mathbf{w}_{t+1})-\nabla f(\mathbf{w}^*)\|^2 \\ \leq -2(\mathbf{w}_{t+1}-\mathbf{w}^*)^T(\mathbf{I}-\mathbf{Z})^{1/2}(\mathbf{s}_{t+1}-\mathbf{s}^*)-2\epsilon(\mathbf{w}_{t+1}-\mathbf{w}^*)^T(\mathbf{w}_{t+1}-\mathbf{w}_t). \tag{6.44}$$

Based on the result in (6.33), we can substitute $(\mathbf{w}_{t+1}-\mathbf{w}^*)^T(\mathbf{I}-\mathbf{Z})^{1/2}$ by $(1/\alpha)(\mathbf{s}_{t+1}-\mathbf{s}_t)^T$. Thus, we can rewrite (6.44) as

$$\frac{2\alpha mM}{m+M}\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2+\frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1})-\nabla f(\mathbf{w}^*)\|^2 \\ \leq -2(\mathbf{s}_{t+1}-\mathbf{s}_t)^T(\mathbf{s}_{t+1}-\mathbf{s}^*)-2\alpha\epsilon(\mathbf{w}_{t+1}-\mathbf{w}^*)^T(\mathbf{w}_{t+1}-\mathbf{w}_t). \tag{6.45}$$

Notice that for any vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ we can write $2(\mathbf{a}-\mathbf{b})^T(\mathbf{a}-\mathbf{c})=\|\mathbf{a}-\mathbf{b}\|^2+\|\mathbf{a}-\mathbf{c}\|^2-\|\mathbf{b}-\mathbf{c}\|^2$. By setting $\mathbf{a}=\mathbf{s}_{t+1}$, $\mathbf{b}=\mathbf{s}_t$, and $\mathbf{c}=\mathbf{s}^*$ we obtain that the inner product $2(\mathbf{s}_{t+1}-\mathbf{s}_t)^T(\mathbf{s}_{t+1}-\mathbf{s}^*)$ in (6.45) can be written as $\|\mathbf{s}_{t+1}-\mathbf{s}_t\|^2+\|\mathbf{s}_{t+1}-\mathbf{s}^*\|^2-\|\mathbf{s}_t-\mathbf{s}^*\|^2$. Likewise, setting $\mathbf{a}=\mathbf{w}_{t+1}$, $\mathbf{b}=\mathbf{w}_t$, and $\mathbf{c}=\mathbf{w}^*$ implies that the inner product $2(\mathbf{w}_{t+1}-\mathbf{w}_t)^T(\mathbf{w}_{t+1}-\mathbf{w}^*)$ in (6.45) is equal to $\|\mathbf{w}_{t+1}-\mathbf{w}_t\|^2+\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2-\|\mathbf{w}_t-\mathbf{w}^*\|^2$. Hence, (6.45) can be simplified as

$$\frac{2\alpha mM}{m+M}\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2+\frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1})-\nabla f(\mathbf{w}^*)\|^2 \\ \leq \alpha\epsilon\|\mathbf{w}_t-\mathbf{w}^*\|^2-\alpha\epsilon\|\mathbf{w}_{t+1}-\mathbf{w}_t\|^2-\alpha\epsilon\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2 \\ +\|\mathbf{s}_t-\mathbf{s}^*\|^2-\|\mathbf{s}_{t+1}-\mathbf{s}_t\|^2-\|\mathbf{s}_{t+1}-\mathbf{s}^*\|^2. \tag{6.46}$$

Now using the definitions of the variable $\mathbf{u}$ and matrix $\mathbf{G}$ in (6.40) we can substitute $\|\mathbf{s}_t-\mathbf{s}^*\|^2-\|\mathbf{s}_{t+1}-\mathbf{s}^*\|^2+\alpha\epsilon\|\mathbf{w}_t-\mathbf{w}^*\|^2-\alpha\epsilon\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2$ by $\|\mathbf{u}_t-\mathbf{u}^*\|^2_{\mathbf{G}}-\|\mathbf{u}_{t+1}-\mathbf{u}^*\|^2_{\mathbf{G}}$. Moreover, the squared norm $\|\mathbf{s}_{t+1}-\mathbf{s}_t\|^2$ is equivalent to $\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2_{\alpha^2(\mathbf{I}-\mathbf{Z})}$ based on the result in (6.33). By applying these substitutions we can rewrite (6.46) as

$$\frac{2\alpha mM}{m+M}\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2+\frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1})-\nabla f(\mathbf{w}^*)\|^2 \\ \leq \|\mathbf{u}_t-\mathbf{u}^*\|^2_{\mathbf{G}}-\|\mathbf{u}_{t+1}-\mathbf{u}^*\|^2_{\mathbf{G}}-\alpha\epsilon\|\mathbf{w}_{t+1}-\mathbf{w}_t\|^2-\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2_{\alpha^2(\mathbf{I}-\mathbf{Z})}. \tag{6.47}$$

Regrouping the terms in (6.47) leads to the following lower bound for the difference $\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2 - \|\mathbf{u}_{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2$,

$$\begin{aligned}
&\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2 - \|\mathbf{u}_{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2 \\
&\geq \frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 + \alpha\epsilon\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 + \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2_{\frac{2\alpha mM}{m+M}\mathbf{I}+\alpha^2(\mathbf{I}-\mathbf{Z})}.
\end{aligned} \tag{6.48}$$

Observe that the result in (6.48) provides a lower bound for the decrement $\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2 - \|\mathbf{u}_{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2$. To prove the claim in (6.41), we need to show that for a positive constant δ we have $\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2 - \|\mathbf{u}_{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2 \geq \delta\|\mathbf{u}_{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2$. Therefore, the inequality in (6.41) is satisfied if we can show that the lower bound in (6.48) is greater than $\delta\|\mathbf{u}_{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2$ or equivalently

$$\begin{aligned}
&\delta\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2 + \delta\alpha\epsilon\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 \\
&\leq \frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 + \alpha\epsilon\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 + \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2_{\frac{2\alpha mM}{m+M}\mathbf{I}+\alpha^2(\mathbf{I}-\mathbf{Z})}.
\end{aligned} \tag{6.49}$$

To prove that the inequality in (6.49) holds for some $\delta > 0$, we first find an upper bound for the squared norm $\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2$ in terms of the summands in the right hand side of (6.49). To do so, consider the relation (6.34) along with the fact that $\mathbf{s}_{t+1}$ and $\mathbf{s}^*$ both lie in the column space of $(\mathbf{I} - \mathbf{Z})^{1/2}$. Note that there always exists a unique $\mathbf{s}^*$ that lies in the column space of $(\mathbf{I} - \mathbf{Z})^{1/2}$ – check Lemma 1 in [54]. Since we know that both $\mathbf{s}_{t+1}$ and $\mathbf{s}^*$ lie in the column space of $(\mathbf{I} - \mathbf{Z})^{1/2}$, there exists a vector $\mathbf{r} \in \mathbb{R}^{Vp}$ such that $\mathbf{s}^* - \mathbf{s}_{t+1} = (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{r}$. This relation implies that $\|(\mathbf{I} - \mathbf{Z})^{1/2}(\mathbf{s}_{t+1} - \mathbf{s}^*)\|^2$ can be written as $\|(\mathbf{I} - \mathbf{Z})\mathbf{r}\|^2 = \mathbf{r}^T(\mathbf{I} - \mathbf{Z})^2\mathbf{r}$. The eigenvalues of the matrix $(\mathbf{I} - \mathbf{Z})^2$ are the squared of eigenvalues of the matrix $(\mathbf{I} - \mathbf{Z})$. Thus, we can write $\mathbf{r}^T(\mathbf{I} - \mathbf{Z})^2\mathbf{r} \geq \hat{\lambda}_{\min}(\mathbf{I} - \mathbf{Z})\mathbf{r}^T(\mathbf{I} - \mathbf{Z})\mathbf{r}$, where $\hat{\lambda}_{\min}(\mathbf{I} - \mathbf{Z})$ is the smallest non-zero eigenvalue of the matrix $\mathbf{I} - \mathbf{Z}$. Observing this inequality and the definition $\mathbf{s}^* - \mathbf{s}_{t+1} = (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{r}$ we can write

$$\left\|(\mathbf{I} - \mathbf{Z})^{1/2}(\mathbf{s}_{t+1} - \mathbf{s}^*)\right\|^2 \geq \hat{\lambda}_{\min}(\mathbf{I} - \mathbf{Z})\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2. \tag{6.50}$$

Moreover, from the inequality in (6.34) we obtain that $\left\|(\mathbf{I} - \mathbf{Z})^{1/2}(\mathbf{s}_{t+1} - \mathbf{s}^*)\right\|^2$ is bounded above by

$$\|(\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_{t+1} - \mathbf{s}^*\|^2 \leq \frac{\beta\epsilon^2}{(\beta - 1)}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 + \beta\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2, \tag{6.51}$$

where $\beta > 1$ is a tunable free parameter. Replacing the norm $\|(\mathbf{I}-\mathbf{Z})^{1/2}\mathbf{s}_{t+1} - \mathbf{s}^*\|^2$ in (6.51) by its lower bound in (6.50) follows that $\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2$ is bounded above by

$$\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2 \leq \frac{\beta\epsilon^2}{(\beta-1)\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 + \frac{\beta}{\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2. \tag{6.52}$$

Considering the result in (6.52) to satisfy the inequality in (6.49), which is a sufficient condition for the claim in (6.41), it remains to show that

$$\begin{aligned}&\frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 + \alpha\epsilon\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 + \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2_{\frac{2\alpha mM}{m+M}\mathbf{I}+\alpha^2(\mathbf{I}-\mathbf{Z})}\\ &\geq \frac{\delta\epsilon^2\beta/(\beta-1)}{\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 + \delta\epsilon\alpha\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 + \frac{\delta\beta}{\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2.\end{aligned} \tag{6.53}$$

To enable (6.53) and consequently enabling (6.49), we only need to verify that there exists $\delta > 0$ such that

$$\frac{2\alpha mM}{m+M}\mathbf{I} + \alpha^2(\mathbf{I}-\mathbf{Z}) \succcurlyeq \delta\alpha\epsilon\mathbf{I}, \quad \frac{2\alpha}{m+M} \geq \frac{\delta\beta}{\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}, \quad \alpha\epsilon \geq \frac{\delta\beta\epsilon^2}{(\beta-1)\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}. \tag{6.54}$$

The conditions in (6.54) are satisfied if the constant δ is chosen as in (6.42). Therefore, for δ in (6.42) the claim in (6.49) holds, which implies the claim in (6.41). ■

The result in Theorem 9 shows linear convergence of the sequence $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}}$ generated by PMM where the factor of linear convergence is $1/(1+\delta)$. Observe that larger δ implies smaller linear convergence factor $1/(1+\delta)$ and faster convergence. Notice that all the terms in the minimization in (6.42) are positive and therefore the constant δ is strictly larger than 0. In addition, the result in Theorem 9 holds for any feasible set of parameters $\beta > 1$, $\epsilon > 0$, and $\alpha > 0$; however, maximizing the parameter δ requires properly choosing the set of parameters β, ϵ, and α.

Observe that when the first positive eigenvalue $\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})$ of the matrix $\mathbf{I}-\mathbf{Z}$, which is the second smallest eigenvalue of $\mathbf{I}-\mathbf{Z}$, is small the constant δ becomes close to zero and convergence becomes slow. Notice that small $\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})$ shows that the graph is not highly connected. This observation matches the intuition that when the graph has less edges the speed of convergence is slower. Additionally, the upper bounds in (6.42) show that when the condition number M/m of the global objective function f is large, δ becomes small and the linear convergence becomes slow.

Although PMM enjoys a fast linear convergence rate, each iteration of PMM requires infinite rounds of communications which make it infeasible. In the following section, we study convergence properties of ESOM as a second order approximation of PMM that is implementable in decentralized settings.

6.4.2 Convergence of ESOM

We proceed to show that the sequence of iterates $\mathbf{w}_t$ generated by ESOM converges linearly to the optimal argument $\mathbf{w}^* = [\tilde{\mathbf{w}}^*; \dots; \tilde{\mathbf{w}}^*]$. To do so, we first prove linear convergence of the Lyapunov function $\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2$ as defined in (6.40). Moreover, we show that by increasing the Hessian inverse approximation accuracy, ESOM factor of linear convergence can be arbitrary close to the linear convergence factor of PMM in Theorem 9.

Notice that ESOM is built on a second order approximation of the proximal augmented Lagrangian used in the update of PMM. To guarantee that the second order approximation suggested in ESOM is feasible, the local objective functions f_v are required to be twice differentiable as assumed in Assumption 12. The twice differentiability of the local objective functions f_v implies that the aggregate function f, which is the sum of a set of twice differentiable functions, is also twice differentiable. This observation shows that the global objective function $\nabla^2 f(\mathbf{w})$ is definable. Considering this observation, we prove some preliminary results for the iterates generated by ESOM in the following lemma.

Lemma 19 *Consider the updates of ESOM in* (6.14) *and* (6.15)*. Recall the definitions of the augmented Lagrangian Hessian* $\mathbf{H}_t$ *in* (6.9) *and the approximate Hessian inverse* $\tilde{\mathbf{H}}_t^{-1}(K)$ *in* (6.13)*. If Assumption 12 holds, then the primal and dual iterates generated by ESOM satisfy*

$$\mathbf{s}_{t+1} - \mathbf{s}_t - \alpha(\mathbf{I} - \mathbf{Z})^{1/2}(\mathbf{w}_{t+1} - \mathbf{w}^*) = \mathbf{0}. \tag{6.55}$$

Moreover, we can show that

$$\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*) + (\mathbf{I} - \mathbf{Z})^{1/2}(\mathbf{s}_{t+1} - \mathbf{s}^*) + \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t) + \mathbf{e}_t = \mathbf{0}, \tag{6.56}$$

where the error vector $\mathbf{e}_t$ *is defined as*

$$\mathbf{e}_t := \nabla f(\mathbf{w}_t) + \nabla^2 f(\mathbf{w}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t) - \nabla f(\mathbf{w}_{t+1}) + \left(\tilde{\mathbf{H}}_t(K) - \mathbf{H}_t\right)(\mathbf{w}_{t+1} - \mathbf{w}_t). \tag{6.57}$$

Proof: Consider the primal update of ESOM in (6.14). By regrouping the terms we obtain

$$\nabla f(\mathbf{w}_t) + (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t + \alpha(\mathbf{I} - \mathbf{Z})\mathbf{w}_t + \tilde{\mathbf{H}}_t(\mathbf{w}_{t+1} - \mathbf{w}_t) = \mathbf{0}, \tag{6.58}$$

where $\tilde{\mathbf{H}}_t$ is the inverse of the Hessian inverse approximation $\tilde{\mathbf{H}}_t^{-1}(K)$. Recall the definition of the exact Hessian $\mathbf{H}_t$ in (6.9). Adding and subtracting the term $\mathbf{H}_t(\mathbf{w}_{t+1} - \mathbf{w}_t)$ to the expression in (6.58) yields

$$\begin{aligned} &\nabla f(\mathbf{w}_t) + \nabla^2 f(\mathbf{w}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t) + (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t \\ &+ \alpha(\mathbf{I} - \tilde{\mathbf{Z}})\mathbf{w}_{t+1} + \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t) + (\tilde{\mathbf{H}}_t - \mathbf{H}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t) = \mathbf{0}. \end{aligned} \tag{6.59}$$

Now using the definition of the error vector $\mathbf{e}_t$ in (6.57) we can rewrite (6.59) as

$$\nabla f(\mathbf{w}_{t+1}) + (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t + \alpha(\mathbf{I} - \tilde{\mathbf{Z}})\mathbf{w}_{t+1} + \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t) + \mathbf{e}_t = \mathbf{0}. \tag{6.60}$$

Notice that the result in (6.60) is identical to the expression for PMM in (6.35) except for the error term $\mathbf{e}_t$. To prove the claim in (6.56) from (6.60), it remains to follow the steps in (6.36)-(6.39). ■

The results in Theorem 19 show the relationships between the primal $\mathbf{w}$ and dual $\mathbf{s}$ iterates generated by ESOM and the optimal arguments $\mathbf{w}^*$ and $\mathbf{s}^*$. The first result in (6.55) is identical to the convergence property of PMM in (6.33), while the second result in (6.56) differs from (6.34) in having the extra summand $\mathbf{e}_t$. The vector $\mathbf{e}_t$ can be interpreted as the error of second order approximation for ESOM at step t. To be more precise, the optimality condition of the primal update of PMM is given by $\nabla f(\mathbf{w}_{t+1}) + (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t + \alpha(\mathbf{I} - \mathbf{Z})\mathbf{w}_{t+1} + \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t) = \mathbf{0}$ as shown in (6.34). Notice that the second order approximation of this condition is equivalent to $\nabla f(\mathbf{w}_t) + \nabla^2 f(\mathbf{w}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t) + (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t + \alpha(\mathbf{I} - \mathbf{Z})\mathbf{w}_{t+1} + \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t) = \mathbf{0}$. However, the exact Hessian inverse $\mathbf{H}_t^{-1} = (\nabla^2 f(\mathbf{w}_t) + \epsilon\mathbf{I} + \alpha(\mathbf{I} - \tilde{\mathbf{Z}}))^{-1}$ cannot be computed in a distributed manner to solve the optimality condition. Thus, it is approximated by the approximate Hessian inverse matrix $\tilde{\mathbf{H}}_t^{-1}(K)$ as introduced in (6.13). This shows that the approximate optimality condition in ESOM is $\nabla f(\mathbf{w}_t) + (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{s}_t + \alpha(\mathbf{I} - \tilde{\mathbf{Z}})\mathbf{w}_t + \tilde{\mathbf{H}}_t(\mathbf{w}_{t+1} - \mathbf{w}_t) = \mathbf{0}$. Hence, the difference between the optimality conditions of PMM and ESOM is $\mathbf{e}_t = \nabla f(\mathbf{w}_t) - \nabla f(\mathbf{w}_{t+1}) + \alpha(\mathbf{I} - \tilde{\mathbf{Z}})(\mathbf{w}_t - \mathbf{w}_{t+1}) + \tilde{\mathbf{H}}_t(\mathbf{w}_{t+1} - \mathbf{w}_t) - \epsilon(\mathbf{w}_{t+1} - \mathbf{w}_t)$. By adding and subtracting the term $\mathbf{H}_t(\mathbf{w}_{t+1} - \mathbf{w}_t)$, the definition of the error vector $\mathbf{e}_t$ in (6.57) follows.

The observation that the vector $\mathbf{e}_t$ characterizes the error of second order approximation in ESOM, motivates analyzing an upper bound for the error vector norm $\|\mathbf{e}_t\|$. To prove that the norm $\|\mathbf{e}_t\|$ is bounded above we assume the following condition is satisfied.

Assumption 13 *The global objective function Hessian $\nabla^2 f(\mathbf{w})$ is Lipschitz continuous with constant L, i.e.,*

$$\|\nabla^2 f(\mathbf{w}) - \nabla^2 f(\tilde{\mathbf{w}})\| \leq L\|\mathbf{w} - \tilde{\mathbf{w}}\|. \tag{6.61}$$

The conditions imposed by Assumption 13 are customary in the analysis of second-order

methods; see, e.g., [77]. In the following lemma, we use the assumption in (6.61) to prove an upper bound for the error norm $\|\mathbf{e}_t\|$ in terms of $\|\mathbf{w}_{t+1} - \mathbf{w}_t\|$.

Lemma 20 *Consider ESOM as introduced in* (6.8)-(6.15) *and recall the definition of the error vector* $\mathbf{e}_t$ *in* (6.57). *Further, define* $c > 0$ *as a lower bound for the local weights* w_{vv}. *If Assumptions 12 and 13 hold, then the error vector norm* $\|\mathbf{e}_t\|$ *is bounded above by*

$$\|\mathbf{e}_t\| \leq \Gamma_t \|\mathbf{w}_{t+1} - \mathbf{w}_t\|, \tag{6.62}$$

where Γ_t *is defined as*

$$\Gamma_t := \min\left\{2M, \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|\right\} + (M + \epsilon + 2\alpha(1-c))\,\rho^{K+1}, \tag{6.63}$$

and $\rho := 2\alpha(1-c)/(2\alpha(1-c) + m + \epsilon)$.

Proof: To prove the result in (6.62), we first use the result in Proposition 2 of [77]. It shows that when the eigenvalues of the Hessian $\nabla^2 f(\mathbf{w})$ are bounded above by M and the Hessian is Lipschitz continuous with constant L we can write

$$\|\nabla f(\mathbf{w}_t) + \nabla^2 f(\mathbf{w}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t) - \nabla f(\mathbf{w}_{t+1})\| \leq \|\mathbf{w}_{t+1} - \mathbf{w}_t\| \min\left\{2M, \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|\right\}. \tag{6.64}$$

Considering the result in (6.64), it remains to find an upper bound for the second term of the error vector $\mathbf{e}_t$ which is $(\tilde{\mathbf{H}}_t(K) - \mathbf{H}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t)$. To do so, we develop first an upper bound for the norm $\|\tilde{\mathbf{H}}_t(K) - \mathbf{H}_t\|$. Notice that by factoring the term $\tilde{\mathbf{H}}_t(K)^{1/2}$ from left and right, and using the Cauchy-Schwarz inequality we obtain that

$$\left\|\tilde{\mathbf{H}}_t(K) - \mathbf{H}_t\right\| \leq \left\|\tilde{\mathbf{H}}_t(K)^{\frac{1}{2}}\right\|^2 \left\|\mathbf{I} - \tilde{\mathbf{H}}_t^{-\frac{1}{2}}(K)\mathbf{H}_t\tilde{\mathbf{H}}_t^{-\frac{1}{2}}(K)\right\|. \tag{6.65}$$

Note that the eigenvalues of the matrices $\mathbf{I} - \mathbf{H}_t\tilde{\mathbf{H}}_t^{-1}(K)$ and $\mathbf{I} - \tilde{\mathbf{H}}_t^{-1/2}(K)\mathbf{H}_t\tilde{\mathbf{H}}_t^{-1/2}(K)$ are the same since these two matrices are *similar*. In linear algebra, two matrices $\mathbf{A}$ and $\tilde{\mathbf{A}}$ are called similar if $\tilde{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$ for an invertible matrix $\mathbf{P}$. Thus, we proceed to find bounds for the eigenvalues of $\mathbf{I} - \mathbf{H}_t\tilde{\mathbf{H}}_t^{-1}(K)$, to bound the norm in (6.65). According to Lemma 3 in [64], we can simplify $\mathbf{I} - \mathbf{H}_t\tilde{\mathbf{H}}_t^{-1}(K)$ as

$$\mathbf{I} - \mathbf{H}_t\tilde{\mathbf{H}}_t^{-1}(K) = (\mathbf{B}\mathbf{D}_t^{-1})^{K+1}. \tag{6.66}$$

Note that the matrices $\mathbf{B}$ and $\mathbf{D}_t$ in this chapter are different from the ones in [64], but the analyses of them are very similar. Following the proof of Proposition 2 in [64], we define

$\hat{\mathbf{D}} := 2\alpha(\mathbf{I} - \mathbf{Z}_d)$. Notice that the matrix $\hat{\mathbf{D}}$ is bock diagonal where its vth diagonal block is $2\alpha(1 - w_{vv})\mathbf{I}_p$. Thus, $\hat{\mathbf{D}}$ is positive definite and invertible. Instead of studying an upper bound for the eigenvalues of $\mathbf{B}\mathbf{D}_t^{-1}$, we try to find an upper bound for the eigenvalues of its similar matrix $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ which is symmetric. We are allowed to write the product $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ as

$$\mathbf{D}_t^{-\frac{1}{2}}\mathbf{B}\mathbf{D}_t^{-\frac{1}{2}} = \left(\mathbf{D}_t^{-\frac{1}{2}}\hat{\mathbf{D}}^{\frac{1}{2}}\right)\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{B}\hat{\mathbf{D}}^{-1/2}\right)\left(\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{D}_t^{-\frac{1}{2}}\right). \tag{6.67}$$

The next step is to find an upper bound for the eigenvalues of $\mathbf{B}\hat{\mathbf{D}}^{-1}$ in (6.67). Based on the definitions of matrices $\mathbf{B}$ and $\hat{\mathbf{D}}$, the product $\mathbf{B}\hat{\mathbf{D}}^{-1}$ is given by

$$\mathbf{B}\hat{\mathbf{D}}^{-1} = (\mathbf{I} - 2\mathbf{Z}_d + \mathbf{Z})\,(2(\mathbf{I} - \mathbf{Z}_d))^{-1}. \tag{6.68}$$

According to the result in Proposition 2 of [64], the eigenvalues of the matrix $(\mathbf{I} - 2\mathbf{Z}_d + \mathbf{Z})(2(\mathbf{I} - \mathbf{Z}_d))^{-1}$ are uniformly bounded by 0 and 1. Thus, we obtain that the eigenvalues of $\hat{\mathbf{D}}^{-1/2}\mathbf{B}\hat{\mathbf{D}}^{-1/2}$ are bounded by 0 and 1 and we can write

$$\|\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{B}\hat{\mathbf{D}}^{-\frac{1}{2}}\| \leq 1. \tag{6.69}$$

According to the definitions of the matrices $\hat{\mathbf{D}}$ and $\mathbf{D}_t$, the product $\hat{\mathbf{D}}^{1/2}\mathbf{D}_t^{-1/2}$ is block diagonal and the vth diagonal block is given by

$$\left[\hat{\mathbf{D}}\mathbf{D}_t^{-1}\right]_{vv} = \left(\frac{\nabla^2 f_v(\mathbf{w}_{v,t}) + \epsilon\mathbf{I}}{2\alpha(1 - w_{vv})} + \mathbf{I}\right)^{-1}. \tag{6.70}$$

Based on Assumption 12, the eigenvalues of the local Hessians $\nabla^2 f_v(\mathbf{w}_v)$ are bounded by m and M. Further, notice that the diagonal elements w_{vv} of the weight matrix $\mathbf{W}$ are bounded below by c. Considering these bounds, we can show that the eigenvalues of the matrices $(1/2\alpha(1 - w_{vv}))(\nabla^2 f_v(\mathbf{w}_{v,t}) + \epsilon\mathbf{I}) + \mathbf{I}$ for all $v = 1, \ldots, V$ are bounded below by

$$\left[\frac{m + \epsilon}{2\alpha(1 - c)} + 1\right]\mathbf{I} \preceq \frac{\nabla^2 f_v(\mathbf{w}_{v,t}) + \epsilon\mathbf{I}}{2\alpha(1 - w_{vv})} + \mathbf{I}. \tag{6.71}$$

By considering the bounds in (6.71), the eigenvalues of each block of the matrix $\hat{\mathbf{D}}\mathbf{D}_t^{-1}$, introduced in (6.70), are bounded above as

$$\left(\frac{\nabla^2 f_v(\mathbf{w}_{v,t}) + \epsilon\mathbf{I}}{2\alpha(1 - w_{vv})} + \mathbf{I}\right)^{-1} \preceq \left[\frac{m + \epsilon}{2\alpha(1 - c)} + 1\right]^{-1}\mathbf{I}. \tag{6.72}$$

The upper bound in (6.72) for the eigenvalues of each diagonal block of the matrix $\hat{\mathbf{D}}\mathbf{D}_t^{-1}$

implies that the matrix norm $\|\hat{\mathbf{D}}\mathbf{D}_t^{-1}\|$ is bounded above by

$$\|\hat{\mathbf{D}}\mathbf{D}_t^{-1}\| \leq \rho := \frac{2\alpha(1-c)}{2\alpha(1-c)+m+\epsilon}. \tag{6.73}$$

Considering the upper bounds in (6.69) and (6.73) and the relation in (6.67) we obtain that

$$\|\mathbf{D}_t^{-\frac{1}{2}}\mathbf{B}\mathbf{D}_t^{-\frac{1}{2}}\| \leq \rho. \tag{6.74}$$

Thus, the eigenvalues of the positive definite symmetric matrix $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ are bounded by ρ. Hence, the eigenvalues of its similar matrix $\mathbf{B}\mathbf{D}_t^{-1}$ are bounded by ρ. This bound along with the result in (6.66) shows that the eigenvalues of the matrix $\mathbf{I} - \mathbf{H}_t\tilde{\mathbf{H}}_t^{-1}(K)$ are uniformly bounded by 0 and ρ^{K+1}. Therefore, the eigenvalues of its similar symmetric matrix $\mathbf{I} - \tilde{\mathbf{H}}_t^{-1/2}(K)\mathbf{H}_t\tilde{\mathbf{H}}_t^{-1/2}(K)$ are between 0 and ρ^K which implies that $\|\mathbf{I} - \tilde{\mathbf{H}}_t^{-1/2}(K)\mathbf{H}_t\tilde{\mathbf{H}}_t^{-1/2}(K)\| \leq \rho^{K+1}$. This result in conjunction with the inequality in (6.65) yields

$$\left\|\tilde{\mathbf{H}}_t(K) - \mathbf{H}_t\right\| \leq \rho^{K+1}\left\|\tilde{\mathbf{H}}_t(K)^{\frac{1}{2}}\right\|^2. \tag{6.75}$$

To bound the norm $\|\tilde{\mathbf{H}}_t(K)\|$, we first find a lower bound for the eigenvalues of the approximate Hessian inverse $\tilde{\mathbf{H}}_t^{-1}(K)$. Notice that according to the definition of the approximate Hessian inverse in (6.13), we can write

$$\tilde{\mathbf{H}}_t^{-1}(K) := \mathbf{D}_t^{-1} + \mathbf{D}_t^{-1}\ \sum_{u=1}^{K}(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2})^u\ \mathbf{D}_t^{-1/2}. \tag{6.76}$$

Notice that according to the result in Proposition 1 of [64], the matrix $(\mathbf{I} - 2\mathbf{Z}_d + \mathbf{Z})$ is positive semidefinite which implies that $\mathbf{B} = \alpha\,(\mathbf{I} - 2\mathbf{Z}_d + \mathbf{Z})$ is also positive semidefinite. Thus, all the K summands in (6.76) are positive semidefinite and as a result we obtain that

$$\mathbf{D}_t^{-1} \preceq\ \tilde{\mathbf{H}}_t^{-1}(K). \tag{6.77}$$

The eigenvalues of $\mathbf{I} - \mathbf{Z}_d$ are bounded above by $1 - c$, since all the local weights w_{vv} are larger than c. This observation in conjunction with the strong convexity of the global objective function f implies that the eigenvalues of $\mathbf{D}_t = \nabla^2 f(\mathbf{w}_t) + \epsilon\mathbf{I} + 2\alpha(\mathbf{I} - \mathbf{Z}_d)$ are bounded above by $M + \epsilon + 2\alpha(1-c)$. Therefore, the eigenvalues of $\mathbf{D}_t^{-1}$ are bounded below as

$$\frac{1}{M+\epsilon+2\alpha(1-c)}\ \mathbf{I}\ \preceq\ \mathbf{D}_t^{-1}. \tag{6.78}$$

The results in (6.77) and (6.78) imply that the eigenvalues of the approximate Hessian inverse $\tilde{\mathbf{H}}_t^{-1}(K)$ are greater than $1/(M + \epsilon + 2\alpha(1-c))$. Therefore, the eigenvalues of the

positive definite matrix $\tilde{\mathbf{H}}_t(K)$ are smaller than $M + \epsilon + 2\alpha(1-c)$ and we can write

$$\left\|\tilde{\mathbf{H}}_t(K)\right\| \leq M + \epsilon + 2\alpha(1-c). \tag{6.79}$$

Considering the inequalities in (6.75) and (6.79) and using the Cauchy-Schwarz inequality we can show that the norm $\|(\tilde{\mathbf{H}}_t(K) - \mathbf{H}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t)\|$ is bounded above by

$$\left\|(\tilde{\mathbf{H}}_t(K) - \mathbf{H}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t)\right\| \leq (M + \epsilon + 2\alpha(1-c))\,\rho^{K+1}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|. \tag{6.80}$$

Observing the inequalities in (6.64) and (6.80) and using the triangle inequality the claim in (6.62) follows. ■

First, note that the lower bound $c > 0$ on the local weights w_{vv} is implied from the fact that all the local weights are positive. In particular, we can define the lower bound c as $c := \min_v w_{vv}$. The result in (6.62) shows that the error of second order approximation in ESOM vanishes as the sequence of iterates $\mathbf{w}_t$ approaches the optimal argument $\mathbf{w}^*$. We will show in Theorem 10 that $\|\mathbf{w}_t - \mathbf{w}^*\|$ converges to zero which implies that the limit of the sequence $\|\mathbf{w}_{t+1} - \mathbf{w}_t\|$ is zero.

To understand the definition of Γ_t in (6.63), we have to decompose the error vector $\mathbf{e}_t$ in (6.57) into two parts. The first part is $\nabla f(\mathbf{w}_t) + \nabla^2 f(\mathbf{w}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t) - \nabla f(\mathbf{w}_{t+1})$ which comes from the fact that ESOM minimizes a second order approximation of the proximal augmented Lagrangian instead of the exact proximal augmented Lagrangian. This term can be bounded by $\min\{2M, (L/2)\|\mathbf{w}_{t+1} - \mathbf{w}_t\|\}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|$ as shown in Lemma 20. The second part of the error vector $\mathbf{e}_t$ is $(\tilde{\mathbf{H}}_t(K) - \mathbf{H}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t)$ which shows the error of Hessian inverse approximation. Notice that computation of the exact Hessian inverse $\mathbf{H}_t^{-1}$ is not possible and ESOM approximates the exact Hessian by the approximation $\tilde{\mathbf{H}}_t^{-1}(K)$. According to the results in [64], the difference $\|\tilde{\mathbf{H}}_t(K) - \mathbf{H}_t\|$ can upper bounded by $(M + \epsilon + 2(1-c)/\alpha)\rho^{K+1}$ which justifies the second term of the expression for Γ_t in (6.63). In the following theorem, we use the result in Lemma 20 to show that the sequence of Lyapunov functions $\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2$ generated by ESOM converges to zero linearly.

Theorem 10 *Consider ESOM as introduced in* (6.8)-(6.15). *Consider $\beta > 1$ and $\phi > 1$ as arbitrary constants that are strictly larger than* 1, *and ζ as a positive constant that is chosen from the interval $\zeta \in ((m+M)/2mM, \epsilon/\Gamma_t^2)$. Further, recall the definitions of the vector $\mathbf{u}$ and matrix $\mathbf{G}$ in* (6.40) *and consider $\hat{\lambda}_{\min}(\mathbf{I} - \mathbf{Z})$ as the smallest non-zero eigenvalue of the matrix $\mathbf{I} - \mathbf{Z}$. If Assumptions 12 and 13 hold, then the sequence of Lyapunov functions $\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2$ generated by ESOM satisfies*

$$\|\mathbf{u}_{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2 \;\leq\; \frac{1}{1+\delta_t'}\;\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2. \tag{6.81}$$

where the sequence δ'_t is given by

$$\delta'_t = \min\left\{ \frac{2\alpha\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}{\phi\beta(m+M)}, \quad \left[\frac{2mM}{\epsilon(m+M)} - \frac{1}{\zeta\epsilon}\right], \right. \tag{6.82}$$
$$\left. \frac{(\beta-1)\alpha\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})}{\beta\epsilon}\left[1-\frac{\zeta\Gamma_t^2}{\epsilon}\right]\left[1+\frac{\phi\Gamma_t^2(\beta-1)}{(\phi-1)\epsilon^2}\right]^{-1}\right\}.$$

Proof: Notice that in proving the claim in (6.81) we use some of the steps in the proof of Theorem 9 to avoid rewriting similar equations. First, note that according to the result in (6.56), the difference $\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)$ for the ESOM method can be written as

$$\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*) = -(\mathbf{I}-\mathbf{Z})^{1/2}(\mathbf{s}_{t+1}-\mathbf{s}^*) - \epsilon(\mathbf{w}_{t+1}-\mathbf{w}_t) - \mathbf{e}_t. \tag{6.83}$$

Now recall the the inequality in (6.43) and substitute the gradients difference $\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)$ in the inner product $(\mathbf{w}_{t+1}-\mathbf{w}^*)^T(\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*))$ by the expression in the right hand side of (6.83). Applying this substitution and multiplying both sides of the implied inequality by 2α follows

$$\begin{aligned}
&\frac{2\alpha mM}{m+M}\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2 + \frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 \\
&\leq -2\alpha\epsilon(\mathbf{w}_{t+1}-\mathbf{w}^*)^T(\mathbf{w}_{t+1}-\mathbf{w}_t) - 2\alpha(\mathbf{w}_{t+1}-\mathbf{w}^*)^T\mathbf{e}_t \\
&\quad - 2\alpha(\mathbf{w}_{t+1}-\mathbf{w}^*)^T(\mathbf{I}-\mathbf{Z})^{1/2}(\mathbf{s}_{t+1}-\mathbf{s}^*).
\end{aligned} \tag{6.84}$$

By following the steps in (6.44)-(6.48), the result in (6.84) leads to a lower bound for $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}} - \|\mathbf{u}_{t+1}-\mathbf{u}^*\|^2_{\mathbf{G}}$ as

$$\begin{aligned}
\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}} - \|\mathbf{u}_{t+1}-\mathbf{u}^*\|^2_{\mathbf{G}} \geq{}& \frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 + \alpha\epsilon\|\mathbf{w}_{t+1}-\mathbf{w}_t\|^2 \\
&+ \|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2_{\frac{2\alpha mM}{m+M}\mathbf{I}+\alpha^2(\mathbf{I}-\mathbf{Z})} + 2\alpha(\mathbf{w}_{t+1}-\mathbf{w}^*)^T\mathbf{e}_t.
\end{aligned} \tag{6.85}$$

Note that the inner product $2(\mathbf{w}_{t+1}-\mathbf{w}^*)^T\mathbf{e}_t$ is bounded below by $-(1/\zeta)\|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2 - \zeta\|\mathbf{e}_t\|^2$ for any positive constant $\zeta > 0$. Thus, the lower bound in (6.85) can be updated as

$$\begin{aligned}
\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}} - \|\mathbf{u}_{t+1}-\mathbf{u}^*\|^2_{\mathbf{G}} \geq{}& \|\mathbf{w}_{t+1}-\mathbf{w}^*\|^2_{(\frac{2\alpha mM}{m+M}-\frac{\alpha}{\zeta})\mathbf{I}+\alpha^2(\mathbf{I}-\mathbf{Z})} + \alpha\epsilon\|\mathbf{w}_{t+1}-\mathbf{w}_t\|^2 \\
&+ \frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 - \alpha\zeta\|\mathbf{e}_t\|^2.
\end{aligned} \tag{6.86}$$

To establish (6.81), we need to show that the difference $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}} - \|\mathbf{u}_{t+1}-\mathbf{u}^*\|^2_{\mathbf{G}}$ is bounded below by $\delta'_t\|\mathbf{u}_{t+1}-\mathbf{u}^*\|^2_{\mathbf{G}}$. To do so, we show that the lower bound for $\|\mathbf{u}_t -$

$\mathbf{u}^*\|_{\mathbf{G}}^2 - \|\mathbf{u}_{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2$ in (6.86) is larger than $\delta_t'\|\mathbf{u}_{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2$, i.e.,

$$\delta_t'\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2 + \delta_t'\alpha\epsilon\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 \leq \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2_{(\frac{2\alpha mM}{m+M} - \frac{\alpha}{\zeta})\mathbf{I} + \alpha^2(\mathbf{I}-\mathbf{Z})} + \alpha\epsilon\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 + \frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 - \alpha\zeta\|\mathbf{e}_t\|^2. \tag{6.87}$$

We proceed to find an upper bound for the squared norm $\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2$ in terms of the summands in the right hand side of (6.87). Consider the relation (6.60) as well as the fact that $\mathbf{s}_{t+1}$ and $\mathbf{s}^*$ both lie in the column space of $(\mathbf{I} - \mathbf{Z})^{1/2}$. It follows that $\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2$ is bounded above by

$$\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2 \leq \frac{\beta\epsilon^2}{(\beta-1)\hat{\lambda}}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 + \frac{\beta\phi}{(\phi-1)\hat{\lambda}}\|\mathbf{e}_t\|^2 + \frac{\phi\beta}{\hat{\lambda}}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2, \tag{6.88}$$

where we have used $\hat{\lambda}$ instead of $\hat{\lambda}_{\min}(\mathbf{I}-\mathbf{Z})$ to simplify notation. By substituting the upper bound in (6.88) for the squared norm $\|\mathbf{s}_{t+1} - \mathbf{s}^*\|^2$ in (6.87) we obtain a sufficient condition for the result in (6.87) which is given by

$$\begin{aligned}
&\delta_t'\alpha\epsilon\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 + \frac{\delta'\beta\epsilon^2}{(\beta-1)\hat{\lambda}}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 + \frac{\delta_t'\phi\beta}{\hat{\lambda}}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 + \frac{\delta_t'\beta\phi\alpha^2\|\mathbf{e}_t\|^2}{(\phi-1)\hat{\lambda}} \\
&\leq \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2_{(\frac{2\alpha mM}{m+M} - \frac{\alpha}{\zeta})\mathbf{I} + \alpha^2(\mathbf{I}-\mathbf{Z})} + \alpha\epsilon\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\
&\quad + \frac{2\alpha}{m+M}\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 - \alpha\zeta\|\mathbf{e}_t\|^2.
\end{aligned} \tag{6.89}$$

Substitute the squared norm $\|\mathbf{e}_t\|^2$ terms in (6.89) by the upper bound in (6.62). It follows from this substitution and regrouping the terms that

$$\begin{aligned}
0 \leq \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2_{(\frac{2\alpha mM}{m+M} - \frac{\alpha}{\zeta} - \delta_t'\alpha\epsilon)\mathbf{I} + \alpha^2(\mathbf{I}-\mathbf{Z})} + \left(\frac{2\alpha}{m+M} - \frac{\delta_t'\phi\beta}{\hat{\lambda}}\right)\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2 \\
+ \left[\alpha\epsilon - \frac{\delta_t'\beta\epsilon^2}{(\beta-1)\hat{\lambda}} - \frac{\delta_t'\beta\phi\Gamma^2}{(\phi-1)\hat{\lambda}} - \alpha\zeta\Gamma^2\right]\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2.
\end{aligned} \tag{6.90}$$

Notice that if the inequality in (6.90) is satisfied, then the result in (6.89) holds which implies the result in (6.87) and the linear convergence claim in (6.81). To satisfy the inequality in (6.90) we need to make sure that the coefficients of the terms $\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2$, $\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2$, and $\|\nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}^*)\|^2$ are non-negative. Therefore, the inequality in (6.90) holds if

δ'_t satisfies

$$\frac{2\alpha mM}{m+M} - \frac{\alpha}{\zeta} - \delta'_t \alpha\epsilon \geq 0, \quad \frac{2\alpha}{m+M} \geq \frac{\delta'_t \phi\beta}{\hat{\lambda}}, \quad \alpha\epsilon \geq \frac{\delta'_t \beta\epsilon^2}{(\beta-1)\hat{\lambda}} + \frac{\delta'_t \beta\phi\Gamma^2}{(\phi-1)\hat{\lambda}} + \alpha\zeta\Gamma^2. \quad (6.91)$$

The conditions in (6.91) are satisfied if δ'_t is chosen as in (6.82). Thus, δ'_t in (6.82) satisfies the conditions in (6.91) and the claim in (6.81) holds. ■

The result in Theorem 10 shows linear convergence of the sequence $\|\mathbf{u}_t - \mathbf{u}^*\|^2_{\mathbf{G}}$ generated by ESOM where the factor of linear convergence is $1/(1+\delta')$. Notice that the positive constant ζ is chosen from the interval $((m+M)/2mM, \epsilon/\Gamma_t^2)$. This interval is non-empty if and only if the proximal parameter ϵ satisfies the condition $\epsilon > \Gamma_t^2(m+M)/2mM$. However, Γ_t also depends on ϵ which makes it unclear if there always exists a choice of ϵ that satisfies the inequality $\epsilon > \Gamma_t^2(m+M)/2mM$. In the following proposition, we prove that the interval $((m+M)/2mM, \epsilon/\Gamma_t^2)$ is non-empty for a proper choice of ϵ.

Proposition 8 *Consider ESOM as introduced in* (6.8)-(6.15). *Recall the definition of* Γ_t *in* (6.63). *If the constant* ϵ *is chosen such that*

$$\epsilon > \frac{m+M}{2mM}\left(2M + 2\alpha(1-c)\frac{M}{m}\right)^2, \quad (6.92)$$

then the inequality $\epsilon > \Gamma_t^2(m+M)/2mM$ *holds and the set* $((m+M)/2mM, \epsilon/\Gamma_t^2)$ *is non-empty.*

Proof: Note that the condition $\epsilon > \Gamma_t^2(m+M)/2mM$ is equivalent to

$$\Gamma_t < \frac{\sqrt{2\epsilon mM}}{\sqrt{m+M}}. \quad (6.93)$$

According to the definition of Γ_t, the expression $\rho := 2\alpha(1-c)/(2\alpha(1-c)+m+\epsilon)$, and the fact that $2M \geq \min\left\{2M, \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|\right\}$, we can write

$$\Gamma_t \leq 2M + (M + \epsilon + 2\alpha(1-c))\left(\frac{2\alpha(1-c)}{2\alpha(1-c)+m+\epsilon}\right)^{K+1}. \quad (6.94)$$

The results in (6.93) and (6.94) show that the inequality $\epsilon > \Gamma_t^2(m+M)/2mM$ holds if the following inequality holds,

$$2M + (M + \epsilon + 2\alpha(1-c))\left(\frac{2\alpha(1-c)}{2\alpha(1-c)+m+\epsilon}\right)^{K+1} < \frac{\sqrt{2\epsilon mM}}{\sqrt{m+M}}. \quad (6.95)$$

Thus, if the condition in (6.95) holds then we have $\epsilon > \Gamma_t^2(m+M)/2mM$. Note that

$(2\alpha(1-c)/(2\alpha(1-c)+m+\epsilon))^{K+1} \leq 2\alpha(1-c)/(2\alpha(1-c)+m+\epsilon)$ for any $K \geq 0$. Thus, if the following inequality is satisfied the inequality in (6.95) is also valid,

$$2M + (M+\epsilon+2\alpha(1-c))\left[\frac{2\alpha(1-c)}{2\alpha(1-c)+m+\epsilon}\right] < \frac{\sqrt{2\epsilon mM}}{\sqrt{m+M}}. \tag{6.96}$$

Considering that $m < M$ and $2\alpha(1-c)+\epsilon > 0$, we obtain that that $(M+\epsilon+2\alpha(1-c))/(m+\epsilon+2\alpha(1-c)) \leq M/m$. Replacing $(M+\epsilon+2\alpha(1-c))/(m+\epsilon+2\alpha(1-c))$ in (6.96) by the upper bound M/m implies that

$$2M + 2\alpha(1-c)\frac{M}{m} < \frac{\sqrt{2\epsilon mM}}{\sqrt{m+M}}. \tag{6.97}$$

Note that if the condition in (6.97) holds then the condition in (6.96) is satisfied. The result in (6.97) shows that if ϵ satisfies

$$\epsilon > \frac{m+M}{2mM}\left(2M + 2\alpha(1-c)\frac{M}{m}\right)^2, \tag{6.98}$$

then the inequality in (6.97) and consequently the inequalities in (6.96) and (6.95) hold true which follows that the condition $\epsilon > \Gamma_t^2(m+M)/2mM$ is satisfied. ■

It follows from the result in Theorem 10 that the sequence of primal variables $\mathbf{w}_t$ converges to the optimal argument $\mathbf{w}^*$ defined in (6.4).

Corollary 1 *Under the assumptions in Theorem 10, the sequence of squared errors* $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ *generated by ESOM converges to zero at a linear rate, i.e.,*

$$\|\mathbf{w}_t - \mathbf{w}^*\|^2 \leq \left(\frac{1}{1+\min_t\{\delta_t'\}}\right)^t \frac{\|\mathbf{u}_0 - \mathbf{u}^*\|_{\mathbf{G}}^2}{\alpha\epsilon}. \tag{6.99}$$

Proof: According to the definition of the sequence $\mathbf{u}_t$ and matrix $\mathbf{G}$, we can write $\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2 = \alpha\epsilon\|\mathbf{w}_t - \mathbf{w}^*\|^2 + \|\mathbf{s}_t - \mathbf{s}^*\|^2$ which implies that $\|\mathbf{w}_t - \mathbf{w}^*\|^2 \leq (1/\alpha\epsilon)\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2$. Considering this result and linear convergence of the sequence $\|\mathbf{u}_t - \mathbf{u}^*\|_{\mathbf{G}}^2$ in (6.81), the claim in (6.99) follows. ■

6.4.3 Convergence rates comparison

The expression for δ_t' in (6.82) verifies the intuition that the convergence rate of ESOM is slower than PMM. This is true, since the upper bounds for δ in PMM are larger than their equivalent upper bounds for δ_t' in ESOM. We obtain that δ_t' is smaller than δ which implies that the linear convergence factor $1/(1+\delta)$ of PMM is smaller than $1/(1+\delta_t')$ for ESOM.

Therefore, for all steps t, the linear convergence of PMM is faster than ESOM. Although, linear convergence factor of ESOM $1/(1+\delta_t')$ is larger than $1/(1+\delta)$ for PMM, as time passes the gap between these two constants becomes smaller. In particular, after a number of iterations $(L/2)\|\mathbf{w}_{t+1}-\mathbf{w}_t\|$ becomes smaller than $2M$, and Γ_t can be simplified as

$$\Gamma_t \leq \frac{L}{2}\|\mathbf{w}_{t+1}-\mathbf{w}_t\| + (2\alpha(1-c)+M+\epsilon)\,\rho^{K+1}. \tag{6.100}$$

The term $(L/2)\|\mathbf{w}_{t+1}-\mathbf{w}_t\|$ eventually approaches zero, while the second term $(2(1-c)/\alpha+M+\epsilon)\rho^{K+1}$ is constant. Although, the second term is not approaching zero, by proper choice of ρ and K, this term can become arbitrary close to zero. Notice that when Γ_t approaches zero, if we set $\zeta = 1/\Gamma_t$ the upper bounds in (6.82) for δ_t' approach the upper bounds for δ of PMM in (6.42).

Therefore, as time passes Γ_t becomes smaller, and the factor of linear convergence for ESOM $1/(1+\delta_t')$ becomes closer to the linear convergence factor of PMM $1/(1+\delta)$.

6.5 Numerical experiments

In this section, we compare the performances of ESOM, EXTRA, Decentralized Quadratically approximated ADMM (DQM), and Network Newton (NN). First, we consider a linear least squares problem and then we use the mentioned methods to solve a logistic regression problem.

6.5.1 Decentralized linear least squares

Consider a decentralized linear least squares problem where each agent $v \in \{1,\cdots,V\}$ holds its private measurement equation, $\mathbf{y}_v = \mathbf{M}_v\tilde{\mathbf{w}} + \boldsymbol{\nu}_v$, where $\mathbf{y}_v \in \mathbb{R}^{m_v}$ and $\mathbf{M}_v \in \mathbb{R}^{m_v \times p}$ are measured data, $\tilde{\mathbf{w}} \in \mathbb{R}^p$ is the unknown variable, and $\boldsymbol{\nu}_v \in \mathbb{R}^{m_v}$ is some unknown noise. The decentralized linear least squares estimates $\tilde{\mathbf{w}}$ by solving the optimization problem

$$\tilde{\mathbf{w}}^* = \underset{\tilde{\mathbf{w}}}{\operatorname{argmin}} \sum_{v=1}^{V} \|\mathbf{M}_v\tilde{\mathbf{w}} - \mathbf{y}_v\|_2^2. \tag{6.101}$$

The network in this experiment is randomly generated with connectivity ratio $r = 3/V$, where r is defined as the number of edges divided by the number of all possible ones, $V(V-1)/2$. We set $V = 20$, $p = 5$, and $m_v = 5$ for all $v = 1,\ldots,V$. The vectors $\mathbf{y}_v$ and matrices $\mathbf{M}_v$ as well as the noise vectors $\boldsymbol{\nu}_{(i)}$, for all i are generated following the standard normal distribution. We precondition the aggregated data matrices $\mathbf{M}_v$ so that the condition number of the problem is 10. The decision variables $\mathbf{w}_v$ are initialized as $\mathbf{w}_{v,0} = 0$ for all nodes $v = 1,\ldots,V$ and the initial distance to the optimal is $\|\mathbf{w}_{v,0} - \tilde{\mathbf{w}}^*\| = 100$.

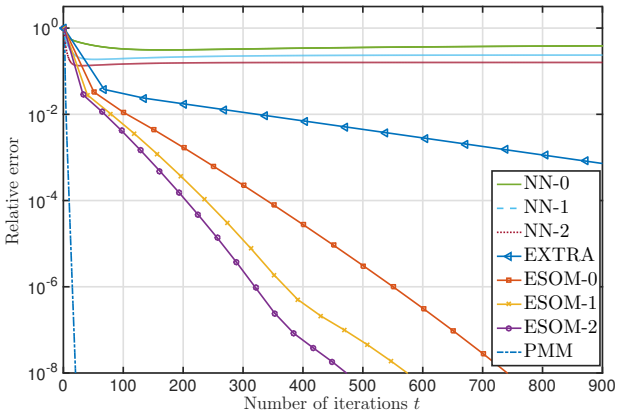


Figure 6.1: Relative error $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}_0 - \mathbf{w}^*\|$ of EXTRA, ESOM-K, NN-K, and PMM versus number of iterations for the least squares problem. Using a larger K for ESOM-K leads to faster convergence and makes the convergence path closer to the one for PMM.

We use Metropolis constant edge weight matrix as the mixing matrix $\mathbf{W}$ in all experiments. We run PMM, EXTRA, and ESOM-K with fixed hand-optimized stepsizes α. The best choices of α for ESOM-0, ESOM-1, and ESOM-2 are $\alpha = 0.03$, $\alpha = 0.04$, and $\alpha = 0.05$, respectively. The stepsize $\alpha = 0.1$ leads to the best performance for EXTRA which is considered in the numerical experiments. Notice that for variations of NN-K, there is no optimal choice of stepsize – smaller stepsize leads to more accurate but slow convergence, while large stepsize accelerates the convergence but to a less accurate neighborhood of the optimal solution. Therefore, for NN-0, NN-1, and NN-2 we set $\alpha = 0.001$, $\alpha = 0.008$, and $\alpha = 0.02$, respectively. Although the PMM algorithm is not implementable in a decentralized fashion, we use its convergence path – which is generated in a centralized manner – as our benchmark. The choice of stepsize for PMM is $\alpha = 2$.

Fig. 6.1 illustrates the relative error $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}_0 - \mathbf{w}^*\|$ versus the number of iterations. Notice that the vector $\mathbf{w}_t$ is the concatenation of the local vectors $\mathbf{w}_{v,t}$ and the optimal vector $\mathbf{w}^*$ is defined as $\mathbf{w}^* = [\tilde{\mathbf{w}}^*; \ldots; \tilde{\mathbf{w}}^*] \in \mathbb{R}^{Vp}$. Observe that all the variations of NN-K fail to converge to the optimal argument and they converge linearly to a neighborhood of the optimal solution $\mathbf{w}^*$. Among the decentralized algorithms with exact linear convergence rate, EXTRA has the worst performance and all the variations of ESOM-K outperform EXTRA. Recall that the problem condition number is 10 in our experiment and the difference between EXTRA and ESOM-K is more significant for problems with larger condition numbers. Further, choosing a larger value of K for ESOM-K leads to faster convergence and as we increase K the convergence path of ESOM-K approaches the convergence path of PMM.

EXTRA requires one round of communications per iteration, while NN-K and ESOM-

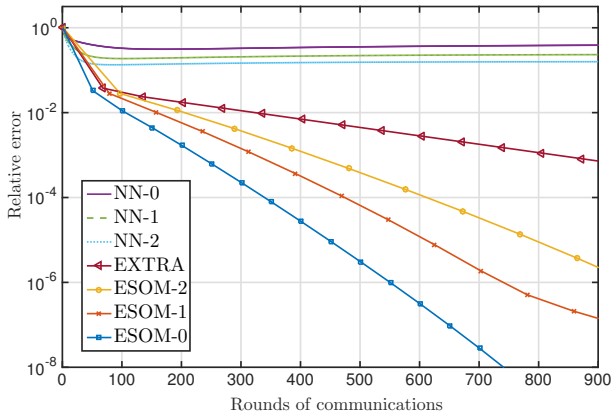


Figure 6.2: Relative error $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}_0 - \mathbf{w}^*\|$ of EXTRA, ESOM-K, NN-K, and PMM versus rounds of communications with neighboring nodes for the least squares problem. ESOM-0 is the most efficient algorithm in terms of communication cost among all the methods.

K require $K+1$ rounds of local communications per iteration. Thus, convergence paths of these methods in terms of rounds of communications might be different from the ones in Fig. 6.1. The convergence paths of NN, ESOM, EXTRA in terms of rounds of local communications are shown in Fig. 6.2. In this plot we ignore PMM, since it requires infinite rounds of communications per iteration. The main difference between Figs. 6.1 and 6.2 is in the performances of ESOM-0, ESOM-1, and ESOM-2. All of the variations of ESOM outperform EXTRA in terms of rounds of communications, while the best performance belongs to ESOM-0. This observation shows that increasing the approximation level K does not necessary improve the performance of ESOM-K in terms of communication cost.

6.5.2 Decentralized logistic regression

We consider the application of ESOM for solving a logistic regression problem in a form

$$\tilde{\mathbf{w}}^* := \underset{\tilde{\mathbf{w}}\in\mathbb{R}^p}{\operatorname{argmin}} \frac{\lambda}{2}\|\tilde{\mathbf{w}}\|^2 + \sum_{v=1}^{V}\sum_{i=1}^{q_v} \ln\left(1+\exp\left(-(\mathbf{x}_{vi}^T\tilde{\mathbf{w}})y_{vi}\right)\right), \tag{6.102}$$

where every agent v has access to q_v training samples $(\mathbf{x}_{vi}, y_{vi}) \in \mathbb{R}^p \times \{-1,+1\}$, $i = 1,\cdots,q_v$, including explanatory/feature variables $\mathbf{x}_{vi}$ and binary outputs/outcomes y_{vi}. The regularization term $(\lambda/2)\|\tilde{\mathbf{w}}\|^2$ is added to avoid overfitting where $\lambda > 0$. Hence, in the decentralized setting the local objective function f_v of node v is given by

$$f_v(\tilde{\mathbf{w}}) = \frac{\lambda}{2V}\|\tilde{\mathbf{w}}\|^2 + \sum_{i=1}^{q_v} \ln\left(1+\exp\left(-(\mathbf{x}_{vi}^T\tilde{\mathbf{w}})y_{vi}\right)\right). \tag{6.103}$$

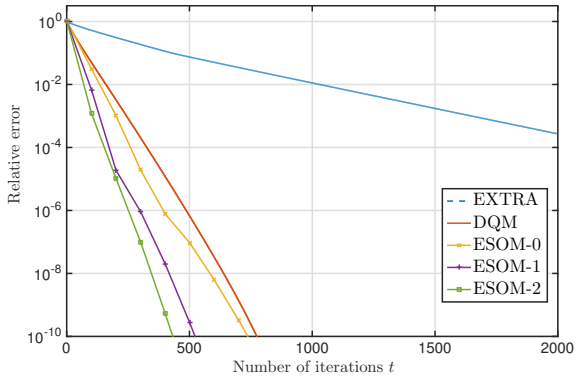


Figure 6.3: Relative error $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}_0 - \mathbf{w}^*\|$ of EXTRA, ESOM-K, and DQM versus number of iterations for the logistic regression problem. EXTRA is significantly slower than the ESOM methods. The proposed methods (ESOM-K) outperform DQM.

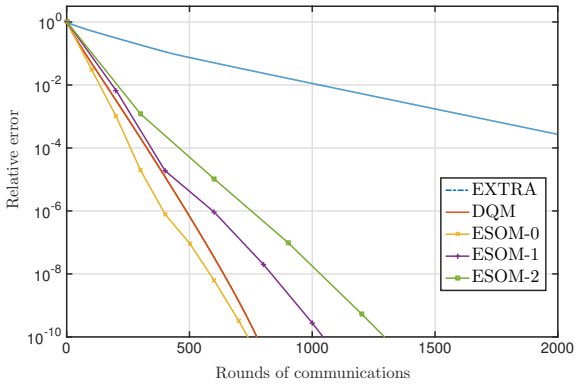


Figure 6.4: Relative error $\|\mathbf{w}_t - \mathbf{w}^*\|/\|\mathbf{w}_0 - \mathbf{w}^*\|$ of EXTRA, ESOM-K, and DQM versus rounds of communications for the logistic regression problem. ESOM-0 has the best performance in terms of rounds of communications and it outperforms DQM.

The settings are as follows. The connected network is randomly generated with $V = 20$ agents and connectivity ratio $r = 3/V$. Each agent holds 3 samples, i.e., $q_v = 3$, for all v. The dimension of sample vectors $\mathbf{x}_{vi}$ is $p = 3$. The samples are randomly generated, and the optimal logistic classifier $\tilde{\mathbf{w}}^*$ is pre-computed through centralized adaptive gradient method. We use Metropolis constant edge weight matrix as the mixing matrix $\mathbf{W}$ in ESOM-K. The stepsize α for ESOM-0, ESOM-1, ESOM-2, EXTRA, and DQM are hand-optimized and the best of each is used for the comparison.

Fig. 6.3 and Fig 6.4 showcase the convergence paths of ESOM-0, ESOM-1, ESOM-2, EXTRA, and DQM versus number of iterations and rounds of communications, respectively. The results match the observations for the least squares problem in Fig. 6.1 and Fig. 6.2. Different versions of ESOM-K converge faster than EXTRA both in terms of communication

cost and number of iterations. Moreover, ESOM-2 converges faster than ESOM-1 and ESOM-0 in terms of number of iterations, while ESOM-0 has the best performance in terms of communication cost for achieving a target accuracy. Comparing the convergence paths of ESOM-0, ESOM-1, and ESOM-2 with DQM shows that number of iterations required for the convergence of DQM is larger than the required iterations for ESOM-0, ESOM-1, and ESOM-2. In terms of communication cost, DQM has a better performance relative to ESOM-1 and ESOM-2, while ESOM-0 is the most efficient algorithm.

Chapter 7

Decentralized stochastic optimization via gradient averaging

7.1 Context and background

In Chapters 5 and 6, we studied methods for solving ERM problems via decentralized optimization, and, in particular, focused on the use of Network Newton and ESOM algorithms. These two methods alongside with the other decentralized optimization methods mentioned in previous chapters build on the fact that local gradients are computationally affordable for nodes in the network. In this chapter, we concentrate on the cases that the number of assigned samples to each node is very large and computation of local gradients is beyond the computational capacity of nodes.

To explain this scenario, consider a training set of size N where the samples are distributed among nodes a connected network of size V. Further, consider a variable $\mathbf{w} \in \mathbb{R}^p$ and a local objective function $f_v : \mathbb{R}^p \to \mathbb{R}$ associated to node v. The local objective function $f_v(\mathbf{w})$ is defined as the average of q_v local instantaneous functions $f_{v,i}(\mathbf{w})$ that can be individually evaluated at node v. Agents cooperate to solve the global optimization problem

$$\tilde{\mathbf{w}}^* := \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{v=1}^{V} f_v(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{v=1}^{V} \frac{1}{q_v} \sum_{i=1}^{q_v} f_{v,i}(\mathbf{w}). \tag{7.1}$$

The formulation in (7.1) models a training set with a total of $N = \sum_{v=1}^{V} q_v$ training samples that are distributed among the V agents for parallel processing conducive to the determination of the optimal classifier $\tilde{\mathbf{w}}^*$ [6, 26, 118]. Although we make no formal assumption, in cases of practical importance the total number of training samples $\sum_{v=1}^{V} q_v$ is very large, and, therefore, the number of elements q_v available at a specific node is large. Our interest here is in solving (7.1) with a method that has the following three properties:

- Decentralized; nodes operate on their local functions and communicate with neighbors only.
- Stochastic; nodes determine a descent direction by evaluating only one out of the q_v functions $f_{v,i}$ at each iteration.
- Linear convergence rate; the expected distance to the optimum is scaled by a subunit factor at each iteration.

Decentralized optimization is relatively mature and various methods are known with complementary advantages. These methods include decentralized gradient descent (DGD) [44, 80, 126], network Newton [64, 65], decentralized dual averaging [33, 119], the exact first order algorithm (EXTRA) [111], as well as the alternating direction method of multipliers (ADMM) [19, 41, 103, 112] and its linearized variants [53, 54, 75]. The ADMM, its variants, and EXTRA converge linearly to the optimal argument but DGD, network Newton, and decentralized dual averaging have sublinear convergence rates. Of particular importance to this chapter, is the fact that DGD has (inexact) linear convergence to a neighborhood of the optimal argument when it uses constant stepsizes. It can achieve exact convergence by using diminishing stepsizes, but the convergence rate degrades to sublinear. This lack of linear convergence is solved by EXTRA through the use of iterations that rely on information of two consecutive steps [111].

All of the algorithms mentioned above require the computationally costly evaluation of the local gradients $\nabla f_v(\mathbf{w}) = (1/q_v) \sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{w})$. This cost can be avoided by stochastic decentralized algorithms that reduce computational cost of iterations by substituting all local gradients with their stochastic approximations. This reduces the computational cost per iteration but results in sublinear convergences rates of order $O(1/t)$ even if the corresponding deterministic algorithm exhibits linear convergence. This is a drawback that also exists in centralized stochastic optimization where linear convergence rates in expectation are established by decreasing the variance of the stochastic gradient approximation [31, 45, 47, 49, 104, 109]. In this chapter we build on the ideas of the stochastic averaging gradient (SAG) algorithm [104] and its unbiased version SAGA [31]. Both of these algorithms use the idea of stochastic incremental averaging gradients. At each iteration only one of the stochastic gradients is updated and the average of all of the most recent stochastic gradients is used for estimating the gradient.

In this chapter we aim to present the decentralized double stochastic averaging gradient (DSA) method, a novel decentralized stochastic algorithm for solving (7.1). The method exploits a new interpretation of EXTRA as a saddle point method and uses stochastic averaging gradients in lieu of gradients. DSA is *decentralized* because it is implementable in a network setting where nodes can communicate only with their neighbors. It is *double*

because iterations utilize the information of two consecutive iterates. It is *stochastic* because the gradient of only one randomly selected function is evaluated at each iteration and it is an *averaging* method because it uses an average of stochastic gradients to approximate the local gradients. DSA is proven to converge linearly to the optimal argument $\tilde{\mathbf{w}}^*$ in expectation when the local instantaneous functions $f_{v,i}$ are strongly convex, with Lipschitz continuous gradients. This is in contrast to all other decentralized stochastic methods to solve (7.1) that converge at sublinear rates.

We begin the chapter with a discussion of DGD, EXTRA and stochastic averaging gradient. With these definitions in place we define the DSA algorithm by replacing the gradients used in EXTRA by stochastic averaging gradients (Section 7.2). We follow with a digression on the limit points of DGD and EXTRA iterations to explain the reason why DGD does not achieve exact convergence but EXTRA is expected to do so (Section 7.2.1). A reinterpretation of EXTRA as a saddle point method that solves for the critical points of the augmented Lagrangian of a constrained optimization problem equivalent to (7.1) is then introduced. It follows from this reinterpretation that DSA is a stochastic saddle point method (Section 7.2.2). The fact that DSA is a stochastic saddle point method is the critical enabler of the subsequent convergence analysis (Section 7.3). In particular, it is possible to guarantee that strong convexity and gradient Lipschitz continuity of the local instantaneous functions $f_{v,i}$ imply that a Lyapunov function associated with the sequence of iterates generated by DSA converges linearly to its optimal value in expectation (Theorem 11). Linear convergence in expectation of the local iterates to the optimal argument $\tilde{\mathbf{w}}^*$ of (7.1) follows as a trivial consequence (Corollary 2). We complement this result by showing convergence of all the local variables to the optimal argument $\tilde{\mathbf{w}}^*$ with probability 1 (Theorem 12).

The advantages of DSA relative to a group of stochastic and deterministic alternatives in solving a logistic regression problem are then studied in numerical experiments (Section 7.4). These results demonstrate that DSA is the only decentralized stochastic algorithm that reaches the optimal solution with a linear convergence rate. We further show that DSA outperforms deterministic algorithms when the metric is the number of times that elements of the training set are evaluated. The behavior of DSA for different network topologies is also evaluated.

Notation Lowercase boldface $\mathbf{v}$ denotes a vector and uppercase boldface $\mathbf{A}$ a matrix. For column vectors $\mathbf{w}_1, \ldots, \mathbf{w}_V$ we use the notation $\mathbf{w} = [\mathbf{w}_1; \ldots; \mathbf{w}_V]$ to represent the stack column vector $\mathbf{w}$. We use $\|\mathbf{v}\|$ to denote the Euclidean norm of vector $\mathbf{v}$ and $\|\mathbf{A}\|$ to denote the Euclidean norm of matrix $\mathbf{A}$. For a vector $\mathbf{v}$ and a positive definite matrix $\mathbf{A}$, the $\mathbf{A}$-weighted norm is defined as $\|\mathbf{v}\|_{\mathbf{A}} := \sqrt{\mathbf{v}^T\mathbf{A}\mathbf{v}}$. The null space of matrix $\mathbf{A}$ is denoted by $\text{null}(\mathbf{A})$ and the span of a vector by $\text{span}(\mathbf{w})$. The operator $\mathbb{E}_{\mathbf{w}}[\cdot]$ stands for expectation over random variable $\mathbf{w}$ and $\mathbb{E}[\cdot]$ for expectation with respect to the distribution

of a stochastic process.

7.2 Decentralized double stochastic averaging gradient

Consider a connected network that contains V nodes such that each node v can only communicate with peers in its neighborhood $\mathcal{N}_v$. Define $\mathbf{w}_v \in \mathbb{R}^p$ as a local copy of the variable $\mathbf{w}$ that is kept at node v. In decentralized optimization, agents try to minimize their local functions $f_v(\mathbf{w}_v)$ while ensuring that their local variables $\mathbf{w}_v$ coincide with the variables $\mathbf{w}_u$ of all neighbors $u \in \mathcal{N}_v$ – which, given that the network is connected, ensures that the variables $\mathbf{w}_v$ of all nodes are the same and renders the problem equivalent to (7.1). DGD is a well known method for decentralized optimization that relies on the introduction of nonnegative weights $w_{vu} \geq 0$ that are not null if and only if $u = v$ or if $u \in \mathcal{N}_v$. Letting $t \in \mathbb{N}$ be a discrete time index and α a given stepsize, DGD is defined by the recursion

$$\mathbf{w}_v^{t+1} = \sum_{u=1}^{V} w_{vu} \mathbf{w}_u^t - \alpha \nabla f_v(\mathbf{w}_v^t), \qquad v = 1, \ldots, V. \tag{7.2}$$

Since $w_{vu} = 0$ when $u \neq v$ and $u \notin \mathcal{N}_v$, it follows from (7.2) that node v updates $\mathbf{w}_v$ by performing an average over the variables $\mathbf{w}_u^t$ of its neighbors $u \in \mathcal{N}_v$ and its own $\mathbf{w}_v^t$, followed by descent through the negative local gradient $-\nabla f_v(\mathbf{w}_v^t)$. If a constant stepsize is used, DGD iterates $\mathbf{w}_v^t$ approach a neighborhood of the optimal argument $\tilde{\mathbf{w}}^*$ of (7.1) but don't converge exactly. To achieve exact convergence diminishing stepsizes are used but the resulting convergence rate is sublinear [80].

EXTRA is a method that resolves either of these issues by mixing two consecutive DGD iterations with different weight matrices and opposite signs. To be precise, introduce a second set of weights $\tilde{w}_{vu}$ with the same properties as the weights w_{vu} and define EXTRA through the recursion

$$\mathbf{w}_v^{t+1} = \mathbf{w}_v^t + \sum_{m=1}^{N} w_{vu} \mathbf{w}_u^t - \sum_{m=1}^{N} \tilde{w}_{vu} \mathbf{w}_u^{t-1} - \alpha \left[\nabla f_v(\mathbf{w}_v^t) - \nabla f_v(\mathbf{w}_v^{t-1}) \right], \quad v = 1, \ldots, V. \tag{7.3}$$

Observe that (7.3) is well defined for $t > 0$. For $t = 0$ we utilize the regular DGD iteration in (7.2). In the nomenclature of this chapter we say that EXTRA performs a decentralized double gradient descent step because it operates in a decentralized manner while utilizing a difference of two gradients as descent direction. Minor modification as it is, the use of this gradient difference in lieu of simple gradients, endows EXTRA with exact linear convergence to the optimal argument $\tilde{\mathbf{w}}^*$ under mild assumptions [111].

If we recall the definitions of the local functions $f_v(\mathbf{w}_v)$ and the instantaneous local

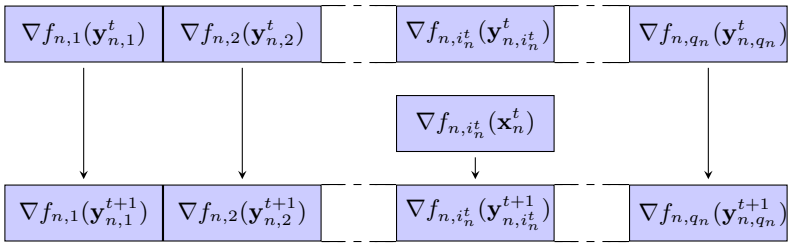


Figure 7.1: Stochastic averaging gradient table at node v. At each iteration t a random local instantaneous gradient $\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)$ is updated by $\nabla f_{v,i_v^t}(\mathbf{w}_v^t)$. The rest of the local instantaneous gradients remain unchanged, i.e., $\nabla f_{v,i}(\mathbf{y}_{v,i}^{t+1}) = \nabla f_{v,i}(\mathbf{y}_{v,i}^t)$ for $i \neq i_v^t$. This list is used to compute the stochastic averaging gradient in (7.7).

functions $f_{v,i}(\mathbf{w}_v)$ available at node v, the implementation of EXTRA requires that each node v computes the full gradient of its local objective function f_v at $\mathbf{w}_v^t$ as

$$\nabla f_v(\mathbf{w}_v^t) = \frac{1}{q_v} \sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{w}_v^t). \tag{7.4}$$

This is computationally expensive when the number of instantaneous functions q_v is large. To resolve this issue, local stochastic gradients can be substituted for the local objective functions gradients in (7.3). These stochastic gradients approximate the gradient $\nabla f_v(\mathbf{w}_v)$ of node v by randomly choosing one of the instantaneous functions gradients $\nabla f_{v,i}(\mathbf{w}_v)$. If we let $i_v^t \in \{1, \dots q_v\}$ denote a function index that we choose at time t at node v uniformly at random and independently of the history of the process, then the stochastic gradient is defined as

$$\hat{\mathbf{s}}_n(\mathbf{w}_v^t) := \nabla f_{v,i_v^t}(\mathbf{w}_v^t). \tag{7.5}$$

We can then write a stochastic version of EXTRA by replacing $\nabla f_v(\mathbf{w}_v^t)$ by $\hat{\mathbf{s}}_n(\mathbf{w}_v^t)$ and $\nabla f_v(\mathbf{w}_v^{t-1})$ by $\hat{\mathbf{s}}_n(\mathbf{w}_v^{t-1})$. Such an algorithm would have a small computational cost per iteration. On the negative side, it either has a linear convergence to a neighborhood of the optimal solution $\mathbf{w}^*$ with constant stepsize α, or it would converge sublinearly to the optimal argument when the stepisize diminishes as time passes. Here however, we want to design an algorithm with low computational complexity that converges linearly to the exact solution $\mathbf{w}^*$.

To reduce this noise we propose the use of stochastic averaging gradients instead ([31]). The idea is to maintain a list of gradients of all instantaneous functions in which one randomly chosen element is replaced at each iteration and to use an average of the elements of this list for gradient approximation; see Figure 7.1. Formally, define the variable $\mathbf{y}_{v,i} \in \mathbb{R}^p$ to represent the iterate value the last time that the instantaneous gradient of function $f_{v,i}$ was evaluated. If we let $i_v^t \in \{1, \dots, q_v\}$ denote the function index chosen at time t at node

Algorithm 9 DSA algorithm at node v

Require: Vectors $\mathbf{w}_v^0$. Gradient table initialized with gradients $\nabla f_{v,i}(\mathbf{y}_{v,i}^0)$ where $\mathbf{y}_{v,i}^0 = \mathbf{w}_v^0$.

1: **for** $t = 0, 1, 2, \ldots$ **do**

2: Exchange variable $\mathbf{w}_v^t$ with neighboring nodes $u \in \mathcal{N}_n$.

3: Choose i_v^t uniformly at random from the set $\{1, \ldots, q_n\}$.

4: Compute and store $\hat{\mathbf{g}}_v^t$ [cf. (7.7)] $\hat{\mathbf{g}}_v^t = \nabla f_{v,i_v^t}(\mathbf{w}_v^t) - \nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t) + \dfrac{1}{q_v}\displaystyle\sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{y}_{v,i}^t)$

5: Set $\mathbf{y}_{n,i_v^t}^{t+1} = \mathbf{w}_v^t$ and store $\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^{t+1}) = \nabla f_{v,i_v^t}(\mathbf{w}_v^t)$ in i_v^t gradient table position.

6: **if** $t = 0$ **then**

7: Update variable $\mathbf{w}_v^t$ [cf. (7.9)]: $\mathbf{w}_v^{t+1} = \displaystyle\sum_{u=1}^{V} w_{vu}\mathbf{w}_v^t - \alpha\hat{\mathbf{g}}_v^t$

8: **else**

9: Update variable $\mathbf{w}_v^t$ [cf. (7.8)]: $\mathbf{w}_v^{t+1} = \mathbf{w}_v^t + \displaystyle\sum_{u=1}^{V} w_{vu}\mathbf{w}_v^t - \sum_{u=1}^{V} \tilde{w}_{nm}\mathbf{w}_v^{t-1} - \alpha\left[\hat{\mathbf{g}}_v^t - \hat{\mathbf{g}}_v^{t-1}\right]$

10: **end if**

11: **end for**

v, as we did in (7.5), the variables $\mathbf{y}_{v,i}$ are updated recursively as

$$\mathbf{y}_{v,i}^{t+1} = \mathbf{w}_v^t, \quad \text{if } i = i_v^t, \qquad \mathbf{y}_{v,i}^{t+1} = \mathbf{y}_{v,i}^t, \quad \text{if } i \neq i_v^t. \tag{7.6}$$

With these definitions in hand we can define the stochastic averaging gradient at node v as

$$\hat{\mathbf{g}}_v^t := \nabla f_{v,i_v^t}(\mathbf{w}_v^t) - \nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t) + \frac{1}{q_v}\sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{y}_{v,i}^t). \tag{7.7}$$

Observe that to implement (7.7) the gradients $\nabla f_{v,i}(\mathbf{y}_{v,i}^t)$ are stored in the local gradient table shown in Figure 7.1.

The DSA algorithm is a variation of EXTRA that substitutes the local gradients $\nabla f_v(\mathbf{w}_v^t)$ in (7.3) for the local stochastic average gradients $\hat{\mathbf{g}}_v^t$ in (7.7),

$$\mathbf{w}_v^{t+1} = \mathbf{w}_v^t + \sum_{m=1}^{N} w_{vu}\mathbf{w}_u^t - \sum_{m=1}^{N} \tilde{w}_{vu}\mathbf{w}_u^{t-1} - \alpha\left[\hat{\mathbf{g}}_v^t - \hat{\mathbf{g}}_v^{t-1}\right]. \tag{7.8}$$

The DSA initial update is given by applying the same substitution for the update of DGD in (7.2) as

$$\mathbf{w}_v^1 = \sum_{m=1}^{N} w_{vu}\mathbf{w}_u^0 - \alpha\ \hat{\mathbf{g}}_v^0. \tag{7.9}$$

DSA is summarized in Algorithm 9 for $t \geq 0$. The DSA update in (7.8) is implemented in Step 9. This step requires access to the local iterates $\mathbf{w}_m^t$ of neighboring nodes $u \in \mathcal{N}_v$ which are collected in Step 2. Furthermore, implementation of the DSA update also requires

access to the stochastic averaging gradients $\hat{\mathbf{g}}_n^{t-1}$ and $\hat{\mathbf{g}}_v^t$. The latter is computed in Step 4 and the former is computed and stored at the same step in the previous iteration. The computation of the stochastic averaging gradients requires the selection of the index i_v^t. This index is chosen uniformly at random in Step 3. Determination of stochastic averaging gradients also necessitates access and maintenance of the gradients table in Figure 7.1. The i_v^t element of this table is updated in Step 5 by replacing $\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)$ with $\nabla f_{v,i_v^t}(\mathbf{w}_v^t)$, while the other vectors remain unchanged. To implement the first DSA iteration at time $t = 0$ we have to perform the update in (7.9) instead of the update in (7.8) as in Step 7. Further observe that the auxiliary variables $\mathbf{y}_{v,i}^0$ are initialized to the initial iterate $\mathbf{w}_v^0$. This implies that the initial values of the stored gradients are $\nabla f_{v,i}(\mathbf{y}_{v,i}^0) = \nabla f_{v,i}(\mathbf{w}_v^0)$.

We point out that the weights w_{vu} and $\tilde{w}_{vu}$ can't be arbitrary. If we define weight matrices $\mathbf{W}$ and $\tilde{\mathbf{W}}$ with elements w_{vu} and $\tilde{w}_{vu}$, respectively, they have to satisfy conditions that we state as an assumption for future reference.

Assumption 14 *The weight matrices* $\mathbf{W}$ *and* $\tilde{\mathbf{W}}$ *must satisfy the following properties*

(a) Both are symmetric, $\mathbf{W} = \mathbf{W}^T$ *and* $\tilde{\mathbf{W}} = \tilde{\mathbf{W}}^T$.

(b) The null space of $\mathbf{I} - \tilde{\mathbf{W}}$ *includes the span of* $\mathbf{1}$*, i.e.,* $null(\mathbf{I} - \tilde{\mathbf{W}}) \supseteq span(\mathbf{1})$*, the null space of* $\mathbf{I} - \mathbf{W}$ *is the span of* $\mathbf{1}$*, i.e.,* $null(\mathbf{I} - \mathbf{W}) = span(\mathbf{1})$*, and the null space of the difference* $\tilde{\mathbf{W}} - \mathbf{W}$ *is the span of* $\mathbf{1}$*, i.e.,* $null(\tilde{\mathbf{W}} - \mathbf{W}) = span(\mathbf{1})$.

(c) They satisfy the spectral ordering $\mathbf{W} \preceq \tilde{\mathbf{W}} \preceq (\mathbf{I} + \mathbf{W})/2$ *and the matrix* $\tilde{\mathbf{W}}$ *is positive definite* $\mathbf{0} \prec \tilde{\mathbf{W}}$.

Requiring the matrix $\mathbf{W}$ to be symmetric and with specific null space properties is necessary to let all agents converge to the same optimal variable. Analogous properties are necessary in DGD and are not difficult to satisfy. The condition on spectral ordering is specific to EXTRA but is not difficult to satisfy either. E.g., if we have a matrix $\mathbf{W}$ that satisfies all the conditions in Assumption 14, the weight matrix $\tilde{\mathbf{W}} = (\mathbf{I} + \mathbf{W})/2$ makes Assumption 14 valid.

We also point that, as written in (7.7), computation of local stochastic averaging gradients $\hat{\mathbf{g}}_v^t$ is costly because it requires evaluation of the sum $\sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{y}_{v,i}^t)$ at each iteration. To be more precise, if we implement the update in (7.7) naively, at each iteration we should compute the sum $\sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{y}_{v,i}^t)$ which has a computational cost of the order $O(q_v)$. This cost can be avoided by updating the sum at each iteration with the recursive formula

$$\sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{y}_{v,i}^t) \;=\; \sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{y}_{v,i}^{t-1}) + \nabla f_{v,i_v^{t-1}}(\mathbf{w}_v^{t-1}) - \nabla f_{v,i_v^{t-1}}(\mathbf{y}_{v,i_v^{t-1}}^{t-1}). \tag{7.10}$$

Using the update in (7.10), we can update the sum $\sum_{i=1}^{q_v} \nabla f_{v,i}(\mathbf{y}_{v,i}^t)$ required for (7.7) in

a computationally efficient manner. Important properties and interpretations of EXTRA and DSA are presented in the following sections after pertinent remarks.

Remark 11 The local stochastic averaging gradients in (7.7) are unbiased estimates of the local gradients $\nabla f_v(\mathbf{w}_v^t)$. Indeed, if we let $\mathcal{F}_t$ measure the history of the system up until time t we have that the sum in (7.7) is deterministic given this sigma-algebra. This observation implies that the conditional expectation $\mathbb{E}\left[(1/q_v)\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{y}_{v,i}^t) \mid \mathcal{F}^t\right]$ can be simplified as $(1/q_v)\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{y}_{v,i}^t)$. Thus, the conditional expectation of the stochastic averaging gradient is,

$$\mathbb{E}\left[\hat{\mathbf{g}}_v^t \,\middle|\, \mathcal{F}^t\right] = \mathbb{E}\Big[\nabla f_{v,i_v^t}(\mathbf{w}_v^t) \,\Big|\, \mathcal{F}^t\Big] - \mathbb{E}\Big[\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t) \,\Big|\, \mathcal{F}^t\Big] + \frac{1}{q_v}\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{y}_{v,i}^t). \tag{7.11}$$

With the index i_v^t chosen equiprobably from the set $\{1,\ldots,q_v\}$, the expectation of the second term in (7.11) is the same as the sum in the last term – each of the indexes is chosen with probability $1/q_v$. In other words, we can write $\mathbb{E}\Big[\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t) \,\Big|\, \mathcal{F}^t\Big] = (1/q_v)\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{y}_{v,i}^t)$. Therefore, these two terms cancel out each other and, since the expectation of the first term in (7.11) is simply $\mathbb{E}\big[\nabla f_{v,i_v^t}(\mathbf{w}_v^t) \,\big|\, \mathcal{F}^t\big] = (1/q_v)\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{w}_v^t) = \nabla f_v(\mathbf{w}_v^t)$, we can simplify (7.11) to

$$\mathbb{E}\left[\hat{\mathbf{g}}_v^t \,\middle|\, \mathcal{F}^t\right] = \nabla f_v(\mathbf{w}_v^t). \tag{7.12}$$

The expression in (7.12) means, by definition, that $\hat{\mathbf{g}}_v^t$ is an unbiased estimate of $\nabla f_v(\mathbf{w}_v^t)$ when the history $\mathcal{F}^t$ is given.

Remark 12 The local stochastic averaging gradient $\hat{\mathbf{g}}_v^t$ at node v contains three terms. The first two terms $\nabla f_{v,i_v^t}(\mathbf{w}_v^t)$ and $\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)$ are the new and old gradients of the chosen objective function f_{v,i_v^t} at node v, respectively. The last term $(1/q_v)\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{y}_{v,i}^t)$ is the average of the average of all the instantaneous gradients available at node v. This update can be considered as a localized version of the stochastic averaging gradient update in the SAGA algorithm [31]. Notice that instead of the difference $\nabla f_{v,i_v^t}(\mathbf{w}_v^t) - \nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)$ in (7.7) we could use the difference $(\nabla f_{v,i_v^t}(\mathbf{w}_v^t) - \nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t))/q_v$ which would lead to stochastic averaging gradient suggested in the SAG algorithm [104]. As studied in [31], both of these approximations lead to a variance reduction method. The one suggested by SAGA is an unbiased estimator of the exact gradient $(1/q_v)\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{w}_v^t)$, while the one suggested by SAG is a biased estimator of the gradient with smaller variance. Since the analysis of the unbiased estimator suggested by SAGA is simpler, we use its idea to define the local stochastic averaging gradient $\hat{\mathbf{g}}_v^t$ in (7.7).

7.2.1 Limit points of DGD and EXTRA

The derivation of EXTRA hinges on the observation that the optimal argument of (7.1) is not a fixed point of the DGD iteration in (7.2) but is a fixed point of the iteration in (7.3). To explain this point define $\mathbf{w} := [\mathbf{w}_1; \ldots; \mathbf{w}_V] \in \mathbb{R}^{Vp}$ as a vector that concatenates the local iterates $\mathbf{w}_v$ and the aggregate function $f : \mathbb{R}^{Vp} \to \mathbb{R}$ as the one that takes values $f(\mathbf{w}) = f(\mathbf{w}_1, \ldots, \mathbf{w}_V) := \sum_{v=1}^{V} f_v(\mathbf{w}_v)$. Decentralized optimization entails the minimization of $f(\mathbf{w})$ subject to the constraint that all local variables are equal,

$$\begin{aligned} \mathbf{w}^* := \operatorname{argmin} f(\mathbf{w}) = f(\mathbf{w}_1, \ldots, \mathbf{w}_V) = \sum_{v=1}^{V} f_v(\mathbf{w}_v), \\ \text{s. t.} \qquad \mathbf{w}_v = \mathbf{w}_u, \quad \text{for all } v, u. \end{aligned} \tag{7.13}$$

The problems in (7.1) and (7.13) are equivalent in the sense that the vector $\mathbf{w}^* \in \mathbb{R}^{Vp}$ is a solution of (7.13) if it satisfies $\mathbf{w}_v^* = \tilde{\mathbf{w}}^*$ for all v, or, equivalently, if we can write $\mathbf{w}^* = [\tilde{\mathbf{w}}^*; \ldots; \tilde{\mathbf{w}}^*]$. Regardless of interpretation, the Karush, Kuhn, Tucker (KKT) conditions of (7.13) dictate that that optimal argument $\mathbf{w}^*$ must sastisfy

$$\mathbf{w}^* \subset \operatorname{span}(\mathbf{1}_V \otimes \mathbf{I}_p), \qquad (\mathbf{1}_V \otimes \mathbf{I}_p)^T \nabla f(\mathbf{w}^*) = \mathbf{0}. \tag{7.14}$$

The first condition in (7.14) requires that all the local variables $\mathbf{w}_v^*$ be equal, while the second condition requires the sum of local gradients to vanish at the optimal point. This latter condition is not the same as $\nabla f(\mathbf{w}) = \mathbf{0}$. If we observe that the gradient $\nabla f(\mathbf{w}^t)$ of the aggregate function can be written as $\nabla f(\mathbf{w}) = [\nabla f_1(\mathbf{w}_1); \ldots; \nabla f_V(\mathbf{w}_V)] \in \mathbb{R}^{Vp}$, the condition $\nabla f(\mathbf{w}) = \mathbf{0}$ implies that all the local gradients are null, i.e., that $\nabla f_v(\mathbf{w}_v) = \mathbf{0}$ for all v. This is stronger than having their sum being null as required by (7.14).

Define now the extended weight matrices as the Kronecker products $\mathbf{Z} := \mathbf{W} \otimes \mathbf{I} \in \mathbb{R}^{Vp \times Vp}$ and $\tilde{\mathbf{Z}} := \tilde{\mathbf{W}} \otimes \mathbf{I} \in \mathbb{R}^{Vp \times Vp}$. Note that the required conditions for the weight matrices $\mathbf{W}$ and $\tilde{\mathbf{W}}$ in Assumption 14 enforce some conditions on the extended weight matrices $\mathbf{Z}$ and $\tilde{\mathbf{Z}}$. Based on Assumption 14(a), the matrices $\mathbf{Z}$ and $\tilde{\mathbf{Z}}$ are also symmetric, i.e., $\mathbf{Z} = \mathbf{Z}^T$ and $\tilde{\mathbf{Z}} = \tilde{\mathbf{Z}}^T$. Conditions in Assumption 14(b) imply that $\operatorname{null}\{\tilde{\mathbf{Z}} - \mathbf{Z}\} = \operatorname{span}\{\mathbf{1} \otimes \mathbf{I}\}$, $\operatorname{null}\{\mathbf{I} - \mathbf{Z}\} = \operatorname{span}\{\mathbf{1} \otimes \mathbf{I}\}$, and $\operatorname{null}\{\mathbf{I} - \tilde{\mathbf{Z}}\} \supseteq \operatorname{span}\{\mathbf{1} \otimes \mathbf{I}\}$. Lastly, the spectral properties of matrices $\mathbf{W}$ and $\tilde{\mathbf{W}}$ in Assumption 14(c) yield that matrix $\tilde{\mathbf{Z}}$ is positive definite and the expression $\mathbf{Z} \preceq \tilde{\mathbf{Z}} \preceq (\mathbf{I} + \mathbf{Z})/2$ holds.

According to the definition of the extended weight matrix $\mathbf{Z}$, the DGD iteration in (7.2) is equivalent to

$$\mathbf{w}^{t+1} = \mathbf{Z}\mathbf{w}^t - \alpha \nabla f(\mathbf{w}^t), \tag{7.15}$$

where, according to (7.13), the gradient $\nabla f(\mathbf{w}^t)$ of the aggregate function can be written

as $\nabla f(\mathbf{w}^t) = [\nabla f_1(\mathbf{w}_1^t); \ldots; \nabla f_V(\mathbf{w}_V^t)] \in \mathbb{R}^{Vp}$. Likewise, the EXTRA iteration in (7.3) can be written as

$$\mathbf{w}^{t+1} = (\mathbf{I} + \mathbf{Z})\mathbf{w}^t - \tilde{\mathbf{Z}}\mathbf{w}^{t-1} - \alpha \left[\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^{t-1})\right]. \tag{7.16}$$

The fundamental difference between DGD and EXTRA is that a fixed point of (7.15) does not necessarily satisfy (7.14), whereas the fixed points of (7.16) are guaranteed to do so. Indeed, taking limits in (7.15) we see that the fixed points $\mathbf{w}^\infty$ of DGD must satisfy

$$(\mathbf{I} - \mathbf{Z})\mathbf{w}^\infty + \alpha \nabla f(\mathbf{w}^\infty) = \mathbf{0}, \tag{7.17}$$

which is incompatible with (7.14) except in peculiar circumstances – such as, e.g., when all local functions have the same minimum. The limit points of EXTRA, however, satisfy the relationship

$$\mathbf{w}^\infty - \mathbf{w}^\infty = (\mathbf{Z} - \tilde{\mathbf{Z}})\mathbf{w}^\infty - \alpha[\nabla f(\mathbf{w}^\infty) - \nabla f(\mathbf{w}^\infty)]. \tag{7.18}$$

Canceling out the variables on the left hand side and the gradients in the right hand side it follows that $(\mathbf{Z} - \tilde{\mathbf{Z}})\mathbf{w}^\infty = \mathbf{0}$. Since the null space of of $\mathbf{Z} - \tilde{\mathbf{Z}}$ is $\text{null}(\mathbf{Z} - \tilde{\mathbf{Z}}) = \mathbf{1}_V \otimes \mathbf{I}_p$ by assumption, we must have $\mathbf{w}^\infty \subset \text{span}(\mathbf{1}_V \otimes \mathbf{I}_p)$. This is the first condition in (7.14). For the second condition in (7.14) sum the updates in (7.16) recursively and use the telescopic nature of the sum to write

$$\mathbf{w}^{t+1} = \tilde{\mathbf{Z}}\mathbf{w}^t - \alpha \nabla f(\mathbf{w}^t) - \sum_{s=0}^{t} (\tilde{\mathbf{Z}} - \mathbf{Z})\mathbf{w}^s. \tag{7.19}$$

Substituting the limit point in (7.19) and reordering terms, we see that $\mathbf{w}^\infty$ must satisfy

$$\alpha \nabla f(\mathbf{w}^\infty) = (\mathbf{I} - \tilde{\mathbf{Z}})\mathbf{w}^\infty - \sum_{s=0}^{\infty} (\tilde{\mathbf{Z}} - \mathbf{Z})\mathbf{w}^s. \tag{7.20}$$

In (7.20) we have that $(\mathbf{I} - \tilde{\mathbf{Z}})\mathbf{w}^\infty = \mathbf{0}$ because the null space of $(\mathbf{I} - \tilde{\mathbf{Z}})$ is $\text{null}(\mathbf{Z} - \tilde{\mathbf{Z}}) = \mathbf{1}_V \otimes \mathbf{I}_p$ by assumption and $\mathbf{w}^\infty \subset \text{span}(\mathbf{1}_V \otimes \mathbf{I}_p)$ as already shown. Implementing this simplification and considering the multiplication of the resulting equality by $(\mathbf{1}_V \otimes \mathbf{I}_p)^T$ we obtain

$$(\mathbf{1}_V \otimes \mathbf{I}_p)^T \alpha \nabla f(\mathbf{w}^\infty) = -\sum_{s=0}^{\infty} (\mathbf{1}_V \otimes \mathbf{I}_p)^T (\mathbf{Z} - \tilde{\mathbf{Z}})\mathbf{w}^s. \tag{7.21}$$

In (7.21), the terms $(\mathbf{1}_V \otimes \mathbf{I}_p)^T (\mathbf{Z} - \tilde{\mathbf{Z}}) = \mathbf{0}$ because the matrices $\mathbf{Z}$ and $\tilde{\mathbf{Z}}$ are symmetric and $(\mathbf{1}_V \otimes \mathbf{I}_p)$ is in the null space of the difference $\mathbf{Z} - \tilde{\mathbf{Z}}$. This implies that $(\mathbf{1}_V \otimes \mathbf{I}_p)^T \alpha \nabla f(\mathbf{w}^\infty) = \mathbf{0}$, which is the second condition in (7.14). Therefore, given the assumption that the sequence

of EXTRA iterates $\mathbf{w}^t$ has a limit point $\mathbf{w}^\infty$ it follows that this limit point satisfies both conditions in (7.14) and for this reason exact convergence with constant stepsize is achievable for EXTRA.

7.2.2 Stochastic saddle point method interpretation of DSA

The convergence proofs of DSA build on a reinterpretation of EXTRA as a saddle point method. To introduce this primal-dual interpretation consider the update in (7.19) and define the sequence of vectors $\mathbf{v}^t = \sum_{s=0}^{t}(\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{w}^s$. The vector $\mathbf{v}^t$ represents the accumulation of variable dissimilarities in different nodes over time. Considering this definition of $\mathbf{v}^t$ we can rewrite (7.19) as

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha\left[\nabla f(\mathbf{w}^t) + \frac{1}{\alpha}(\mathbf{I}-\tilde{\mathbf{Z}})\mathbf{w}^t + \frac{1}{\alpha}(\tilde{\mathbf{Z}}-\mathbf{Z})^{1/2}\mathbf{v}^t\right]. \tag{7.22}$$

Furthermore, based on the definition of the sequence $\mathbf{v}^t = \sum_{s=0}^{t}(\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{w}^s$ we can write the recursive expression

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \alpha\left[\frac{1}{\alpha}(\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{w}^{t+1}\right]. \tag{7.23}$$

Consider $\mathbf{w}$ as a primal variable and $\mathbf{v}$ as a dual variable. Then, the updates in (7.22) and (7.23) are equivalent to the updates of a saddle point method with stepsize α that solves for the critical points of the augmented Lagrangian

$$\mathcal{L}(\mathbf{w}, \mathbf{v}) = f(\mathbf{w}) + \frac{1}{\alpha}\mathbf{v}^T(\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{w} + \frac{1}{2\alpha}\mathbf{w}^T(\mathbf{I} - \tilde{\mathbf{Z}})\mathbf{w}. \tag{7.24}$$

In the Lagrangian in (7.24) the factor $(1/\alpha)\mathbf{v}^T(\tilde{\mathbf{Z}}-\mathbf{Z})^{1/2}\mathbf{w}$ stems from the linear constraint $(\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{w} = \mathbf{0}$ and the quadratic term $(1/2\alpha)\mathbf{w}^T(\mathbf{I} - \tilde{\mathbf{Z}})\mathbf{w}$ is the augmented term added to the Lagrangian. Therefore, the optimization problem whose augmented Lagrangian is the one given in (7.24) is

$$\mathbf{w}^* = \underset{\mathbf{w}}{\text{argmin}}\ \ f(\mathbf{w}) \qquad \text{s.t. } \frac{1}{\alpha}(\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{w} = \mathbf{0}. \tag{7.25}$$

Observing that the null space of $(\tilde{\mathbf{Z}}-\mathbf{Z})^{1/2}$ is $\text{null}((\tilde{\mathbf{Z}}-\mathbf{Z})^{1/2}) = \text{null}(\tilde{\mathbf{Z}} - \mathbf{Z}) = \text{span}\{\mathbf{1}_V \otimes \mathbf{I}_p\}$, the constraint in (7.25) is equivalent to the consensus constraint $\mathbf{w}_v = \mathbf{w}_m$ for all n, m that appears in (7.13). This means that (7.25) is equivalent to (7.13), which, as already argued, is equivalent to the original problem in (7.1). Hence, EXTRA is a saddle point method that solves (7.25) which, because of their equivalence, is tantamount to solving (7.1). Considering that saddle point methods converge linearly, it follows that the same is true of EXTRA.

That EXTRA is a saddle point method provides a simple explanation of its convergence properties. For the purposes of this chapter, however, the important fact is that if EXTRA is a saddle point method, DSA is a stochastic saddle point method. To write DSA in this form define $\hat{\mathbf{g}}^t := [\hat{\mathbf{g}}_1^t; \ldots; \hat{\mathbf{g}}_V^t] \in \mathbb{R}^{Vp}$ as the vector that concatenates all the local stochastic averaging gradients at step t. Then, the DSA update in (7.8) can be written as

$$\mathbf{w}^{t+1} = (\mathbf{I} + \mathbf{Z})\mathbf{w}^t - \tilde{\mathbf{Z}}\mathbf{w}^{t-1} - \alpha \left[\hat{\mathbf{g}}^t - \hat{\mathbf{g}}^{t-1}\right]. \tag{7.26}$$

Comparing (7.16) and (7.26) we see that they differ in the latter using stochastic averaging gradients $\hat{\mathbf{g}}^t$ in lieu of the full gradients $\nabla f(\mathbf{w}^t)$. Therefore, DSA is a stochastic saddle point method in which the primal variables are updated as

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha\hat{\mathbf{g}}^t - (\mathbf{I} - \tilde{\mathbf{Z}})\mathbf{w}^t - (\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{v}^t, \tag{7.27}$$

and the dual variables $\mathbf{v}^t$ are updated as

$$\mathbf{v}^{t+1} = \mathbf{v}^t + (\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{w}^{t+1}. \tag{7.28}$$

Notice that the initial primal variable $\mathbf{w}^0$ is an arbitrary vector in $\mathbb{R}^{Vp}$, while according to the definition $\mathbf{v}^t = \sum_{s=0}^{t}(\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{w}^s$. We then need to set the initial multiplier to $\mathbf{v}^0 = (\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}\mathbf{w}^0$. This is not a problem in practice because (7.27) and (7.28) are not used for implementation. In our converge analysis we utilize the (equivalent) stochastic saddle point expressions for DSA shown in (7.27) and (7.28). The expression in (7.8) is used for implementation because it avoids exchanging dual variables – as well as the initialization problem. The convergence analysis is presented in the following section.

7.3 Convergence analysis

Our goal here is to show that as time progresses the sequence of iterates $\mathbf{w}^t$ approaches the optimal argument $\mathbf{w}^*$. To do so, in addition to the conditions on the weight matrices $\mathbf{W}$ and $\tilde{\mathbf{W}}$ in Assumption 14, we assume the instantaneous local functions $f_{v,i}$ have specific properties that we state next.

Assumption 15 The instantaneous local functions $f_{v,i}(\mathbf{w}_v)$ are differentiable and strongly convex with parameter μ.

Assumption 16 The gradient of instantaneous local functions $\nabla f_{v,i}$ are Lipschitz continuous with parameter L, i.e., for all $v \in \{1, \ldots, V\}$ and $i \in \{1, \ldots, q_v\}$ we can write

$$\|\nabla f_{v,i}(\mathbf{a}) - \nabla f_{v,i}(\mathbf{b})\| \leq L \|\mathbf{a} - \mathbf{b}\| \quad \mathbf{a}, \mathbf{b} \in \mathbb{R}^p. \tag{7.29}$$

The condition imposed by Assumption 15 implies that the local functions $f_v(\mathbf{w}_v)$ and the global cost function $f(\mathbf{w}) = \sum_{v=1}^{V} f_v(\mathbf{w}_v)$ are also strongly convex with parameter μ. Likewise, Lipschitz continuity of the local instantaneous gradients considered in Assumption 16 enforces Lipschitz continuity of the local functions gradient $\nabla f_v(\mathbf{w}_v)$ and the aggregate function gradient $\nabla f(\mathbf{w})$ – see, e.g., Lemma 1 in [64].

7.3.1 Preliminaries

In this section we study some basic properties of the sequences of primal and dual variables generated by the DSA algorithm. In the following lemma, we study the relation of the iterates $\mathbf{w}^t$ and $\mathbf{v}^t$ with the optimal primal $\mathbf{w}^*$ and dual $\mathbf{v}^*$ arguments.

Lemma 21 *Consider the DSA algorithm as defined in* (7.6)-(7.9) *and recall the updates of the primal* $\mathbf{w}^t$ *and dual* $\mathbf{v}^t$ *variables in* (7.27) *and* (7.28)*, respectively. Further, define the positive semidefinite matrix* $\mathbf{U} := (\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}$*. If Assumption 14 holds true, then the sequence of primal* $\mathbf{w}^t$ *and dual* $\mathbf{v}^t$ *variables satisfy*

$$\alpha \left[\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\right] = (\mathbf{I} + \mathbf{Z} - 2\tilde{\mathbf{Z}})(\mathbf{w}^* - \mathbf{w}^{t+1}) + \tilde{\mathbf{Z}}(\mathbf{w}^t - \mathbf{w}^{t+1}) - \mathbf{U}(\mathbf{v}^{t+1} - \mathbf{v}^*). \quad (7.30)$$

Proof: Considering the update rule for the dual variable in (7.28) and the definition $\mathbf{U} = (\tilde{\mathbf{Z}} - \mathbf{Z})^{1/2}$, we can substitute $\mathbf{U}\mathbf{v}^t$ in (7.27) by $\mathbf{U}\mathbf{v}^{t+1} - \mathbf{U}^2\mathbf{w}^{t+1}$. Applying this substitution into the DSA primal update in (7.27) yields

$$\alpha\hat{\mathbf{g}}^t = -(\mathbf{I} + \mathbf{Z} - \tilde{\mathbf{Z}})\mathbf{w}^{t+1} + \tilde{\mathbf{Z}}\mathbf{w}^t - \mathbf{U}\mathbf{v}^{t+1}. \quad (7.31)$$

By adding and subtracting $\tilde{\mathbf{Z}}\mathbf{w}^{t+1}$ to the right hand side of (7.31) and considering the fact that $(\mathbf{I} + \mathbf{Z} - 2\tilde{\mathbf{Z}})\mathbf{w}^* = \mathbf{0}$ we obtain

$$\alpha\hat{\mathbf{g}}^t = (\mathbf{I} + \mathbf{Z} - 2\tilde{\mathbf{Z}})(\mathbf{w}^* - \mathbf{w}^{t+1}) + \tilde{\mathbf{Z}}(\mathbf{w}^t - \mathbf{w}^{t+1}) - \mathbf{U}\mathbf{v}^{t+1}. \quad (7.32)$$

One of the KKT conditions of problem (7.25) implies that the optimal variables $\mathbf{w}^*$ and $\mathbf{v}^*$ satisfy $\alpha\nabla f(\mathbf{w}^*) + \mathbf{U}\mathbf{v}^* = \mathbf{0}$ or equivalently $-\alpha\nabla f(\mathbf{w}^*) = \mathbf{U}\mathbf{v}^*$. Adding this equality to both sides of (7.32) follows the claim in (7.30). ■

In the subsequent analyses of convergence of DSA, we need an upper bound for the expected value of squared difference between the stochastic averaging gradient $\hat{\mathbf{g}}^t$ and the optimal argument gradient $\nabla f(\mathbf{w}^*)$ given the observations until step t which is denoted by $\mathbb{E}\left[\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\|^2 \mid \mathcal{F}^t\right]$. To establish this upper bound first we define the sequence $p^t \in \mathbb{R}$

as

$$p^t := \sum_{v=1}^{V} \left[\frac{1}{q_v} \sum_{i=1}^{q_v} \left(f_{v,i}(\mathbf{y}_{v,i}^t) - f_{v,i}(\tilde{\mathbf{w}}^*) - \nabla f_{v,i}(\tilde{\mathbf{w}}^*)^T (\mathbf{y}_{v,i}^t - \tilde{\mathbf{w}}^*) \right) \right]. \tag{7.33}$$

Notice that based on the strong convexity of the local instantaneous functions $f_{v,i}$, each term $f_{v,i}(\mathbf{y}_{v,i}^t) - f_{v,i}(\tilde{\mathbf{w}}^*) - \nabla f_{v,i}(\tilde{\mathbf{w}}^*)^T(\mathbf{y}_{v,i}^t - \tilde{\mathbf{w}}^*)$ is positive and as a result the sequence p^t defined in (7.33) is always positive. In the following lemma, we use the result in Lemma 21 to guarantee an upper bound for the expectation $\mathbb{E}\left[\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\|^2 \mid \mathcal{F}^t\right]$ in terms of p^t and the optimality gap $f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*)$.

Lemma 22 *Consider the DSA algorithm in* (7.6)-(7.9) *and the definition of the sequence* p^t *in* (7.33)*. If Assumptions 14-16 hold true, then the squared norm of the difference between the stochastic averaging gradient* $\hat{\mathbf{g}}^t$ *and the optimal gradient* $\nabla f(\mathbf{w}^*)$ *in expectation is bounded above by*

$$\mathbb{E}\left[\left\| \hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*) \right\|^2 | \mathcal{F}^t \right] \leq 4Lp^t + 2\left(2L - \mu\right)\left(f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*) \right). \tag{7.34}$$

Proof: According to the definition of $\hat{\mathbf{g}}^t$ which is the concatenation of the local stochastic averaging gradients $\hat{\mathbf{g}}_v^t$ and the fact that the expected value of sum is equal to the sum of expected values, we can write the expected value $\mathbb{E}\left[\left\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\right\|^2 \mid \mathcal{F}^t\right]$ as

$$\mathbb{E}\left[\left\| \hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*) \right\|^2 \mid \mathcal{F}^t \right] = \sum_{v=1}^{V} \mathbb{E}\left[\left\| \hat{\mathbf{g}}_v^t - \nabla f_v(\tilde{\mathbf{w}}^*) \right\|^2 \mid \mathcal{F}^t \right]. \tag{7.35}$$

We proceed by finding upper bounds for the summands of (7.35). Observe that using the standard variance decomposition for any random variable vector $\mathbf{a}$ we can write $\mathbb{E}\left[\|\mathbf{a}\|^2\right] = \|\mathbb{E}\left[\mathbf{a}\right]\|^2 + \mathbb{E}\left[\|\mathbf{a} - \mathbb{E}\left[\mathbf{a}\right]\|^2\right]$. Notice that the same relation holds true when the expectations are computed with respect to a specific field $\mathcal{F}$. By setting $\mathbf{a} = \hat{\mathbf{g}}_v^t - \nabla f_v(\tilde{\mathbf{w}}^*)$ and considering that $\mathbb{E}\left[\mathbf{a} \mid \mathcal{F}^t\right] = \nabla f_v(\mathbf{w}_v^t) - \nabla f_v(\tilde{\mathbf{w}}^*)$, the variance decomposition implies

$$\begin{aligned} \mathbb{E}\left[\left\| \hat{\mathbf{g}}_v^t - \nabla f_v(\tilde{\mathbf{w}}^*) \right\|^2 \mid \mathcal{F}^t \right] &= \left\| \nabla f_v(\mathbf{w}_v^t) - \nabla f_v(\tilde{\mathbf{w}}^*) \right\|^2 \\ &\quad + \mathbb{E}\left[\left\| \hat{\mathbf{g}}_v^t - \nabla f_v(\tilde{\mathbf{w}}^*) - \nabla f_v(\mathbf{w}_v^t) + \nabla f_v(\tilde{\mathbf{w}}^*) \right\|^2 \mid \mathcal{F}^t \right]. \end{aligned} \tag{7.36}$$

The next step is to find an upper bound for the last term in (7.36). Adding and subtracting $\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)$ and using the inequality $\|\mathbf{a} + \mathbf{b}\|^2 \leq 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$ for the choice of variables $\mathbf{a} = \nabla f_{v,i_v^t}(\mathbf{w}_v^t) - \nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*) - \nabla f_v(\mathbf{w}_v^t) + \nabla f_v(\tilde{\mathbf{w}}^*)$ and $\mathbf{b} = -(\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t) - \nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*) -$

$(1/q_v)\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{y}_{v,i}^t)+\nabla f_v(\tilde{\mathbf{w}}^*))$ lead to

$$\begin{aligned}\mathbb{E}\left[\left\|\hat{\mathbf{g}}_v^t-\nabla f_v(\tilde{\mathbf{w}}^*)-\nabla f_v(\mathbf{w}_v^t)+\nabla f_v(\tilde{\mathbf{w}}^*)\right\|^2\mid\mathcal{F}^t\right]\\ \leq 2\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{w}_v^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)-\nabla f_v(\mathbf{w}_v^t)+\nabla f_v(\tilde{\mathbf{w}}^*)\right\|^2\mid\mathcal{F}^t\right]\\ +2\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)-\frac{1}{q_v}\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{y}_{v,i}^t)+\nabla f_v(\tilde{\mathbf{w}}^*)\right\|^2\mid\mathcal{F}^t\right].\end{aligned}\tag{7.37}$$

In this step we use the standard variance decomposition twice to simplify the two expectations in the right hand side of (7.37). Based on the standard variance decomposition $\mathbb{E}\left[\|\mathbf{a}-\mathbb{E}\left[\mathbf{a}\right]\|^2\right]=\mathbb{E}\left[\|\mathbf{a}\|^2\right]-\|\mathbb{E}\left[\mathbf{a}\right]\|^2$ we obtain $\mathbb{E}\left[\|\mathbf{a}-\mathbb{E}\left[\mathbf{a}\right]\|^2\right]\leq\mathbb{E}\left[\|\mathbf{a}\|^2\right]$. Therefore, by setting $\mathbf{y}=\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)$ and observing that the expected value $\mathbb{E}\left[\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\mid\mathcal{F}^t\right]$ is equal to $(1/q_v)\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{y}_{v,i}^t)-\nabla f_v(\tilde{\mathbf{w}}^*)$ we obtain that

$$\begin{aligned}\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)-\frac{1}{q_v}\sum_{i=1}^{q_v}\nabla f_{v,i}(\mathbf{y}_{v,i}^t)+\nabla f_v(\tilde{\mathbf{w}}^*)\right\|^2\mid\mathcal{F}^t\right]\\ \leq\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\right\|^2\mid\mathcal{F}^t\right].\end{aligned}\tag{7.38}$$

Moreover, by choosing $\mathbf{a}=\nabla f_{v,i_v^t}(\mathbf{w}_v^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)$ and noticing the relation for the expected value which is $\mathbb{E}\left[\nabla f_{v,i_v^t}(\mathbf{w}_v^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\mid\mathcal{F}^t\right]=\nabla f_v(\mathbf{w}_v^t)-\nabla f_v(\tilde{\mathbf{w}}^*)$, the equality $\mathbb{E}\left[\|\mathbf{a}-\mathbb{E}\left[\mathbf{a}\right]\|^2\right]=\mathbb{E}\left[\|\mathbf{a}\|^2\right]-\|\mathbb{E}\left[\mathbf{a}\right]\|^2$ yields

$$\begin{aligned}\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{w}_v^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)-\nabla f_v(\mathbf{w}_v^t)+\nabla f_v(\tilde{\mathbf{w}}^*)\right\|^2\mid\mathcal{F}^t\right]\\ =\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{w}_v^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\right\|^2\mid\mathcal{F}^t\right]-\left\|\nabla f_v(\mathbf{w}_v^t)-\nabla f_v(\tilde{\mathbf{w}}^*)\right\|^2.\end{aligned}\tag{7.39}$$

By substituting the upper bound in (7.38) and the simplification in (7.39) into (7.37), and considering the expression in (7.36) we obtain that

$$\begin{aligned}\mathbb{E}\left[\left\|\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*)\right\|^2\mid\mathcal{F}^t\right]\leq 2\sum_{v=1}^{V}\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\right\|^2\mid\mathcal{F}^t\right]\\ -\sum_{v=1}^{V}\left\|\nabla f_v(\mathbf{w}_v^t)-\nabla f_v(\tilde{\mathbf{w}}^*)\right\|^2+2\sum_{v=1}^{V}\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{w}_v^t)-\nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\right\|^2\mid\mathcal{F}^t\right].\end{aligned}\tag{7.40}$$

We proceed by finding an upper bound for the first sum in the right hand side of (7.40). Notice that if the gradients of the function g are Lipschitz continuous with parameter L, then for any two vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ we can write $g(\mathbf{a}_1)\geq g(\mathbf{a}_2)+\nabla g(\mathbf{a}_2)^T(\mathbf{a}_1-\mathbf{a}_2)+(1/2L)\|\nabla g(\mathbf{a}_1)-\nabla g(\mathbf{a}_2)\|^2$. According to the Lipschitz continuity of the instantaneous

local functions gradient $\nabla f_{v,i}(\mathbf{w}_v)$, we can write the inequality for $g = f_{v,i}$, $\mathbf{a}_1 = \mathbf{y}_{v,i}^t$ and $\mathbf{a}_2 = \tilde{\mathbf{w}}^*$ which is equivalent to

$$\frac{1}{2L}\left\|\nabla f_{v,i}(\mathbf{y}_{v,i}^t) - \nabla f_{v,i}(\tilde{\mathbf{w}}^*)\right\|^2 \leq f_{v,i}(\mathbf{y}_{v,i}^t) - f_{v,i}(\tilde{\mathbf{w}}^*) - \nabla f_{v,i}(\tilde{\mathbf{w}}^*)^T(\mathbf{y}_{v,i}^t - \tilde{\mathbf{w}}^*). \quad (7.41)$$

Summing up both sides of (7.41) for all $i = 1, \ldots, q_v$ and dividing both sides of the implied inequality by q_v yield

$$\begin{aligned}\frac{1}{q_v}\sum_{i=1}^{q_v}&\left\|\nabla f_{v,i}(\mathbf{y}_{v,i}^t) - \nabla f_{v,i}(\tilde{\mathbf{w}}^*)\right\|^2 \\ &\leq 2L\left[\frac{1}{q_v}\sum_{i=1}^{q_v} f_{v,i}(\mathbf{y}_{v,i}^t) - f_{v,i}(\tilde{\mathbf{w}}^*) - \nabla f_{v,i}(\tilde{\mathbf{w}}^*)^T(\mathbf{y}_{v,i}^t - \tilde{\mathbf{w}}^*)\right]. \quad (7.42)\end{aligned}$$

Since the random functions f_{v,i_v^t} has a uniform distribution over the set $\{f_{v,1}, \ldots, f_{v,q_v}\}$, we can substitute the left hand side of (7.42) by $\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{y}_{v,i_v^t}^t) - \nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\right\|^2 \mid \mathcal{F}^t\right]$. Apply this substitution and sum up both sides of (7.42) for $v = 1, \ldots, V$. According to the definition of sequence p^t in (7.33), if we sum up the right hand side of (7.42) over v it can be simplified as $2Lp^t$. Applying these simplifications we obtain

$$\sum_{v=1}^{V}\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{y}_v^t) - \nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\right\|^2 \mid \mathcal{F}^t\right] \leq 2Lp^t. \quad (7.43)$$

Replacing the upper bound in (7.43) into (7.40) and simplifying $\sum_{v=1}^{V}\left\|\nabla f_v(\mathbf{w}_v^t) - \nabla f_v(\tilde{\mathbf{w}}^*)\right\|^2$ to $\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2$ yield

$$\begin{aligned}\mathbb{E}\left[\left\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\right\|^2 \mid \mathcal{F}^t\right] \leq 2\sum_{v=1}^{V}&\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{w}_v^t) - \nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\right\|^2 \mid \mathcal{F}^t\right] \\ &- \left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 + 4Lp^t. \quad (7.44)\end{aligned}$$

To show that the sum in the right hand side of (7.44) is bounded above we use the Lipschitz continuity of the instantaneous functions gradients $\nabla f_{v,i}$. Using the same argument from (7.41) to (7.43) we can write

$$\begin{aligned}\sum_{v=1}^{V}&\mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{w}_v^t) - \nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\right\|^2 \mid \mathcal{F}^t\right] \quad (7.45) \\ &\leq 2L\sum_{v=1}^{V}\frac{1}{q_v}\left[\sum_{i=1}^{q_v} f_{v,i}(\mathbf{w}_v^t) - f_{v,i}(\tilde{\mathbf{w}}^*) - \nabla f_{v,i}(\tilde{\mathbf{w}}^*)^T(\mathbf{w}_v^t - \tilde{\mathbf{w}}^*)\right].\end{aligned}$$

Considering the definition of the local objective functions $f_v(\mathbf{w}_v) = (1/q_v)\sum_{i=1}^{q_v} f_{v,i}(\mathbf{w}_v)$ and the aggregate function $f(\mathbf{w}) := \sum_{v=1}^{V} f_v(\mathbf{w}_v)$, the right hand side of (7.45) can be simplified as

$$\sum_{v=1}^{V} \mathbb{E}\left[\left\|\nabla f_{v,i_v^t}(\mathbf{w}_v^t) - \nabla f_{v,i_v^t}(\tilde{\mathbf{w}}^*)\right\|^2 \mid \mathcal{F}^t\right] \leq 2L\left(f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*)\right). \tag{7.46}$$

Replacing the sum in (7.44) by the upper bound in (7.46) implies

$$\begin{aligned}\mathbb{E}\left[\left\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\right\|^2 \mid \mathcal{F}^t\right] \leq 4Lp^t - \left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 \\ + 4L\left(f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*)\right).\end{aligned} \tag{7.47}$$

Considering the strong convexity of the global objective function f with constant μ we can write

$$\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 \geq 2\mu\left(f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*)\right). \tag{7.48}$$

Therefore, we can substitute $\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\|^2$ in (7.46) by the lower bound in (7.48) and the claim in (7.34) follows. ■

Observe that as the sequence of iterates $\mathbf{w}^t$ approaches the optimal argument $\mathbf{w}^*$, all the local auxiliary variables $\mathbf{y}_{v,i}^t$ converge to $\tilde{\mathbf{w}}^*$ which follows convergence of p^t to null. This observation in association with the result in (7.34) implies that the expected value of the difference between the stochastic averaging gradient $\hat{\mathbf{g}}^t$ and the optimal gradient $\nabla f(\mathbf{w}^*)$ vanishes as the sequence of iterates $\mathbf{w}^t$ approaches the optimal argument $\mathbf{w}^*$.

7.3.2 Convergence

In this section we establish linear convergence of the sequence of iterates $\mathbf{w}^t$ generated by DSA to the optimal argument $\mathbf{w}^*$. To do so, define $0 < \gamma$ and $\Gamma < \infty$ as the smallest and largest eigenvalues of the positive definite matrix $\tilde{\mathbf{Z}}$, respectively. Likewise, define γ' as the smallest non-zero eigenvalue of the matrix $\tilde{\mathbf{Z}} - \mathbf{Z}$ and Γ' as the largest eigenvalue of the matrix $\tilde{\mathbf{Z}} - \mathbf{Z}$. Further, define the vectors $\mathbf{u}^t, \mathbf{u}^* \in \mathbb{R}^{2Vp}$ and matrix $\mathbf{G} \in \mathbb{R}^{2Vp \times 2Vp}$ as

$$\mathbf{u}^* := \begin{bmatrix}\mathbf{w}^* \\ \mathbf{v}^*\end{bmatrix}, \quad \mathbf{u}^t := \begin{bmatrix}\mathbf{w}^t \\ \mathbf{v}^t\end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix}\tilde{\mathbf{Z}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}\end{bmatrix}. \tag{7.49}$$

Observe that the vector $\mathbf{u}^* \in \mathbb{R}^{2Vp}$ concatenates the optimal primal and dual variables and the vector $\mathbf{u}^t \in \mathbb{R}^{2Vp}$ contains primal and dual iterates at step t. Further, $\mathbf{G} \in \mathbb{R}^{2Vp \times 2Vp}$ is a

block diagonal positive definite matrix that we introduce since instead of tracking the value of ℓ_2 norm $\|\mathbf{u}^t-\mathbf{u}^*\|_2^2$ we study the convergence properties of $\mathbf{G}$ weighted norm $\|\mathbf{u}^t-\mathbf{u}^*\|_{\mathbf{G}}^2$. Notice that the weighted norm $\|\mathbf{u}^t-\mathbf{u}^*\|_{\mathbf{G}}^2$ is equivalent to $(\mathbf{u}^t-\mathbf{u}^*)^T\mathbf{G}(\mathbf{u}^t-\mathbf{u}^*)$. Our goal is to show that the sequence $\|\mathbf{u}^t-\mathbf{u}^*\|_{\mathbf{G}}^2$ converges linearly to null. To do this we show linear convergence of a Lyapunov function of the sequence $\|\mathbf{u}^t-\mathbf{u}^*\|_{\mathbf{G}}^2$. The Lyapunov function is defined as $\|\mathbf{u}^t-\mathbf{u}^*\|_{\mathbf{G}}^2+cp^t$ where $c>0$ is a positive constant.

To prove linear convergence of the sequence $\|\mathbf{u}^t-\mathbf{u}^*\|_{\mathbf{G}}^2+cp^t$ we first show an upper bound for the expected error $\mathbb{E}\left[\|\mathbf{u}^{t+1}-\mathbf{u}^*\|_{\mathbf{G}}^2 \mid \mathcal{F}^t\right]$ in terms of $\|\mathbf{u}^t-\mathbf{u}^*\|_{\mathbf{G}}^2$ and some parameters that capture optimality gap.

Lemma 23 *Consider the DSA algorithm as defined in* (7.6)-(7.9). *Further recall the definitions of* p^t *in* (7.33) *and* $\mathbf{u}^t$, $\mathbf{u}^*$, *and* $\mathbf{G}$ *in* (7.49). *If Assumptions 14-16 hold true, then for any positive constant* $\eta>0$ *we can write*

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{u}^{t+1}-\mathbf{u}^*\|_{\mathbf{G}}^2 \mid \mathcal{F}^t\right] \leq \|\mathbf{u}^t-\mathbf{u}^*\|_{\mathbf{G}}^2 &- 2\mathbb{E}\left[\|\mathbf{w}^{t+1}-\mathbf{w}^*\|_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}}^2 \mid \mathcal{F}^t\right]+\frac{\alpha 4L}{\eta}p^t\\ &-\mathbb{E}\left[\|\mathbf{w}^{t+1}-\mathbf{w}^t\|_{\tilde{\mathbf{Z}}-2\alpha\eta\mathbf{I}}^2 \mid \mathcal{F}^t\right]-\mathbb{E}\left[\|\mathbf{v}^{t+1}-\mathbf{v}^t\|^2 \mid \mathcal{F}^t\right]\\ &-\left(\frac{4\alpha\mu}{L}-\frac{2\alpha(2L-\mu)}{\eta}\right)\left(f(\mathbf{w}^t)-f(\mathbf{w}^*)-\nabla f(\mathbf{w}^*)^T(\mathbf{w}^t-\mathbf{w}^*)\right).\end{aligned} \tag{7.50}$$

Proof: According to the Lipschitz continuity of the aggregate function gradients $\nabla f(\mathbf{w})$, we can write $(1/L)\|\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*)\|^2 \leq (\mathbf{w}^t-\mathbf{w}^*)^T(\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*))$. By adding and subtracting $\mathbf{w}^{t+1}$ to the term $\mathbf{w}^t-\mathbf{w}^*$ and multiplying both sides of the inequality by 2α we obtain

$$\begin{aligned}\frac{2\alpha}{L}\left\|\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*)\right\|^2 \leq\ & 2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*))\\ &+2\alpha(\mathbf{w}^t-\mathbf{w}^{t+1})^T(\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*)).\end{aligned} \tag{7.51}$$

Expanding the difference $\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*)$ as $\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*)+\nabla f(\mathbf{w}^t)-\hat{\mathbf{g}}^t$ for the first inner product in the right hand side of (7.51) implies

$$\begin{aligned}\frac{2\alpha}{L}\left\|\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*)\right\|^2 \leq\ & 2\alpha(\mathbf{w}^t-\mathbf{w}^{t+1})^T(\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*))\\ &+2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*))+2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\nabla f(\mathbf{w}^t)-\hat{\mathbf{g}}^t).\end{aligned} \tag{7.52}$$

We proceed to simplify the inner product $2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*))$ in the right hand side of (7.52) by substituting $\alpha(\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*))$ with its equivalent as introduced in (7.30). By applying this substitution the inner product $2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*))$ can be simplified

as

$$2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*)) = -2\|\mathbf{w}^{t+1}-\mathbf{w}^*\|^2_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}} + 2(\mathbf{w}^{t+1}-\mathbf{w}^*)^T\tilde{\mathbf{Z}}(\mathbf{w}^t-\mathbf{w}^{t+1}) - 2(\mathbf{w}^{t+1}-\mathbf{w}^*)^T\mathbf{U}(\mathbf{v}^{t+1}-\mathbf{v}^*). \tag{7.53}$$

Based on the KKT condition of problem (7.25), the optimal primal variable satisfies $(\tilde{\mathbf{Z}}-\mathbf{Z})^{1/2}\mathbf{w}^* = \mathbf{0}$ which by considering the definition of the matrix $\mathbf{U} = (\tilde{\mathbf{Z}}-\mathbf{Z})^{1/2}$ we obtain that $\mathbf{U}\mathbf{w}^* = \mathbf{0}$. This observation in conjunction with the update rule of the dual variable $\mathbf{v}^t$ in (7.28) implies that we can substitute $\mathbf{U}(\mathbf{w}^{t+1}-\mathbf{w}^*)$ by $\mathbf{v}^{t+1}-\mathbf{v}^t$. Making this substitution into the last summand of the right hand side of (7.53) and considering the symmetry of the matrix $\mathbf{U}$ yield

$$2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*)) = -2\|\mathbf{w}^{t+1}-\mathbf{w}^*\|^2_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}} + 2(\mathbf{w}^{t+1}-\mathbf{w}^*)^T\tilde{\mathbf{Z}}(\mathbf{w}^t-\mathbf{w}^{t+1}) - 2(\mathbf{v}^{t+1}-\mathbf{v}^t)^T(\mathbf{v}^{t+1}-\mathbf{v}^*). \tag{7.54}$$

According to the definition of the vector $\mathbf{u}$ and matrix $\mathbf{G}$ in (7.49), the last two summands of (7.54) can be simplified as $2(\mathbf{u}^{t+1}-\mathbf{u}^t)^T\mathbf{G}(\mathbf{u}^*-\mathbf{u}^{t+1})$. Moreover, observe that the inner product $2(\mathbf{u}^{t+1}-\mathbf{u}^t)^T\mathbf{G}(\mathbf{u}^*-\mathbf{u}^{t+1})$ can be simplified as $\|\mathbf{u}^t-\mathbf{u}^*\|^2_{\mathbf{G}} - \|\mathbf{u}^{t+1}-\mathbf{u}^*\|^2_{\mathbf{G}} - \|\mathbf{u}^{t+1}-\mathbf{u}^t\|^2_{\mathbf{G}}$. Applying this simplification into (7.54) implies

$$2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*)) = -2\|\mathbf{w}^{t+1}-\mathbf{w}^*\|^2_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}} + \|\mathbf{u}^t-\mathbf{u}^*\|^2_{\mathbf{G}} - \|\mathbf{u}^{t+1}-\mathbf{u}^*\|^2_{\mathbf{G}} - \|\mathbf{u}^{t+1}-\mathbf{u}^t\|^2_{\mathbf{G}}. \tag{7.55}$$

The next step is to find an upper bound for the inner product $2\alpha(\mathbf{w}^t-\mathbf{w}^{t+1})^T(\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*))$. Note that for any two vectors $\mathbf{a}$ and $\mathbf{b}$, and any positive scalar η the inequality $2\mathbf{a}^T\mathbf{b} \leq \eta\|\mathbf{a}\|^2 + \eta^{-1}\|\mathbf{b}\|^2$ holds. Thus, by setting $\mathbf{a} = \mathbf{w}^t - \mathbf{w}^{t+1}$ and $\mathbf{b} = \nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)$ we obtain that

$$2\alpha(\mathbf{w}^t-\mathbf{w}^{t+1})^T(\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*)) \leq \frac{\alpha}{\eta}\|\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*)\|^2 + \alpha\eta\|\mathbf{w}^t-\mathbf{w}^{t+1}\|^2. \tag{7.56}$$

Now we substitute the terms in the right hand side of (7.52) by their simplifications or upper bounds. Replacing the inner product $2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\hat{\mathbf{g}}^t-\nabla f(\mathbf{w}^*))$ by the simplification in (7.55), substituting expression $2\alpha(\mathbf{w}^t-\mathbf{w}^{t+1})^T(\nabla f(\mathbf{w}^t)-\nabla f(\mathbf{w}^*))$ by the upper bound in (7.56), and substituting inner product $2\alpha(\mathbf{w}^{t+1}-\mathbf{w}^*)^T(\nabla f(\mathbf{w}^t)-\hat{\mathbf{g}}^t)$ by the sum $2\alpha(\mathbf{w}^t -$

$\mathbf{w}^*)^T(\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t) + 2\alpha(\mathbf{w}^{t+1} - \mathbf{w}^t)^T(\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t)$ imply

$$\frac{2\alpha}{L}\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 \leq -2\|\mathbf{w}^{t+1} - \mathbf{w}^*\|^2_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}} + \|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} - \|\mathbf{u}^{t+1} - \mathbf{u}^*\|^2_{\mathbf{G}}$$
$$- \|\mathbf{u}^{t+1} - \mathbf{u}^t\|^2_{\mathbf{G}} + \alpha\eta\|\mathbf{w}^t - \mathbf{w}^{t+1}\|^2 + \frac{\alpha}{\eta}\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\|^2$$
$$+ 2\alpha(\mathbf{w}^t - \mathbf{w}^*)^T(\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t) + 2\alpha(\mathbf{w}^{t+1} - \mathbf{w}^t)^T(\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t). \tag{7.57}$$

Considering that $\mathbf{w}^t - \mathbf{w}^*$ is deterministic given observations until step t and observing the relation $\mathbb{E}\left[\hat{\mathbf{g}}^t \mid \mathcal{F}^t\right] = \nabla f(\mathbf{w}^t)$, we obtain that $\mathbb{E}\left[(\mathbf{w}^t - \mathbf{w}^*)^T(\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t) \mid \mathcal{F}^t\right] = 0$. Therefore, by computing the expected value of both sides of (7.57) given the observations until step t and regrouping the terms we obtain

$$\begin{aligned}
&\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} - \mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^*\|^2_{\mathbf{G}} \mid \mathcal{F}^t\right] \\
&\geq \alpha\left[\frac{2}{L} - \frac{1}{\eta}\right]\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 + \mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^t\|^2_{\mathbf{G}} \mid \mathcal{F}^t\right] \\
&\quad + 2\mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^*\|^2_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}} \mid \mathcal{F}^t\right] - \alpha\eta\mathbb{E}\left[\|\mathbf{w}^t - \mathbf{w}^{t+1}\|^2 \mid \mathcal{F}^t\right] \\
&\quad - \mathbb{E}\left[2\alpha(\mathbf{w}^{t+1} - \mathbf{w}^t)^T(\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t) \mid \mathcal{F}^t\right].
\end{aligned} \tag{7.58}$$

By applying inequality $2\mathbf{a}^T\mathbf{b} \leq \eta\|\mathbf{a}\|^2 + \eta^{-1}\|\mathbf{b}\|^2$ for the vectors $\mathbf{a} = \mathbf{w}^{t+1} - \mathbf{w}^t$ and $\mathbf{b} = \nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t$, we obtain that the inner product $2(\mathbf{w}^{t+1} - \mathbf{w}^t)^T(\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t)$ is bounded above by $\eta\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 + (1/\eta)\|\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t\|^2$. Replacing $2(\mathbf{w}^{t+1} - \mathbf{w}^t)^T(\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t)$ in (7.58) by its upper bound $\eta\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 + (1/\eta)\|\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t\|^2$ yields

$$\begin{aligned}
&\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} - \mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^*\|^2_{\mathbf{G}} \mid \mathcal{F}^t\right] \\
&\geq \alpha\left[\frac{2}{L} - \frac{1}{\eta}\right]\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 + \mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^t\|^2_{\mathbf{G}} \mid \mathcal{F}^t\right] \\
&\quad + 2\mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^*\|^2_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}} \mid \mathcal{F}^t\right] - 2\alpha\eta\mathbb{E}\left[\|\mathbf{w}^t - \mathbf{w}^{t+1}\|^2 \mid \mathcal{F}^t\right] \\
&\quad - \frac{\alpha}{\eta}\mathbb{E}\left[\|\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t\|^2 \mid \mathcal{F}^t\right].
\end{aligned} \tag{7.59}$$

Observe that the squared norm $\|\mathbf{u}^{t+1} - \mathbf{u}^t\|^2_{\mathbf{G}}$ can be expanded as $\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2_{\tilde{\mathbf{Z}}} + \|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2$. Using this simplification for $\|\mathbf{u}^{t+1} - \mathbf{u}^t\|^2_{\mathbf{G}}$ and regrouping the terms in (7.59) lead to

$$\begin{aligned}
&\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} - \mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^*\|^2_{\mathbf{G}} \mid \mathcal{F}^t\right] \\
&\geq \alpha\left[\frac{2}{L} - \frac{1}{\eta}\right]\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 + \mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2_{\tilde{\mathbf{Z}}-2\alpha\eta\mathbf{I}} \mid \mathcal{F}^t\right] + \mathbb{E}\left[\|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2 \mid \mathcal{F}^t\right] \\
&\quad + 2\mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^*\|^2_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}} \mid \mathcal{F}^t\right] - \frac{\alpha}{\eta}\mathbb{E}\left[\|\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t\|^2 \mid \mathcal{F}^t\right].
\end{aligned} \tag{7.60}$$

We proceed by simplifying the expectation $\mathbb{E}\left[\|\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t\|^2 \,|\, \mathcal{F}^t\right]$ in (7.60). Note that by adding and subtracting the term $\nabla f(\mathbf{w}^*)$, we can rewrite the term $\mathbb{E}\left[\|\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t\|^2 \,|\, \mathcal{F}^t\right]$ as $\mathbb{E}\left[\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*) + \nabla f(\mathbf{w}^*) - \hat{\mathbf{g}}^t\|^2 \mid \mathcal{F}^t\right]$ and by expanding the squared norm and simplifying the terms we obtain

$$\mathbb{E}\left[\left\|\nabla f(\mathbf{w}^t) - \hat{\mathbf{g}}^t\right\|^2 \mid \mathcal{F}^t\right] = \mathbb{E}\left[\left\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\right\|^2 \mid \mathcal{F}^t\right] - \mathbb{E}\left[\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 \mid \mathcal{F}^t\right]. \tag{7.61}$$

Substituting the simplification in (7.61) into (7.60) yields

$$\begin{aligned}
&\|\mathbf{u}^t - \mathbf{u}^*\|_{\mathbf{G}}^2 - \mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2 \mid \mathcal{F}^t\right] \\
&\geq \frac{2\alpha}{L}\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 + \mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^t\|_{\tilde{\mathbf{Z}}-2\alpha\eta\mathbf{I}}^2 \mid \mathcal{F}^t\right] + \mathbb{E}\left[\|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2 \mid \mathcal{F}^t\right] \\
&\quad + 2\mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^*\|_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}}^2 \,|\, \mathcal{F}^t\right] - \frac{\alpha}{\eta}\mathbb{E}\left[\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\|^2 \mid \mathcal{F}^t\right].
\end{aligned} \tag{7.62}$$

Considering the strong convexity of the global objective function f with constant μ we can write $\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2 \geq 2\mu\left(f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*)\right)$. Substituting the squared norm $\left\|\nabla f(\mathbf{w}^t) - \nabla f(\mathbf{w}^*)\right\|^2$ by this lower bound in (7.62) follows

$$\begin{aligned}
&\|\mathbf{u}^t - \mathbf{u}^*\|_{\mathbf{G}}^2 - \mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2 \mid \mathcal{F}^t\right] \\
&\geq \frac{4\alpha\mu}{L}\left(f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*)\right) + \mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^t\|_{\tilde{\mathbf{Z}}-2\alpha\eta\mathbf{I}}^2 \mid \mathcal{F}^t\right] \\
&\quad + \mathbb{E}\left[\|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2 \mid \mathcal{F}^t\right] + 2\mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^*\|_{\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}}^2 \,|\, \mathcal{F}^t\right] - \frac{\alpha}{\eta}\mathbb{E}\left[\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\|^2 \mid \mathcal{F}^t\right].
\end{aligned} \tag{7.63}$$

Substituting the upper bound for the expectation $\mathbb{E}\left[\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\|^2 \mid \mathcal{F}^t\right]$ in (7.34) into (7.63) and regrouping the terms show the validity of the claim in (7.50). ■

Lemma 23 shows an upper bound for the squared norm $\|\mathbf{u}^{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2$ which is the first part of the Lyapunov function $\|\mathbf{u}^t - \mathbf{u}^*\|_{\mathbf{G}}^2 + cp^t$ at step $t+1$. Likewise, we provide an upper bound for the second term of the Lyapunov function at time $t+1$ which is p^{t+1} in terms of p^t and some parameters that capture optimality gap. This bound is studied in the following lemma.

Lemma 24 *Consider the DSA algorithm as defined in* (7.6)-(7.9) *and the definition of* p^t *in* (7.33)*. Further, define* $q_{\min}$ *and* $q_{\max}$ *as the smallest and largest values for the number of instantaneous functions at a node, respectively. If Assumptions 14-16 hold true, then for*

all $t > 0$ the sequence p^t satisfies

$$\mathbb{E}\left[p^{t+1} \mid \mathcal{F}^t\right] \leq \left[1 - \frac{1}{q_{\max}}\right] p^t + \frac{1}{q_{\min}} \left[f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*)\right]. \tag{7.64}$$

Proof: Given the information until time t, each auxiliary vector $\mathbf{y}_{v,i}^{t+1}$ is a random variable that takes values $\mathbf{y}_{v,i}^t$ and $\mathbf{w}_v^t$ with associated probabilities $1 - 1/q_v$ and $1/q_v$, respectively. This observation holds since with probability $1/q_v$ node v may choose index i to update at time $t+1$ and with probability $1 - (1/q_v)$ choose other indices. Therefore, we can write

$$\mathbb{E}\left[\frac{1}{q_v}\sum_{i=1}^{q_v}\left(\nabla f_{v,i}(\tilde{\mathbf{w}}^*)^T(\mathbf{y}_{v,i}^{t+1} - \tilde{\mathbf{w}}^*)\right) \mid \mathcal{F}^t\right] = \left[1 - \frac{1}{q_v}\right]\frac{1}{q_v}\sum_{i=1}^{q_v}\nabla f_{v,i}(\tilde{\mathbf{w}}^*)^T(\mathbf{y}_{v,i}^t - \tilde{\mathbf{w}}^*) + \frac{1}{q_v}\nabla f_v(\tilde{\mathbf{w}}^*)^T(\mathbf{w}_v^t - \tilde{\mathbf{w}}^*). \tag{7.65}$$

Likewise, the distribution of random function $f_{v,i}(\mathbf{y}_{v,i}^{t+1})$ given observation until time t has two possibilities $f_{v,i}(\mathbf{y}_{v,i}^t)$ and $f_{v,i}(\mathbf{w}_v^t)$ with associated probabilities $1 - 1/q_v$ and $1/q_v$, respectively. Hence, we can write $\mathbb{E}\left[f_{v,i}(\mathbf{y}_{v,i}^{t+1}) \mid \mathcal{F}^t\right] = (1-1/q_v)f_{v,i}(\mathbf{y}_{v,i}^t) + (1/q_v)f_{v,i}(\mathbf{w}_v^t)$. By summing this relation for all $i \in 1, \ldots, q_v$ and divining both sides of the resulted expression by q_v we obtain

$$\mathbb{E}\left[\frac{1}{q_v}\sum_{i=1}^{q_v} f_{v,i}(\mathbf{y}_{v,i}^{t+1}) \mid \mathcal{F}^t\right] = \left[1 - \frac{1}{q_v}\right]\frac{1}{q_v}\sum_{i=1}^{q_v} f_{v,i}(\mathbf{y}_{v,i}^t) + \frac{1}{q_v}f_v(\mathbf{w}_v^t). \tag{7.66}$$

To simplicity equations let us define the sequence p_n^t as

$$p_n^t := \frac{1}{q_v}\sum_{i=1}^{q_v} f_{v,i}(\mathbf{y}_{v,i}^t) - f_v(\tilde{\mathbf{w}}^*) - \frac{1}{q_v}\sum_{i=1}^{q_v}\nabla f_{v,i}(\tilde{\mathbf{w}}^*)^T(\mathbf{y}_{v,i}^t - \tilde{\mathbf{w}}^*). \tag{7.67}$$

Subtracting (7.65) from (7.66) and adding $-f_v(\tilde{\mathbf{w}}^*)$ to the both sides of equality in association with the definition of the sequence p_n^t in (7.67) yield

$$\mathbb{E}\left[p_n^{t+1} \mid \mathcal{F}^t\right] = \left[1 - \frac{1}{q_v}\right] p_n^t + \frac{1}{q_v}\left[f_v(\mathbf{w}_v^t) - f_v(\tilde{\mathbf{w}}^*) - \nabla f_v(\tilde{\mathbf{w}}^*)^T(\mathbf{w}_v^t - \tilde{\mathbf{w}}^*)\right]. \tag{7.68}$$

We proceed to find and upper bound for the terms in the right hand side of (7.68). First note that according to the strong convexity of the local instantaneous functions $f_{v,i}$ and local functions f_v both terms in the right hand side of (7.68) are non-negative. Observing that the number of instantaneous functions at each node q_v satisfies the condition $q_{\min} \leq q_v \leq q_{\max}$, we obtain

$$1 - \frac{1}{q_v} \leq 1 - \frac{1}{q_{\max}}, \qquad \frac{1}{q_v} \leq \frac{1}{q_{\min}}. \tag{7.69}$$

Substituting the upper bounds in (7.69) into (7.68), summing both sides of the implied inequality over $v \in \{1, \ldots, V\}$, and considering the definitions of the optimal argument $\mathbf{w}^* = [\tilde{\mathbf{w}}^*; \ldots; \tilde{\mathbf{w}}^*]$ and the aggregate function $f(\mathbf{w}) = \sum_{v=1}^{V} f_v(\mathbf{w}_v)$ lead to

$$\sum_{v=1}^{V} \mathbb{E}\left[p_n^{t+1} \mid \mathcal{F}^t\right] \leq \left[1 - \frac{1}{q_{\max}}\right] \sum_{v=1}^{V} p_n^t + \frac{1}{q_{\min}} \left[f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T (\mathbf{w}^t - \mathbf{w}^*)\right]. \tag{7.70}$$

Now observe that according to the definitions of the sequences p^t and p_n^t in (7.33) and (7.67), respectively, p^t is the sum of p_n^t for all v, i.e. $p^t = \sum_{v=1}^{V} p_n^t$. Hence, we can rewrite (7.70) as

$$\mathbb{E}\left[p^{t+1} \mid \mathcal{F}^t\right] \leq \left[1 - \frac{1}{q_{\max}}\right] p^t + \frac{1}{q_{\min}} \left[f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T (\mathbf{w}^t - \mathbf{w}^*)\right]. \tag{7.71}$$

Therefore, the claim in (7.64) is valid. ■

Lemma 24 provides an upper bound for p^{t+1} in terms of its previous value p^t and the optimality error $f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T (\mathbf{w}^t - \mathbf{w}^*)$. Combining the results in Lemmata 23 and 24 we can show that in expectation the Lyapunov function $\|\mathbf{u}^{t+1} - \mathbf{u}^*\|_{\mathbf{G}}^2 + c\, p^{t+1}$ at step $t+1$ is strictly smaller than its previous value $\|\mathbf{u}^t - \mathbf{u}^*\|_{\mathbf{G}}^2 + c\, p^t$ at step t, but before showing this result we prove the following intermediate lemma to establish an upper bound for the squared error norm $\|\mathbf{v}^t - \mathbf{v}^*\|^2$.

Lemma 25 *Consider the DSA algorithm as defined in* (7.6)-(7.9)*. Further, recall* γ' *as the smallest non-zero eigenvalue and* Γ' *as the largest eigenvalue of the matrix* $\tilde{\mathbf{Z}} - \mathbf{Z}$*. If Assumptions 14-16 hold, then the squared norm of the difference* $\|\mathbf{v}^t - \mathbf{v}^*\|^2$ *is bounded above as*

$$\|\mathbf{v}^t - \mathbf{v}^*\|^2 \leq \frac{8}{\gamma'} \mathbb{E}\left[\left\|\mathbf{w}^{t+1} - \mathbf{w}^*\right\|_{(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})^2}^2 \mid \mathcal{F}^t\right] + \frac{8}{\gamma'} \mathbb{E}\left[\left\|\mathbf{w}^t - \mathbf{w}^{t+1}\right\|_{\tilde{\mathbf{Z}}^2}^2 \mid \mathcal{F}^t\right] + \frac{16\alpha^2 L}{\gamma'}\, p^t$$
$$+ \frac{2\Gamma'}{\gamma'} \mathbb{E}\left[\|\mathbf{v}^t - \mathbf{v}^{t+1}\|^2 \mid \mathcal{F}^t\right] + \frac{8\alpha^2 (2L - \mu)}{\gamma'} \left[f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T (\mathbf{w}^t - \mathbf{w}^*)\right]. \tag{7.72}$$

Proof: Consider the inequality $\|\mathbf{a}+\mathbf{b}\|^2 \leq 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$ for the case that $\mathbf{a} = \mathbf{U}(\mathbf{v}^{t+1} - \mathbf{v}^*)$, $\mathbf{b} = \mathbf{U}(\mathbf{v}^t - \mathbf{v}^{t+1})$ which can be written as

$$\|\mathbf{U}(\mathbf{v}^t - \mathbf{v}^*)\|^2 \leq 2\|\mathbf{U}(\mathbf{v}^{t+1} - \mathbf{v}^*)\|^2 + 2\|\mathbf{U}(\mathbf{v}^t - \mathbf{v}^{t+1})\|^2. \tag{7.73}$$

We proceed by finding an upper bound for $2\|\mathbf{U}(\mathbf{v}^{t+1} - \mathbf{v}^*)\|^2$. Based on the result of Lemma 21 in (7.30), the term $\mathbf{U}(\mathbf{v}^{t+1} - \mathbf{v}^*)$ is equal to the sum of vectors $\mathbf{a} + \mathbf{b}$ where $\mathbf{a} = (\mathbf{I} + \mathbf{Z} - 2\tilde{\mathbf{Z}})(\mathbf{w}^{t+1} - \mathbf{w}^*) - \tilde{\mathbf{Z}}(\mathbf{w}^t - \mathbf{w}^{t+1})$ and $\mathbf{b} = -\alpha \hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)$. Therefore, using

the inequality $\|\mathbf{a}+\mathbf{b}\|^2 \leq 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$ we can write

$$\left\|\mathbf{U}(\mathbf{v}^{t+1}-\mathbf{v}^*)\right\|^2 \leq 2\left\|(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})(\mathbf{w}^{t+1}-\mathbf{w}^*) - \tilde{\mathbf{Z}}(\mathbf{w}^t-\mathbf{w}^{t+1})\right\|^2 + 2\alpha^2\left\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\right\|^2. \tag{7.74}$$

By using the inequality $\|\mathbf{a}+\mathbf{b}\|^2 \leq 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$ one more time for vectors $\mathbf{a} = (\mathbf{I} + \mathbf{Z} - 2\tilde{\mathbf{Z}})(\mathbf{w}^{t+1} - \mathbf{w}^*)$ and $\mathbf{b} = -\tilde{\mathbf{Z}}(\mathbf{w}^t - \mathbf{w}^{t+1})$, we obtain an upper bound for the term $\|(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})(\mathbf{w}^{t+1}-\mathbf{w}^*) - \tilde{\mathbf{Z}}(\mathbf{w}^t - \mathbf{w}^{t+1})\|^2$. Substituting this upper bound into (7.74) yields

$$\left\|\mathbf{U}(\mathbf{v}^{t+1}-\mathbf{v}^*)\right\|^2 \leq 4\left\|\mathbf{w}^{t+1}-\mathbf{w}^*\right\|^2_{(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})^2} + 4\left\|\mathbf{w}^t - \mathbf{w}^{t+1}\right\|^2_{\tilde{\mathbf{Z}}^2} + 2\alpha^2\left\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\right\|^2. \tag{7.75}$$

Inequality (7.75) shows an upper bound for $2\|\mathbf{U}(\mathbf{v}^{t+1}-\mathbf{v}^*)\|^2$ in (7.73). Moreover, we know that the second term $\|\mathbf{U}(\mathbf{v}^t - \mathbf{v}^{t+1})\|^2$ is also bounded above by $\Gamma'\|\mathbf{v}^t - \mathbf{v}^{t+1}\|^2$ where Γ' is the largest eigenvalue of matrix $\tilde{\mathbf{Z}} - \mathbf{Z} = \mathbf{U}^2$. Substituting these upper bounds into (7.73) and computing the expected value of both sides given the information until step t yield

$$\begin{aligned}\|\mathbf{U}(\mathbf{v}^t - \mathbf{v}^*)\|^2 \leq\ & 8\mathbb{E}\left[\left\|\mathbf{w}^{t+1}-\mathbf{w}^*\right\|^2_{(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})^2} \mid \mathcal{F}^t\right] + 8\mathbb{E}\left[\left\|\mathbf{w}^t - \mathbf{w}^{t+1}\right\|^2_{\tilde{\mathbf{Z}}^2} \mid \mathcal{F}^t\right] \\ & + 4\alpha^2\mathbb{E}\left[\left\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\right\|^2 \mid \mathcal{F}^t\right] + 2\Gamma'\mathbb{E}\left[\|\mathbf{v}^t - \mathbf{v}^{t+1}\|^2 \mid \mathcal{F}^t\right].\end{aligned} \tag{7.76}$$

Note the vectors $\mathbf{v}^t$ and $\mathbf{v}^*$ lie in the column space of the matrix $\mathbf{U}$. Thus, we obtain that $\|\mathbf{U}(\mathbf{v}^t - \mathbf{v}^*)\|^2 \geq \gamma'\|\mathbf{v}^t - \mathbf{v}^*\|^2$. Substituting this lower bound for $\|\mathbf{U}(\mathbf{v}^t - \mathbf{v}^*)\|^2$ in (7.76) and deviding both sides of the imposed inequality by γ' yield

$$\begin{aligned}\|\mathbf{v}^t - \mathbf{v}^*\|^2 \leq\ & \frac{8}{\gamma'}\mathbb{E}\left[\left\|\mathbf{w}^{t+1}-\mathbf{w}^*\right\|^2_{(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})^2} \mid \mathcal{F}^t\right] + \frac{8}{\gamma'}\mathbb{E}\left[\left\|\mathbf{w}^t - \mathbf{w}^{t+1}\right\|^2_{\tilde{\mathbf{Z}}^2} \mid \mathcal{F}^t\right] \\ & + \frac{4\alpha^2}{\gamma'}\mathbb{E}\left[\left\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\right\|^2 \mid \mathcal{F}^t\right] + \frac{2\Gamma'}{\gamma'}\mathbb{E}\left[\|\mathbf{v}^t - \mathbf{v}^{t+1}\|^2 \mid \mathcal{F}^t\right].\end{aligned} \tag{7.77}$$

By substituting the expectation $\mathbb{E}\left[\|\hat{\mathbf{g}}^t - \nabla f(\mathbf{w}^*)\|^2 \mid \mathcal{F}^t\right]$ in the right hand side of (7.77) with its upper bound in (7.34), the claim in (7.72) follows. ■

Using the result in Lemma 25 we show that the sequence $\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\ p^t$ converges linearly to zero.

Theorem 11 *Consider the DSA algorithm as defined in* (7.6)-(7.9). *Further recall the definition of the sequence* p^t *in* (7.33). *Define* η *as an arbitrary positive constant chosen from the interval*

$$\eta \in \left(\frac{L^2 q_{\max}}{\mu q_{\min}} + \frac{L^2}{\mu} - \frac{L}{2}\ ,\ \infty\right). \tag{7.78}$$

If Assumptions 14-16 hold true and the stepsize α is chosen from the interval $\alpha \in (0, \gamma/2\eta)$, then for arbitrary c chosen from the interval

$$c \in \left(\frac{4\alpha L q_{\max}}{\eta} \ , \ \frac{4\alpha\mu q_{\min}}{L} - \frac{2\alpha q_{\min}(2L-\mu)}{\eta} \right), \tag{7.79}$$

there exits a positive constant $0 < \delta < 1$ such that

$$\mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^*\|^2_{\mathbf{G}} + c\, p^{t+1} \mid \mathcal{F}^t \right] \leq (1-\delta) \left(\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\, p^t \right). \tag{7.80}$$

Proof: Proving the linear convergence claim in (7.80) is equivalent to showing that

$$\delta \|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + \delta c\ p^t \leq \|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} - \mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^*\|^2_{\mathbf{G}} \mid \mathcal{F}^t \right] + c\ (p^t - \mathbb{E}\left[p^{t+1} \mid \mathcal{F}^t \right]). \tag{7.81}$$

Substituting the terms $\mathbb{E}\left[\|\mathbf{u}^{t+1} - \mathbf{u}^*\|^2_{\mathbf{G}} \mid \mathcal{F}^t \right]$ and $\mathbb{E}\left[p^{t+1} \mid \mathcal{F}^t \right]$ by their upper bounds as introduced in Lemma 23 and Lemma 24, respectively, yields a sufficient condition for the claim in (7.81) as

$$\begin{aligned}
\delta \|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + \delta c\ p^t \leq\ & \mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2_{\tilde{\mathbf{Z}} - 2\alpha\eta\mathbf{I}} \mid \mathcal{F}^t \right] + \mathbb{E}\left[\|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2 \mid \mathcal{F}^t \right] \\
& + 2\mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^*\|^2_{\mathbf{I} + \mathbf{Z} - 2\tilde{\mathbf{Z}}} \mid \mathcal{F}^t \right] + \left(\frac{c}{q_{\max}} - \frac{4\alpha L}{\eta} \right) p^t \\
& + \left[\frac{4\alpha\mu}{L} - \frac{2\alpha(2L-\mu)}{\eta} - \frac{c}{q_{\min}} \right] \left[f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T (\mathbf{w}^t - \mathbf{w}^*) \right].
\end{aligned} \tag{7.82}$$

We emphasize that if the inequality in (7.82) holds, then the inequalities in (7.81) and (7.80) are valid. Note that the weighted norm $\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}}$ in the left hand side of (7.82) can be simplified as $\|\mathbf{w}^t - \mathbf{w}^*\|^2_{\tilde{\mathbf{Z}}} + \|\mathbf{v}^t - \mathbf{v}^*\|^2$. Considering the definition of Γ as the maximum eigenvalue of the matrix $\tilde{\mathbf{Z}}$, we can conclude that $\|\mathbf{w}^t - \mathbf{w}^*\|^2_{\tilde{\mathbf{Z}}}$ is bounded above by $\Gamma \|\mathbf{w}^t - \mathbf{w}^*\|^2$. Considering this relation and observing the upper bound for $\|\mathbf{v}^t - \mathbf{v}^*\|^2$ in (7.72), we obtain that $\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} = \|\mathbf{w}^t - \mathbf{w}^*\|^2_{\tilde{\mathbf{Z}}} + \|\mathbf{v}^t - \mathbf{v}^*\|^2$ is bounded above by

$$\begin{aligned}
\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} \ \leq\ & \frac{8}{\gamma'} \mathbb{E}\left[\left\| \mathbf{w}^{t+1} - \mathbf{w}^* \right\|^2_{(\mathbf{I} + \mathbf{Z} - 2\tilde{\mathbf{Z}})^2} \mid \mathcal{F}^t \right] + \frac{8}{\gamma'} \mathbb{E}\left[\left\| \mathbf{w}^t - \mathbf{w}^{t+1} \right\|^2_{\tilde{\mathbf{Z}}^2} \mid \mathcal{F}^t \right] + \frac{16\alpha^2 L}{\gamma'}\ p^t \\
& + \frac{2\Gamma'}{\gamma'} \mathbb{E}\left[\|\mathbf{v}^t - \mathbf{v}^{t+1}\|^2 \mid \mathcal{F}^t \right] + \Gamma \|\mathbf{w}^t - \mathbf{w}^*\|^2 \\
& + \frac{8\alpha^2 (2L-\mu)}{\gamma'} \left[f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T (\mathbf{w}^t - \mathbf{w}^*) \right].
\end{aligned} \tag{7.83}$$

Further, the strong convexity of the global objective function f implies that the squared norm $\|\mathbf{w}^t - \mathbf{w}^*\|^2$ is upper bound by $(2/\mu)(f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T (\mathbf{w}^t - \mathbf{w}^*))$. Replacing

the the squared norm $\|\mathbf{w}^t - \mathbf{w}^*\|^2$ in (7.83) by its upper bound leads to

$$\begin{aligned}\|\mathbf{u}^t - \mathbf{u}^*\|_{\mathbf{G}}^2 \leq \frac{8}{\gamma'}\mathbb{E}\left[\left\|\mathbf{w}^{t+1} - \mathbf{w}^*\right\|_{(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})^2}^2 \mid \mathcal{F}^t\right] + \frac{8}{\gamma'}\mathbb{E}\left[\left\|\mathbf{w}^t - \mathbf{w}^{t+1}\right\|_{\tilde{\mathbf{Z}}^2}^2 \mid \mathcal{F}^t\right] + \frac{16\alpha^2 L}{\gamma'}\ p^t \\ + \frac{2\Gamma'}{\gamma'}\mathbb{E}\left[\|\mathbf{v}^t - \mathbf{v}^{t+1}\|^2 \mid \mathcal{F}^t\right] \\ + \left(\frac{8\alpha^2\,(2L-\mu)}{\gamma'} + \frac{2\Gamma}{\mu}\right)\left[f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*)\right]. \quad (7.84)\end{aligned}$$

Replacing $\|\mathbf{u}^t - \mathbf{u}^*\|_{\mathbf{G}}^2$ in (7.82) by the upper bound in (7.84) and regrouping the terms lead to

$$\begin{aligned}0 \leq\ & \mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^t\|_{\tilde{\mathbf{Z}}-\alpha(\eta+\eta)\mathbf{I}-\frac{8\delta}{\gamma'}\tilde{\mathbf{Z}}^2}^2 \mid \mathcal{F}^t\right] \\ &+ \mathbb{E}\left[\|\mathbf{w}^{t+1} - \mathbf{w}^*\|_{(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})^{1/2}\left[2\mathbf{I}-\frac{8\delta}{\gamma'}(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})\right](\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})^{1/2}}^2 \mid \mathcal{F}^t\right] \\ &+ \mathbb{E}\left[\|\mathbf{v}^{t+1} - \mathbf{v}^t\|_{(1-\frac{2\delta\Gamma'}{\gamma'})\mathbf{I}}^2 \mid \mathcal{F}^t\right] + \left[\frac{c}{q_{\max}} - \frac{4\alpha L}{\eta} - \delta c - \frac{16\delta\alpha^2 L}{\gamma'}\right] p^t \\ &+ \left[\frac{4\alpha\mu}{L} - \frac{2\alpha(2L-\mu)}{\eta} - \frac{c}{q_{\min}} - \frac{8\delta\alpha^2\,(2L-\mu)}{\gamma'} - \frac{2\delta\Gamma}{\mu}\right](f(\mathbf{w}^t) - f(\mathbf{w}^*) - \nabla f(\mathbf{w}^*)^T(\mathbf{w}^t - \mathbf{w}^*)).\end{aligned} \quad (7.85)$$

Notice that if the inequality in (7.85) holds true, then the relation in (7.82) is valid and as we mentioned before the claim in (7.81) holds. To verify the sum in the right hand side of (7.85) is always positive and the inequality is valid, we enforce each summands in the right hand side of (7.85) to be non-negative. Therefore, the following conditions should be satisfied

$$\begin{aligned}&\gamma - \alpha(\eta+\eta) - \frac{8\delta}{\gamma'}\Gamma^2 \geq 0, \quad 2 - \frac{8\delta}{\gamma'}\lambda_{max}(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}}) \geq 0, \quad 1 - \frac{2\delta\Gamma'}{\gamma'} \geq 0, \\ &\frac{c}{q_{\max}} - \frac{4\alpha L}{\eta} - \delta c - \frac{16\delta\alpha^2 L}{\gamma'} \geq 0, \quad \frac{4\alpha\mu}{L} - \frac{2\alpha(2L-\mu)}{\eta} - \frac{c}{q_{\min}} - \frac{8\delta\alpha^2\,(2L-\mu)}{\gamma'} - \frac{2\delta\Gamma}{\mu} \geq 0.\end{aligned} \quad (7.86)$$

Recall that γ is the smallest eigenvalue of the positive definite matrix $\mathbf{Z}$. All the inequalities in (7.86) are satisfied, if δ is chosen as

$$\begin{aligned}\delta = \min\Bigg\{ &\frac{(\gamma - 2\alpha\eta)\gamma'}{8\Gamma^2}, \frac{\gamma'}{4\lambda_{max}(\mathbf{I}+\mathbf{Z}-2\tilde{\mathbf{Z}})}, \frac{\gamma'}{2\Gamma'}, \frac{\gamma'(c\eta - 4\alpha L q_{\max})}{\eta q_{\max}(c\gamma' + 16\alpha^2 L)}, \\ &\left[\frac{4\alpha\mu}{L} - \frac{2\alpha(2L-\mu)}{\eta} - \frac{c}{q_{\min}}\right]\left[\frac{8\alpha^2\,(2L-\mu)}{\gamma'} + \frac{2\Gamma}{\mu}\right]^{-1}\Bigg\}.\end{aligned} \quad (7.87)$$

where η, c and α are selected from the intervals

$$\eta \in \left(\frac{L^2 q_{\max}}{\mu q_{\min}} + \frac{L^2}{\mu} - \frac{L}{2} \ , \ \infty \right), \ \alpha \in \left(0 \ , \ \frac{\gamma}{2\eta} \right),$$
$$c \in \left(\frac{4\alpha L q_{\max}}{\eta} \ , \ \frac{4\alpha\mu q_{\min}}{L} - \frac{2\alpha q_{\min}(2L-\mu)}{\eta} \right). \tag{7.88}$$

Notice that considering the conditions for the variables η, α and c in (7.88), the constant δ in (7.87) is strictly positive $\delta > 0$. Moreover, according to the definition in (7.87) the constant δ is smaller than $\gamma'/2\Gamma'$ which leads to the conclusion that $\delta \leq 1/2 < 1$. Therefore, we obtain that $0 < \delta < 1$ and the claim in (7.80) is valid. ■

We point out that the linear convergence constant δ in (7.87) is a function of the strong convexity parameter μ, the Lipschitz continuity constant L, lower and upper bounds on the eigenvalues of the matrices $\tilde{\mathbf{Z}}$, $\tilde{\mathbf{Z}} - \mathbf{Z}$, and $\mathbf{I} + \mathbf{Z} - 2\tilde{\mathbf{Z}}$, the smallest $q_{\min}$ and largest $q_{\max}$ values for the number of instantaneous functions available at a node, and the stepsize α. Insight on the dependence of δ with problem parameters is offered in Section 7.3.3.

The inequality in (7.80) shows that the expected value of the sequence $\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t$ at time $t+1$ given the observation until step t is strictly smaller than the previous iterate at step t. Note that, it is not hard to verify that if the positive constant η is chosen from the interval in (7.78), the interval in (7.79) is non-empty. Computing the expected value with respect to the initial sigma field $\mathbb{E}\left[. \mid \mathcal{F}^0\right] = \mathbb{E}\left[.\right]$ implies that in expectation the sequence $\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t$ converges linearly to null, i.e.,

$$\mathbb{E}\left[\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t\right] \leq (1-\delta)^t \left(\|\mathbf{u}^0 - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^0\right). \tag{7.89}$$

We use the result in (7.89) to establish linear convergence of the sequence of squared norm error $\|\mathbf{w}^t - \mathbf{w}^*\|^2$ in expectation.

Corollary 2 *Consider the DSA algorithm as defined in* (7.6)-(7.9) *and recall* γ *is the minimum eigenvalue of the positive definite matrix* $\tilde{\mathbf{Z}}$*. Suppose the conditions of Theorem 11 hold, then there exits a positive constant* $0 < \delta < 1$ *such that*

$$\mathbb{E}\left[\|\mathbf{w}^t - \mathbf{w}^*\|^2\right] \leq (1-\delta)^t \frac{\left(\|\mathbf{u}^0 - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^0\right).}{\gamma} \tag{7.90}$$

Proof: First note that according to the definitions of $\mathbf{u}$ and $\mathbf{G}$ in (7.49) and the definition of p^t in (7.33) , we can write $\|\mathbf{w}^t - \mathbf{w}^*\|^2_{\tilde{\mathbf{Z}}} \leq \|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t$. Further, note that the weighted norm $\|\mathbf{w}^t - \mathbf{w}^*\|^2_{\tilde{\mathbf{Z}}}$ is lower bounded by $\gamma\|\mathbf{w}^t - \mathbf{w}^*\|^2$, since γ is a lower bound for the eigenvalues of $\tilde{\mathbf{Z}}$. Combine these two observations to obtain $\gamma\|\mathbf{w}^t - \mathbf{w}^*\|^2 \leq \|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + cp^t$. This inequality in conjunction with the expression in (7.89) follows the claim in (7.90). ■

Corollary 2 states that the sequence $\mathbb{E}\left[\|\mathbf{w}^t - \mathbf{w}^*\|^2\right]$ linearly converges to null. Note that the sequence $\mathbb{E}\left[\|\mathbf{w}^t - \mathbf{w}^*\|^2\right]$ is not necessarily monotonically decreasing as the sequence $\mathbb{E}\left[\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t\right]$ is. The result in (7.90) shows linear convergence of the sequence of variables generated by DSA in expectation. In the following Theorem we show that the local variables $\mathbf{w}_v^t$ generated by DSA almost surely converge to the optimal argument of (7.1).

Theorem 12 *Consider the DSA algorithm as defined in* (7.6)-(7.9) *and suppose the conditions of Theorem 11 hold. Then, the sequences of the local variables* $\mathbf{w}_v^t$ *for all* $v = 1, \ldots, V$ *converge almost surely to the optimal argument* $\tilde{\mathbf{w}}^*$*, i.e.,*

$$\lim_{t\to\infty} \mathbf{w}_v^t \;=\; \tilde{\mathbf{w}}^* \quad a.s. \qquad \text{for all } v = 1, \ldots, V. \tag{7.91}$$

Proof: The proof uses the relationship in the statement (7.80) of Theorem 11 to build a supermartingale sequence. To do this define the stochastic processes ζ^t and β^t as

$$\zeta^t := \|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t, \quad \beta^t := \delta\left(\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t\right). \tag{7.92}$$

Note that the stochastic processes ζ^t and β^t are alway non-negative. Let now $\mathcal{F}_t$ be a sigma-algebra measuring ζ^t, β^t, and $\mathbf{u}^t$. Considering the definitions of ζ^t and β^t and the relation in (7.80) we can write

$$\mathbb{E}\left[\zeta^{t+1} \mid \mathcal{F}^t\right] \leq \zeta^t - \beta^t. \tag{7.93}$$

Since the sequences α^t and β^t are nonnegative it follows from (7.93) that they satisfy the conditions of the supermartingale convergence theorem – see e.g., Theorem E7.4 [114]. Therefore, we obtain that: (i) The sequence ζ^t converges almost surely. (ii) The sum $\sum_{t=0}^{\infty} \beta^t < \infty$ is almost surely finite. The definition of β^t in (7.92) implies that

$$\sum_{t=0}^{\infty} \delta\left(\|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t\right) < \infty, \qquad \text{a.s.} \tag{7.94}$$

Since $\|\mathbf{w}^t - \mathbf{w}^*\|^2_{\tilde{\mathbf{Z}}} \leq \|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t$ and the eigenvalues of $\tilde{\mathbf{Z}}$ are lower bounded by γ we can write $\gamma\|\mathbf{w}^t - \mathbf{w}^*\|^2 \leq \|\mathbf{u}^t - \mathbf{u}^*\|^2_{\mathbf{G}} + c\,p^t$. This inequality in association with the fact that the sum in (7.94) is finite leads to

$$\sum_{t=0}^{\infty} \delta\,\gamma\,\|\mathbf{w}^t - \mathbf{w}^*\|^2 < \infty, \qquad \text{a.s.} \tag{7.95}$$

Observing the fact that δ and γ are positive constants, we can conclude from (7.95) that

the sequence $\|\mathbf{w}^t - \mathbf{w}^*\|^2$ is almost surely summable and it converges with probability 1 to zero. ■

Theorem 12 provides almost sure convergence of $\mathbf{w}^t$ to the optimal solution $\mathbf{w}^*$.

7.3.3 Linear convergence constant

The constant δ that controls the speed of convergence can be simplified by selecting specific values for η, α, and c. This uncovers connections to the properties of the local objective functions and the network topology. To make this clearer recall the definitions of γ and Γ as the smallest and largest eigenvalues of the positive definite matrix $\tilde{\mathbf{Z}}$, respectively, and γ' and Γ' as the smallest and largest *positive* eigenvalues of the positive semi-definite matrix $\tilde{\mathbf{Z}} - \mathbf{Z}$, respectively. Further, recall that the local objective functions are strongly convex with constant μ and their gradients are Lipschitz continuous with constant L. Then, define the condition numbers of the objective function and the graph as

$$\kappa_f = \frac{L}{\mu}, \qquad \kappa_g = \frac{\max\{\Gamma, \Gamma'\}}{\min\{\gamma, \gamma'\}}, \tag{7.96}$$

respectively. The condition number of the function is a measure of how difficult it is to minimize the local functions using gradient descent directions. The condition number of the graph is a measure of how slow the graph is in propagating a diffusion process. Both are known to control the speed of convergence of distributed optimization methods. The following corollary illustrates that these condition numbers also determine the convergence speed of DSA.

Corollary 3 *Consider the DSA algorithm as defined in* (7.6)-(7.9) *and suppose the conditions of Theorem 11 hold. Choose the weight matrices* $\mathbf{W}$ *and* $\tilde{\mathbf{W}}$ *as* $\tilde{\mathbf{W}} = (\mathbf{I}+\mathbf{W})/2$*, assign the same number of instantaneous local functions* $f_{v,i}$ *to each node, i.e.,* $q_{\min} = q_{\max} = q$*, and set the constants* η*,* α *and* c *as*

$$\eta = \frac{2L^2}{\mu}, \quad \alpha = \frac{\gamma\mu}{8L^2}, \quad c = \frac{q\gamma\mu^2}{4L^3}\left(1 + \frac{\mu}{4L}\right). \tag{7.97}$$

The linear convergence constant $0 < \delta < 1$ *in* (7.80) *reduces to*

$$\delta = \min\left[\frac{1}{16\kappa_g^2},\ \frac{1}{q[1 + 4\kappa_f(1 + \gamma/\gamma')]},\ \frac{1}{4(\gamma/\gamma')\kappa_f + 32\kappa_g\kappa_f^4}\right]. \tag{7.98}$$

Proof: The given values for η, α, and c satisfy the conditions in Theorem 11. Substitute then these values into the expression for δ in (7.87). Simplify terms and utilize the condition number definitions in (7.96). The second term in the minimization in (7.87) becomes

redundant because it is dominated by the first. ■

Observe that while the choices of η, α, and c in (7.97) satisfy all the required conditions of Theorem 11, they are not necessarily optimal for maximizing the linear convergence constant δ. Nevertheless, the expression in (7.98) shows that the convergence speed of DSA decreases with increases in the graph condition number κ_g, the local functions condition number κ_f, and the number of functions assigned to each node q. For a cleaner expression observe that both, γ and γ' are the minimum eigenvalues of the weight matrix $\mathbf{W}$ and the weight matrix difference $\tilde{\mathbf{W}} - \mathbf{W}$. They can therefore be chosen to be of similar order. For reference, say that we choose $\gamma = \gamma'$ so that the ratio $\gamma/\gamma' = 1$. In that case, the constant δ in (7.98) reduces to

$$\delta = \min\left[\frac{1}{16\kappa_g^2},\ \frac{1}{q(1+8\kappa_f)},\ \frac{1}{4(\kappa_f + 8\kappa_f^4\kappa_g)}\right]. \tag{7.99}$$

The three terms in (7.99) establish separate regimes, problems where the graph condition number is large, problems where the number of functions at each node is large, and problems where the condition number of the local functions are large. In the first regime the first term in (7.99) dominates and establishes a dependence in terms of the square of the graph's condition number. In the second regime the middle term dominates and results in an inverse dependence with the number of functions available at each node. In the third regime, the third term dominates. The dependence in this case is inversely proportional to κ_f^4.

7.4 Numerical experiments

We numerically study the performance of DSA in solving a logistic regression problem. In this problem we are given $N = \sum_{v=1}^{V} q_v$ training samples that we distribute across V distinct nodes. Denote q_v as the number of samples that are assigned to node v. We assume that the samples are distributed equally over the nodes, i.e., $q_v = q_{max} = q_{min} = q = N/V$ for $v = 1, \ldots, V$. The training points at node v are denoted by $\mathbf{x}_{v,i} \in \mathbb{R}^p$ for $i = 1, \ldots, q_v$ with associated labels $y_{v,i} \in \{-1, 1\}$. The goal is to predict the probability $\mathrm{P}(y = 1 \mid \mathbf{x})$ of having label $y = 1$ for sample point $\mathbf{x}$. The logistic regression model assumes that this probability can be computed as $\mathrm{P}(y = 1 \mid \mathbf{x}) = 1/(1 + \exp(-\mathbf{x}^T\mathbf{w}))$ given a linear classifier $\mathbf{w}$ that is computed based on the training samples. It follows from this model that the regularized maximum log likelihood estimate of the classifier $\mathbf{w}$ given the training samples $(\mathbf{x}_{v,i}, y_{v,i})$ for $i = 1, \ldots, q_v$ and $v = 1, \ldots, N$ is the solution of the optimization problem

$$\tilde{\mathbf{w}}^* := \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{v=1}^{V}\sum_{i=1}^{q_v} \log\Big(1 + \exp(-y_{v,i}\mathbf{x}_{v,i}^T\mathbf{w})\Big), \tag{7.100}$$

where the regularization term $(\lambda/2)\|\mathbf{w}\|^2$ is added to reduce overfitting to the training set. The optimization problem in (7.100) can be written in the form of (7.1) by defining the local objective functions f_v as

$$f_v(\mathbf{w}) = \frac{\lambda}{2V}\|\mathbf{w}\|^2 + \sum_{i=1}^{q_v} \log\Big(1 + \exp(-y_{v,i}\mathbf{x}_{v,i}^T\mathbf{w})\Big). \tag{7.101}$$

Observe that the local functions f_v in (7.101) can be written as the average of a set of instantaneous functions $f_{v,i}$ defined as

$$f_{v,i}(\mathbf{w}) = \frac{\lambda}{2V}\|\mathbf{w}\|^2 + q_v \log\Big(1 + \exp\left(-y_{v,i}\mathbf{x}_{v,i}^T\mathbf{w}\right)\Big), \tag{7.102}$$

for all $i = 1, \ldots, q_v$. Considering the definitions of the instantaneous local functions $f_{v,i}$ in (7.102) and the local functions f_v in (7.101), problem (7.100) can be solved using the DSA algorithm.

In our experiments in Sections 7.4.1-7.4.4, we use a synthetic dataset where the components of the feature vectors $\mathbf{x}_{n,i}$ with label $y_{v,i} = 1$ are generated from a normal distribution with mean μ and standard deviation σ_+, while sample points with label $y_{v,i} = -1$ are generated from a normal distribution with mean $-\mu$ and standard deviation σ_-. In Section 7.4.5, we consider a large-scale real dataset for training the classifier.

We consider a network of size V where the edges between nodes are generated randomly with probability p_c. The weight matrix $\mathbf{W}$ is generated using the Laplacian matrix $\mathbf{L}$ of network as

$$\mathbf{W} = \mathbf{I} - \mathbf{L}/\tau, \tag{7.103}$$

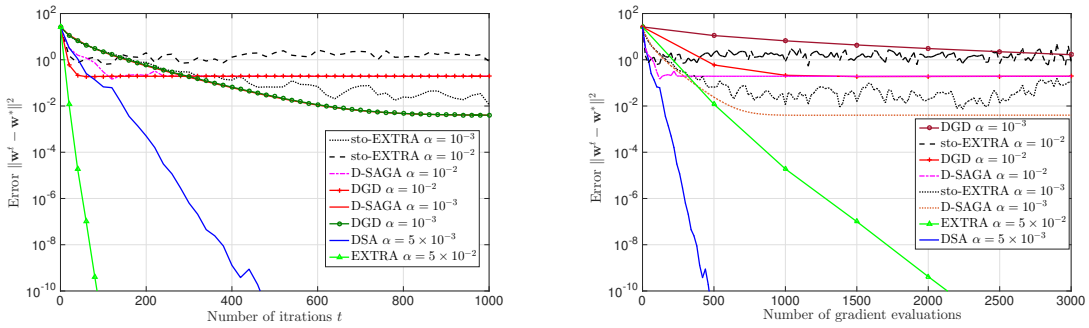
where τ should satisfy $\tau > (1/2)\lambda_{\max}(\mathbf{L})$. In our experiments we set this parameter as $\tau = (2/3)\lambda_{\max}(\mathbf{L})$. We capture the error of each algorithm by the sum of squared differences of the local iterates $\mathbf{w}_v^t$ from the optimal solution $\tilde{\mathbf{w}}^*$ as

$$e^t = \|\mathbf{w}^t - \mathbf{w}^*\|^2 = \sum_{v=1}^{V} \|\mathbf{w}_v^t - \tilde{\mathbf{w}}^*\|^2. \tag{7.104}$$

We use a centralized algorithm for computing the optimal argument $\tilde{\mathbf{w}}^*$ in all of our experiments.

7.4.1 Comparison with decentralized methods

We provide a comparison of DSA with respect to DGD, EXTRA, stochastic EXTRA, and decentralized SAGA. The stochastic EXTRA (sto-EXTRA) is defined by using the stochastic gradient in (7.5) instead of using full gradient as in EXTRA or stochastic averaging



(a) (b)

Figure 7.2: Convergence paths of DSA, EXTRA, DGD, Stochastic EXTRA, and Decentralized SAGA for a logistic regression problem with $N = 500$ samples and $V = 20$ nodes. Distance to optimality $e^t = \|\mathbf{w}^t - \mathbf{w}^*\|^2$ is shown with respect to number of iterations t and number of gradient evaluations in Fig 7.2(a) and Fig. 7.2(b), respectively. DSA and EXTRA converge linearly to the optimal argument $\mathbf{w}^*$, while DGD, Stochastic EXTRA, and Decentralized SAGA with constant step sizes converge to a neighborhood of the optimal solution. Smaller choice of stepsize for DGD, Stochastic EXTRA, and Decentralized SAGA leads to a more accurate convergence, while the speed of convergence becomes slower. DSA outperforms EXTRA in terms of number of gradient evaluations to achieve a target accuracy.

gradient as in DSA. The decentralized SAGA (D-SAGA) is a stochastic version of the DGD algorithm that uses stochastic averaging gradient instead of exact gradient which is the naive approach for developing a decentralized version of the SAGA algorithm. In our experiments, the weight matrix $\tilde{\mathbf{W}}$ in EXTRA, stochastic EXTRA, and DSA is chosen as $\tilde{\mathbf{W}} = (\mathbf{I}+\mathbf{W})/2$. We use the total number of sample points $N = 500$, feature vectors dimension $p = 2$, regularization parameter $\lambda = 10^{-4}$, probability of existence of an edge $p_c = 0.35$. To make the dataset *not* linearly separable we set the mean as $\mu = 2$ and the standard deviations to $\sigma_+ = \sigma_- = 2$. Moreover, the maximum eigenvalue of the Laplacian matrix is $\lambda_{\max}(\mathbf{L}) = 8.017$ which implies that the choice of τ in (7.103) is $\tau = (2/3)\lambda_{\max}(\mathbf{L}) = 5.345$. We set the total number of nodes as $V = 20$ which implies that each node has access to $q = N/V = 25$ sample points.

Fig. 7.2 illustrates the convergence paths of DSA, EXTRA, DGD, Stochastic EXTRA, and Decentralized SAGA with constant stepsizes for $N = 20$ nodes. For EXTRA and DSA different stepsizes are chosen and the best performance for EXTRA and DSA are achieved by $\alpha = 5 \times 10^{-2}$ and $\alpha = 5 \times 10^{-3}$, respectively. It is worth mentioning that the choice of stepsize α for DSA in practice is larger than the theoretical result in Theorem 6 and Corollary 9 which suggest stepsize of the order $O(\mu/L^2)$. As shown in Fig. 7.2, DSA is the only stochastic algorithm that converges linearly. Decentralized SAGA after a few iterations achieves the performance of DGD and they both cannot converge to the optimal argument. By choosing a smaller stepsize as $\alpha = 10^{-3}$, they reach a more accurate convergence relative

to the case that the stepsize is $\alpha = 10^{-2}$; however, the speed of convergence is slower for the smaller stepsize. Stochastic EXTRA also suffers from inexact convergence, but for a different reason. DGD and decentralized SAGA have inexact convergence since they solve a penalty version of the original problem, while stochastic EXTRA can not reach the optimal solution since the noise of stochastic gradient is not vanishing. DSA resolves both issues by combining the idea of stochastic averaging from SAGA to control the noise of stochastic gradient estimation and the double descent idea of EXTRA to solve the correct optimization problem.

Fig. 7.2(a) illustrates convergence paths of the considered methods in terms of number of iterations t. Notice that the number of iterations t indicates the number of local iterations processed at each node. Convergence rate of EXTRA is faster than DSA in terms of number of iterations or equivalently number of communications as shown in Fig. 7.2(a); however, the complexity of each iteration for EXTRA is higher than DSA. Therefore, it is reasonable to compare the performances of these algorithms in terms of number of processed feature vectors or equivalently number of gradient evaluations. For instance, DSA requires $t = 380$ iterations or equivalently 380 gradient evaluations to achieve the error $e^t = 10^{-8}$, while to achieve the same accuracy EXTRA requires $t = 69$ iterations which is equivalent to $t \times q_v = 69 \times 25 = 1725$ processed feature vectors or gradient evaluations.

To illustrate this difference better, we compare the convergence paths of DSA, EXTRA, DGD, Stochastic EXTRA, and Decentralized SAGA in terms of number of gradient evaluations in Fig. 7.2(b). Note that the total number of gradient evaluations at each node for the stochastic methods such as DSA, sto-EXTRA, and D-SAGA is equal to the the number of iterations t, while for EXTRA and DGD – which are deterministic methods – the number of gradient evaluations is equal to the product $t \times q$. This is true since each node in the stochastic methods only evaluates 1 gradient per iteration, while in the deterministic methods each node requires q gradient evaluations per iteration. The convergence paths in Fig. 7.2(b) showcase the advantage of DSA relative to EXTRA in requiring less processed feature vectors (or equivalently gradient evaluations) for achieving a specific accuracy. It is important to mention that the initial gradient evaluations for the DSA method is not considered in Fig. 7.2(b) since the initial decision variable is $\mathbf{w}^0 = \mathbf{0}$ in all experiments and evaluation of the initial gradients $\nabla f_{v,i}(\mathbf{w}^0) = -(1/2)qy_{v,i}\mathbf{x}_{v,i}$ is not computationally expensive relative to the general gradient computation which is given by $\nabla f_{v,i}(\mathbf{w}) = (\lambda \mathbf{w}/V) - (qy_{v,i}\mathbf{x}_{v,i})/(1 + \exp(y_{v,i}\mathbf{w}^T\mathbf{x}_{v,i}))$. However, if we consider this initial processing the plot for DSA in Fig. 7.2(b) will be shifted by $q = 25$ gradient evaluations which doesn't change the conclusion that DSA outperforms EXTRA in terms of gradient evaluations

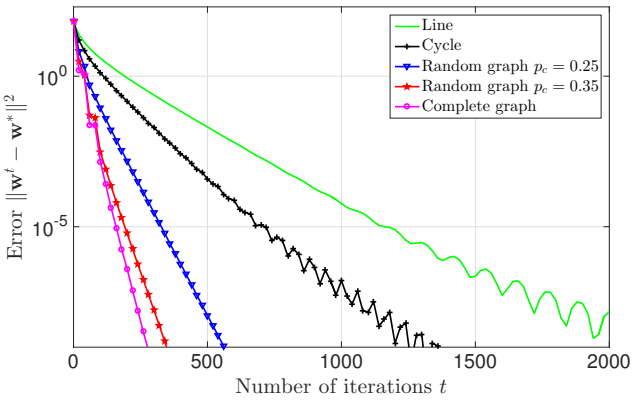
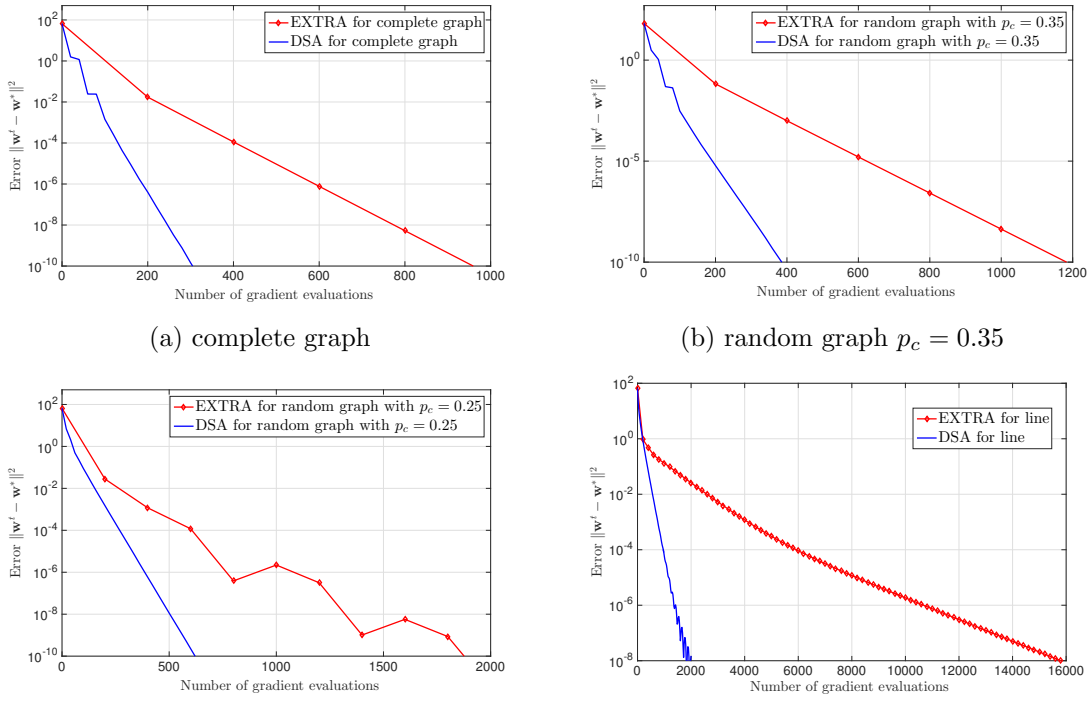


Figure 7.3: Convergence of DSA for different network topologies when the total number of samples is $N = 500$ and the size of network is $V = 50$. Distance to optimality $e^t = \|\mathbf{w}_t - \mathbf{w}^*\|^2$ is shown with respect to number of iterations t. As the graph condition number κ_g becomes larger the linear convergence of DSA becomes slower. The best performance belongs to the complete graph which has the smallest condition number and the slowest convergence path belongs to the line graph which has the largest graph condition number.

7.4.2 Effect of graph condition number κ_g

In this section we study the effect of the graph condition number κ_g as defined in (7.96) on the performance of DSA. We keep the parameters in Fig. 7.2 except for the network size V which we set as $V = 50$. Thus, each node has access to $q = 500/50 = 10$ sample points. The convergence paths of the DSA algorithm for random networks with $p_c = 0.25$ and $p_c = 0.35$, complete graph, cycle, and line are shown in Fig. 7.3. Notice that the graph condition number of the line graph, cycle graph, random graph with $p_c = 0.25$, random graph with $p_c = 0.35$, and complete graph are $\kappa_g = 1.01 \times 10^3$, $\kappa_g = 2.53 \times 10^2$, $\kappa_g = 17.05$, $\kappa_g = 4.87$, and $\kappa_g = 4$, respectively. For each network topology, we have hand-optimized the stepsize α and the best choice of stepsize for the complete graph, random graph with $p_c = 0.35$, random graph with $p_c = 0.25$, cycle, and line are $\alpha = 2 \times 10^{-2}$, $\alpha = 1.5 \times 10^{-2}$, $\alpha = 10^{-2}$, $\alpha = 5 \times 10^{-3}$, and $\alpha = 3 \times 10^{-3}$, respectively.

As we expect for the topologies that the graph has more edges and the graph condition number κ_g is smaller we observe a faster linear convergence for DSA. The best performance belongs to the complete graph which requires $t = 247$ iterations to achieve the relative error $e^t = 10^{-8}$. In the random graphs with connectivity probabilities $p_c = 0.35$ and $p_c = 0.25$, DSA achieves the relative error $e^t = 10^{-8}$ after $t = 310$ and $t = 504$ iterations, respectively. For the cycle and line graphs the numbers of required iterations for reaching the relative error $e^t = 10^{-8}$ are $t = 1133$ and $t = 1819$, respectively. These observations match the theoretical result in (7.99) that DSA converges faster when the graph condition number κ_g



(a) complete graph

(b) random graph $p_c = 0.35$

(c) random graph $p_c = 0.25$

(d) line

Figure 7.4: Convergence paths of DSA and EXTRA for different network topologies when the total number of samples is $N = 500$ and the size of network is $V = 50$. Distance to optimality $e^t = \|\mathbf{w}^t - \mathbf{w}^*\|^2$ is shown with respect to number of gradient evaluations. DSA converges faster relative to EXTRA in all of the considered networks. The difference between the convergence paths of DSA and EXTRA is more substantial when the graph has a large condition number κ_g. The stepsize α for DSA and EXTRA in all the considered cases is hand-optimized and the results for the best choice of α are reported.

is smaller.

We also compare the performances of DSA and EXTRA over different topologies to verify the claim that DSA is more efficient than EXTRA in terms of number of gradient evaluations over different network topologies. The parameters are as in Fig. 7.3 and the stepsize α for EXTRA in different topologies are optimized separately. In particular, the best stepsize for the complete graph, random graph with $p_c = 0.35$, random graph with $p_c = 0.25$, and line are $\alpha = 6 \times 10^{-2}$, $\alpha = 5 \times 10^{-2}$, $\alpha = 3 \times 10^{-2}$, and $\alpha = 5 \times 10^{-2}$, respectively. Fig. 7.4 shows the convergence paths of DSA and EXTRA versus number of gradient evaluations for four different network topologies. We observe that in the considered graphs, DSA achieves a target accuracy $\|\mathbf{w}^t - \mathbf{w}^*\|^2$ faster than EXTRA. In other words, to achieve a specific accuracy $\|\mathbf{w}^t - \mathbf{w}^*\|^2$ DSA requires less number of local gradient evaluations relative to EXTRA. In addition, the gap between the performance of DSA and EXTRA is more substantial when the graph condition number κ_g is larger. In particular, in the case

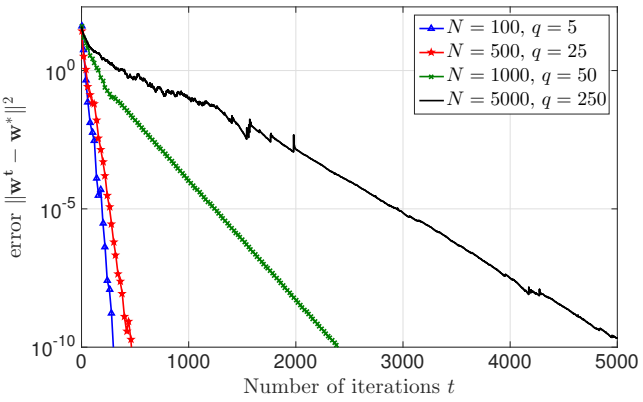


Figure 7.5: Comparison of convergence paths of DSA for different number of samples N when the network size is $V = 20$ and the graph is randomly generated with the connectivity ratio $p_c = 0.35$. Convergence time for DSA increases by increasing the total number of sample points N which is equivalent to increasing the number of samples at each node $q = N/V$.

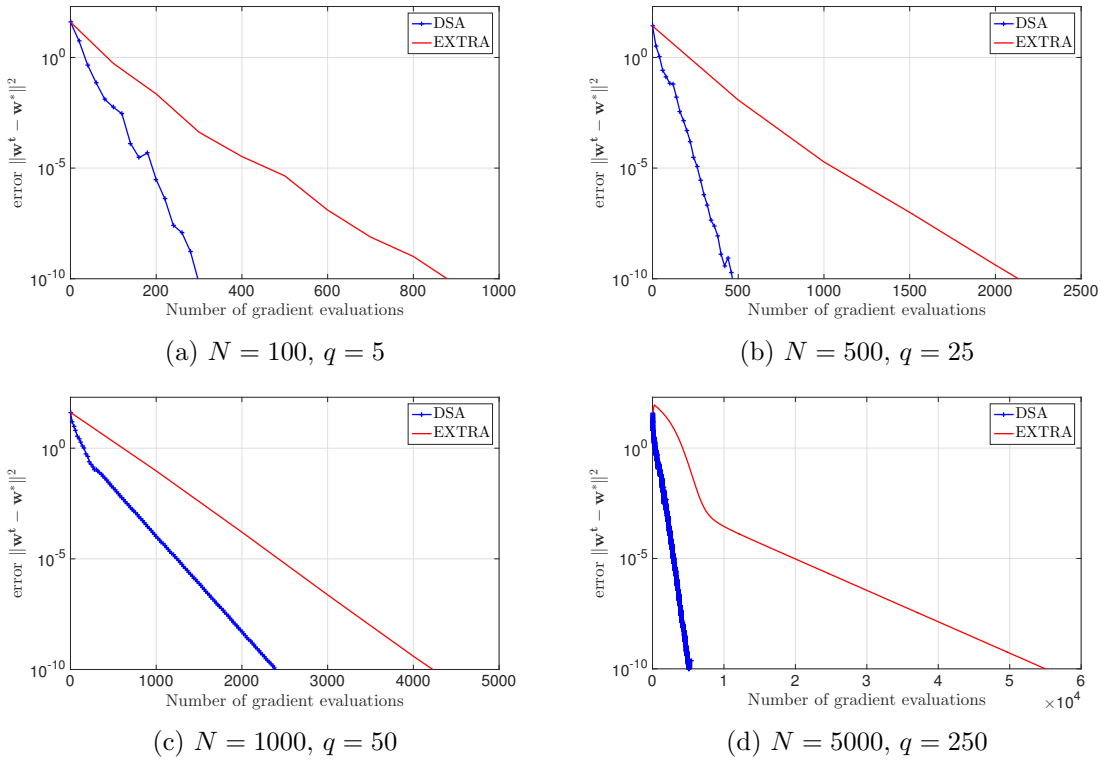
that we have a complete graph, which has a small graph condition number, the difference between the convergence paths of DSA and EXTRA is less significant comparing to the line graph which has a large graph condition number.

7.4.3 Effect of number of functions (samples) at each node q

To evaluate performance for different number of functions (sample points) available at each node which is indicated by q, we use the same setting as in Fig. 7.2; however, we consider scenarios with different number of samples N which leads to different number of samples at each node q. To be more precise, we fix the total number of nodes in the network as $V = 20$ and we consider the cases that the total number of samples are $N = 100$, $N = 500$, $N = 1000$, and $N = 5000$ where the corresponding number of samples at each node are $q = 5$, $q = 25$, $q = 50$, and $q = 250$, respectively. Similar to the experiment in Fig. 7.2, the graph is generated randomly with connectivity ratio $p_c = 0.35$.

For each of these scenarios the DSA stepsize α is hand-optimized and the best choice is used for comparison with others. The results are reported for $\alpha = 10^{-4}$, $\alpha = 10^{-3}$, $\alpha = 5 \times 10^{-3}$, and $\alpha = 10^{-1}$ when the total number of samples are $N = 5000$, $N = 1000$, $N = 500$, $N = 100$, respectively. The resulting convergence paths are shown in Fig. 7.5.

The convergence paths in Fig. 7.5 show that as we increase the total number of samples N and consequently the number of assigned samples to each node q, we observe that DSA converges slower to the optimal argument. This conclusion is expected from the theoretical result in (7.99) which shows that the linear convergence rate of DSA becomes slower by increasing q. In particular, to achieve the target accuracy of $\|\mathbf{w}^t - \mathbf{w}^*\|^2 = 10^{-8}$ DSA



(a) $N = 100$, $q = 5$ (b) $N = 500$, $q = 25$

(c) $N = 1000$, $q = 50$ (d) $N = 5000$, $q = 250$

Figure 7.6: Convergence paths of DSA and EXTRA for the cases that $(N = 100, q = 5)$, $(N = 500, q = 25)$, $(N = 1000, q = 50)$, and $(N = 5000, q = 250)$ are presented. Distance to optimality $e^t = \|\mathbf{w}^t - \mathbf{w}^*\|^2$ is shown with respect to number of gradient evaluations. The total number of nodes in the network is fixed and equal to $V = 20$ and the graph is randomly generated with the connectivity ratio $p_c = 0.35$. DSA converges faster relative to EXTRA and they both converge slower when the total number of samples N increases.

requires $t = 260$, $t = 380$, $t = 1960$, and $t = 4218$ iterations (or equivalently gradient evaluations) for the cases that $q = 5$, $q = 25$, $q = 50$, $q = 250$, respectively.

To have a more comprehensive comparison of DSA and EXTRA, we also compare their performances under the four different settings considered in Fig. 7.5. The convergence paths of these methods in terms of number of gradient evaluations for $(N = 100, q = 5)$, $(N = 500, q = 25)$, $(N = 1000, q = 50)$, and $(N = 5000, q = 250)$ are presented in Fig 7.6. The optimal stepsizes for EXTRA in the considered settings are $\alpha = 4 \times 10^{-1}$, $\alpha = 5 \times 10^{-2}$, $\alpha = 3 \times 10^{-2}$, and $\alpha = \times 10^{-2}$, respectively. An interesting observation is the effect of q on the convergence rate of EXTRA. We observe that EXTRA converges slower as the number of samples at each node q increases which is identical to the observation for DSA in Fig. 7.5. Moreover, for all of the settings considered in Fig. 7.6, DSA outperforms EXTRA in terms of number of required gradient evaluations until convergence. Moreover, by increasing the total number of samples N and subsequently the number of assigned samples to each node q the

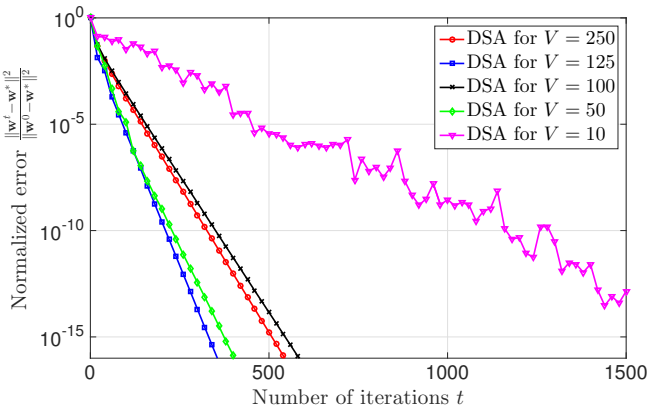


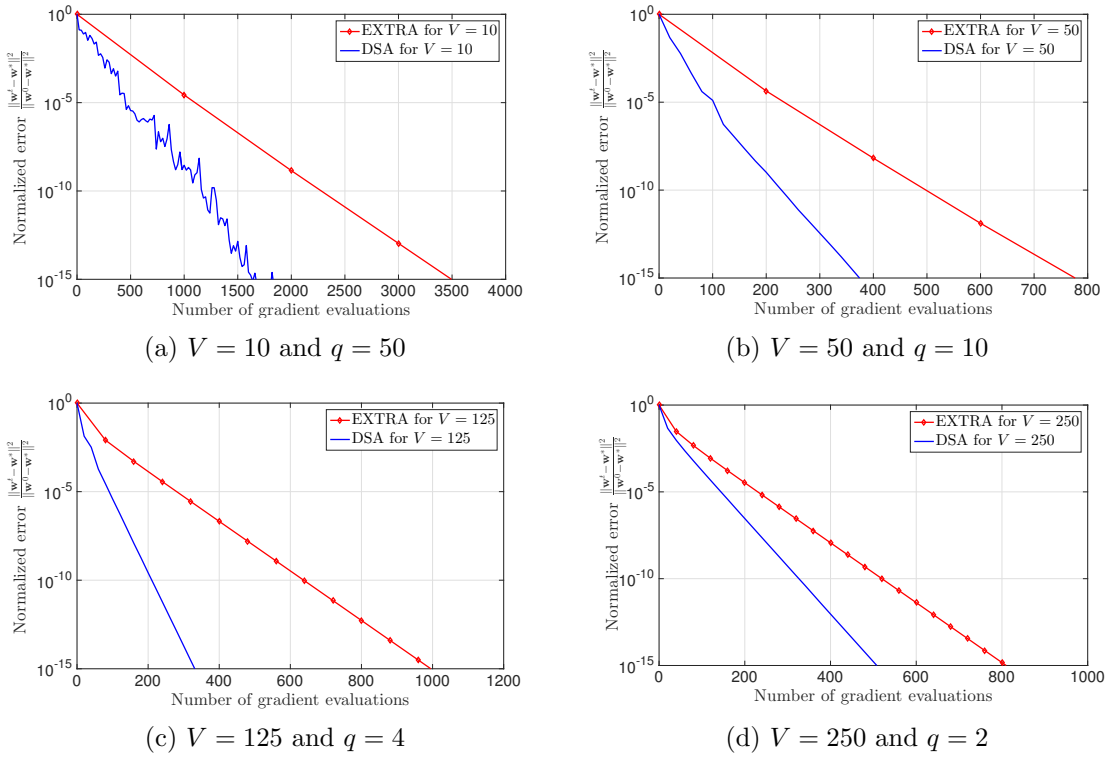
Figure 7.7: Normalized error $\|\mathbf{w}^t - \mathbf{w}^*\|^2/\|\mathbf{w}^0 - \mathbf{w}^*\|^2$ of DSA versus number of iterations t for networks with different number of nodes V when the total number of samples is fixed $N = 500$. The graphs are randomly generated with the connectivity ratio $p_c = 0.35$. Picking a very small or large value for V which leads to a very large or small value for q, respectively, is not preferable. The best performance belongs to the case that $V = 125$ and $q = 4$.

advantage of DSA with respect to EXTRA in terms of computational complexity becomes more significant. This observation justifies the use of DSA for large-scale optimization problems as we consider in Section 7.4.5.

7.4.4 Effect of number of nodes V

In some settings, we can choose the number of nodes (processors) V for training the dataset. In this section, we study the effect of network size V on the convergence path of DSA when a fixed number of samples N is given to train the classifier $\mathbf{w}$. Notice that when N is fixed, by changing the number of nodes V, the number of assigned samples to each node $q = N/V$ changes proportionally. Then, we may want to pick the number of nodes V or equivalently the number of assigned samples to each node q which leads to the best performance of DSA for training N samples. Hence, we fix the total number of sample points as $N = 500$ and assign the same amount of sample points q to each node. We consider 5 different settings with $V = 10$, $V = 50$, $V = 100$, $V = 125$, and $V = 250$ which their corresponding number of assigned samples to each node are $q = 50$, $q = 10$, $q = 5$, $q = 4$, and $q = 2$, respectively. The DSA stepsize for each of the considered settings is hand-optimized. The stepsizes $\alpha = 5 \times 10^{-3}$, $\alpha = 2 \times 10^{-2}$, $\alpha = 6 \times 10^{-2}$, and $\alpha = 8 \times 10^{-2}$ are considered for the cases that the number of assigned samples to each node are $q = 50$, $q = 10$, $q = 5$, $q = 4$, and $q = 2$, respectively.

Fig. 7.7 shows the convergence paths of DSA for networks with different number of nodes. Notice that the normalized error $\tilde{e}^t = \|\mathbf{w}^t - \mathbf{w}^*\|^2/\|\mathbf{w}^0 - \mathbf{w}^*\|^2$ is reported, since the dimension of the vector $\mathbf{w}$ is different for different choices of V. Comparison of the



(a) $V = 10$ and $q = 50$

(b) $V = 50$ and $q = 10$

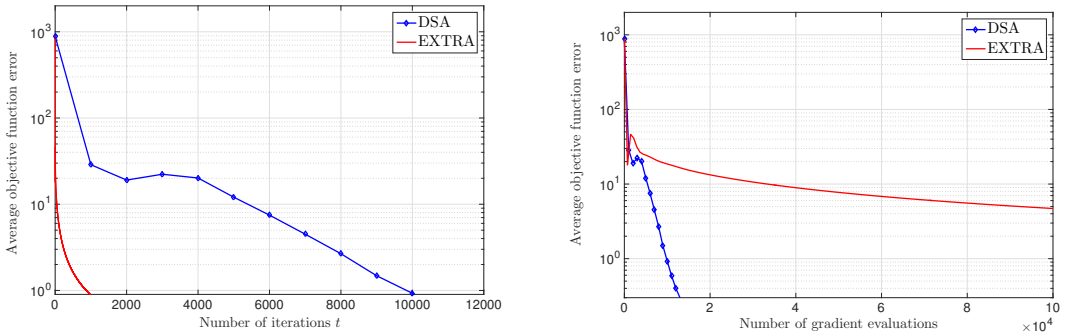
(c) $V = 125$ and $q = 4$

(d) $V = 250$ and $q = 2$

Figure 7.8: Convergence paths of DSA and EXTRA for different number of nodes V when the total number of sample points is fixed as $N = 500$. The graphs are randomly generated with the connectivity ratio $p_c = 0.35$. Normalized distance to optimality $\tilde{e}^t = \|\mathbf{w}^t - \mathbf{w}^*\|^2/\|\mathbf{w}^0 - \mathbf{w}^*\|^2$ is shown with respect to number of gradient evaluations. DSA converges faster relative to EXTRA in all of the considered settings.

convergence paths in Fig. 7.7 shows that the best performance belongs to the case that $N = 125$ and each node has access to $q = 4$ sample points. The performance of DSA becomes worse for the case that there are $V = 5$ nodes in the network and each node has $q = 100$ sample points. This observation implies that the DSA algorithm is also preferable to SAGA which corresponds to the case that $V = 1$. Moreover, we observe that when the number of nodes is large as $V = 250$ and each node has access to $q = 2$ samples, DSA doesn't perform well. Thus, increasing the size of network V doesn't always lead to a better performance for DSA. The best performance is observed when a moderate subset of the samples is assigned to each node.

We also study the convergence rates of DSA and EXTRA in terms of number of gradient evaluations for networks with different number of nodes V. Fig. 7.8 demonstrates the convergence paths of DSA and EXTRA for the cases that $V = 10$, $V = 50$, $V = 125$, and $V = 250$. Similar to DSA, we report the best performance of EXTRA for each setting which is achieved by the stepsizes $\alpha = 5 \times 10^{-2}$, $\alpha = 8 \times 10^{-2}$, $\alpha = 8 \times 10^{-2}$, and $\alpha = 10^{-1}$ for $V = 10$, $V = 50$, $V = 125$, and $V = 250$, respectively. Observe that in all settings DSA



(a) (b)

Figure 7.9: Convergence paths of DSA and EXTRA for the protein homology classification problem with $N = 1.45 \times 10^5$ samples. The graph has $N = 200$ nodes and it is randomly generated with the connectivity ratio $p_c = 0.35$. The average objective function error is shown with respect to number of iterations t and number of gradient evaluations, respectively.

is more efficient relative to EXTRA and it requires less number of gradient evaluations for convergence.

7.4.5 Large-scale classification application

In this section we solve the logistic regression problem in (7.100) for the protein homology dataset provided in KDD Cup 2004. The dataset contains $N = 1.45 \times 10^5$ sample points and each sample point has $p = 74$ features. We consider the case that the sample points are distributed over $V = 200$ nodes which implies that each node has access to $q = 725$ samples. We set the connectivity ratio $p_c = 0.35$ and hand optimize the stepsize α for DSA and EXTRA separately. The best performance of DSA is observed for $\alpha = 2 \times 10^{-7}$ and the best choice of stepize for EXTRA is $\alpha = 6 \times 10^{-7}$. We capture the error in terms of the average objective function error e_{avg}^t of the network which is defined as

$$e_{avg}^t := \frac{1}{V} \sum_{u=1}^{V} \left[\sum_{v=1}^{V} f_v(\mathbf{w}_u^t) - \sum_{v=1}^{V} f_v(\mathbf{w}^*) \right]. \tag{7.105}$$

Note that the difference $\sum_{v=1}^{V} f_v(\mathbf{w}_u^t) - \sum_{v=1}^{V} f_v(\mathbf{w}^*)$ shows the objective function error associated with the decision variable of node u at time t. Thus, the expression in (7.105) indicates the average objective function error of the network at step t.

The average objective function error for DSA and EXTRA in terms of number of iterations t and number of gradient evaluations are presented in Fig. 7.9(a) and Fig. 7.9(b), respectively. As we observe, the results in Fig. 7.9 for the large-scale classification problem match the observations in Fig. 7.2 for the classification problem with a synthetic dataset.

In particular, both algorithms converge linearly, while EXTRA converges faster than DSA in terms of number of iterations or equivalently in terms of communication cost. On the other hand, DSA outperforms EXTRA in terms of computational complexity or number of required gradients to reach a target accuracy. Moreover, notice that the difference between the performances of DSA and EXTRA in terms of number of gradient evaluations is more significant in Fig. 7.9(b) relative to the one in Fig. 7.2(b). Thus, by increasing the problem dimension we obtain more computational complexity benefit by using DSA instead of EXTRA.

Part III

Adaptive Sample Size Methods

Chapter 8

First-order adaptive sample size methods

In the first part of this thesis we presented stochastic quasi-Newton methods for solving ERM problems. These algorithms, as in other stochastic methods, split samples across time to reduce the computational complexity of deterministic methods and approximate curvature of the objective function to accelerate convergence of first-order methods.

In the second part of the thesis we considered the use of decentralized methods for ERM. In this class of algorithms, samples are divided among nodes (processors) of a network and each node only operates on a subset of samples. Indeed, distributing sampling over processors, i.e., splitting across space, is computationally more efficient than operating on a single processor.

In this part of the thesis, we introduce a novel approach for solving large-scale ERM problems via a nested collection of subsets that grows geometrically. In this approach, which is called adaptive sample size, instead of distributing samples across time or space, we operate on a subset of samples at each stage and geometrically increase the size of the training set. The key insight is that the optimal argument associated with a training subset of a certain size is not that far from the optimal argument associated with a larger training subset, since the samples are drawn from a common (unknown) distribution. This means that solutions for an element of the geometric sequence can be used as warm starts for the solution of the subsequent element. We explain adaptive sample size methods in detail in the following chapters.

8.1 Context and background

Finite sum minimization (FSM) problems involve objectives that are expressed as the sum of a typically large number of component functions. Since evaluating descent directions

is costly, it is customary to utilize stochastic descent methods that access only one of the functions at each iteration. When considering first order methods, a fitting measure of complexity is the total number of gradient evaluations that are needed to achieve optimality of order ϵ. The paradigmatic deterministic gradient descent (GD) method serves as a naive complexity upper bound and has long been known to obtain an ϵ-suboptimal solution with $\mathcal{O}(N\kappa \log(1/\epsilon))$ gradient evaluations for an FSM problem with N component functions and condition number κ [84]. Accelerated gradient descent (AGD) [85] improves the computational complexity of GD to $\mathcal{O}(N\sqrt{\kappa}\log(1/\epsilon))$, which is known to be the optimal bound for deterministic first-order methods [84]. In terms of stochastic optimization, it has been only recently that linearly convergent methods have been proposed. Stochastic averaging gradient [31, 49], stochastic variance reduction [45], and dual coordinate descent [109, 110], have all been shown to converge to ϵ-accuracy at a cost of $\mathcal{O}((N+\kappa)\log(1/\epsilon))$ gradient evaluations. The accelerating catalyst framework in [52] further reduces complexity to $\mathcal{O}((N+\sqrt{N\kappa})\log(\kappa)\log(1/\epsilon))$ and the works in [1] and [30] to $\mathcal{O}((N+\sqrt{N\kappa})\log(1/\epsilon))$. The latter matches the upper bound on the complexity of stochastic methods [124].

Perhaps the main motivation for studying FSM is the solution of empirical risk minimization (ERM) problems associated with a large training set. ERM problems are particular cases of FSM, but they do have two specific qualities that come from the fact that ERM is a proxy for statistical loss minimization. The first property is that since the empirical risk and the statistical loss have different minimizers, there is no reason to solve ERM beyond the expected difference between the two objectives. This so-called *statistical accuracy* takes the place of ϵ in the complexity orders of the previous paragraph and is a constant of order $\mathcal{O}(1/N^\alpha)$ where α is a constant from the interval $[0.5, 1]$ depending on the regularity of the loss function; see Section **??**. The second important property of ERM is that the component functions are drawn from a common distribution. This implies that if we consider subsets of the training set, the respective empirical risk functions are not that different from each other and, indeed, their differences are related to the statistical accuracy of the subset.

The relationship of ERM to statistical loss minimization suggests that ERM problems have more structure than FSM problems. This is not exploited by most existing methods which, albeit used for ERM, are in fact designed for FSM. The goal of this paper is to exploit the relationship between ERM and statistical loss minimization to achieve lower overall computational complexity for a broad class of first-order methods applied to ERM. The technique we propose uses subsamples of the training set containing $n \leq N$ component functions that we grow geometrically. In particular, we start by a small number of samples and minimize the corresponding empirical risk added by a regularization term of order V_n up to its statistical accuracy. Note that, based on the first property of ERM, the added adaptive regularization term does not modify the required accuracy while it makes the

problem strongly convex and improves the problem condition number. After solving the subproblem, we double the size of the training set and use the solution of the problem with n samples as a warm start for the problem with $2n$ samples. This is a reasonable initialization since based on the second property of ERM the functions are drawn from a joint distribution, and, therefore, the optimal values of the ERM problems with n and $2n$ functions are not that different from each other. The proposed approach succeeds in exploiting the two properties of ERM problems to improve complexity bounds of first-order methods. In particular, we show that to reach the statistical accuracy of the full training set the adaptive sample size scheme reduces the overall computational complexity of a broad range of first-order methods by a factor of $\log(N^\alpha)$. For instance, the overall computational complexity of adaptive sample size AGD to reach the statistical accuracy of the full training set is of order $\mathcal{O}(N\sqrt{\kappa})$ which is lower than $\mathcal{O}((N\sqrt{\kappa})\log(N^\alpha))$ complexity of AGD.

Related work. The adaptive sample size approach was used in [29] to improve the performance of the SAGA method [31] for solving ERM problems. In the dynamic SAGA (DynaSAGA) method in [29], the size of training set grows at each iteration by adding two new samples, and the iterates are updated by a single step of SAGA. Although DynaSAGA succeeds in improving the performance of SAGA for solving ERM problems, it does not use an adaptive regularization term to tune the problem condition number. Moreover, DynaSAGA only works for strongly convex functions, while in our proposed scheme the functions are convex (not necessarily strongly convex).

8.2 Problem formulation

Consider a decision vector $\mathbf{w} \in \mathbb{R}^p$, a random variable $\boldsymbol{\Theta}$ with realizations $\boldsymbol{\theta}$ and a convex loss function $f(\mathbf{w}; \boldsymbol{\theta})$. We aim to find the optimal argument that minimizes the optimization problem

$$\mathbf{w}^* := \operatorname*{argmin}_{\mathbf{w}} L(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \mathbb{E}_{\boldsymbol{\Theta}}[f(\mathbf{w}, \boldsymbol{\Theta})] = \operatorname*{argmin}_{\mathbf{w}} \int_{\boldsymbol{\Theta}} f(\mathbf{w}, \boldsymbol{\Theta}) P(d\boldsymbol{\theta}), \tag{8.1}$$

where $L(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\Theta}}[f(\mathbf{w}, \boldsymbol{\Theta})]$ is defined as the expected loss, and P is the probability distribution of the random variable $\boldsymbol{\Theta}$. The optimization problem in (8.1) cannot be solved since the distribution P is unknown. However, we have access to a training set $\mathcal{T} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$ containing N independent samples $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ drawn from P, and, therefore, we attempt to minimize the empirical loss associated with the training set $\mathcal{T} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$, which is equivalent to minimizing the problem

$$\mathbf{w}_n^\dagger := \operatorname*{argmin}_{\mathbf{w}} L_n(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{w}, \boldsymbol{\theta}_i), \tag{8.2}$$

for $n = N$. Note that in (8.2) we defined $L_n(\mathbf{w}) := (1/n)\sum_{i=1}^{n} f(\mathbf{w}, \boldsymbol{\theta}_i)$ as the empirical loss.

There is a rich literature on bounds for the difference between the expected loss L and the empirical loss L_n which is also referred to as *estimation error* [15, 16]. We assume here that there exists a constant V_n, which depends on the number of samples n, that upper bounds the difference between the expected and empirical losses for all $\mathbf{w} \in \mathbb{R}^p$

$$\mathbb{E}\left[\sup_{\mathbf{w}\in\mathbb{R}^p} |L(\mathbf{w}) - L_n(\mathbf{w})|\right] \leq V_n, \tag{8.3}$$

where the expectation is with respect to the choice of the training set. The celebrated work of Vapnik in [121, Section 3.4] provides the upper bound $V_n = \mathcal{O}(\sqrt{(1/n)\log(1/n)})$ which can be improved to $V_n = \mathcal{O}(\sqrt{1/n})$ using the chaining technique (see, e.g., [17]). Bounds of the order $V_n = O(1/n)$ have been derived more recently under stronger regularity conditions that are not uncommon in practice, [4, 16, 36]. In this paper, we report our results using the general bound $V_n = O(1/n^{\alpha})$ where α can be any constant form the interval $[0.5, 1]$.

The observation that the optimal values of the expected loss and empirical loss are within a V_n distance of each other implies that there is no gain in improving the optimization error of minimizing L_n beyond the constant V_n. In other words, if we find an approximate solution $\mathbf{w}_n$ such that the optimization error is bounded by $L_n(\mathbf{w}_n) - L_n(\mathbf{w}_n^{\dagger}) \leq V_n$, then finding a more accurate solution to reduce the optimization error is not beneficial since the overall error, i.e., the sum of estimation and optimization errors, does not become smaller than V_n. Throughout the paper we say that $\mathbf{w}_n$ solves the ERM problem in (8.2) to within its statistical accuracy if it satisfies $L_n(\mathbf{w}_n) - L_n(\mathbf{w}_n^{\dagger}) \leq V_n$.

We can further leverage the estimation error to add a regularization term of the form $(cV_n/2)\|\mathbf{w}\|^2$ to the empirical loss to ensure that the problem is strongly convex. To do so, we define the regularized empirical risk $R_n(\mathbf{w}) := L_n(\mathbf{w}) + (cV_n/2)\|\mathbf{w}\|^2$ and the corresponding optimal argument

$$\mathbf{w}_n^* := \operatorname*{argmin}_{\mathbf{w}} R_n(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} L_n(\mathbf{w}) + \frac{cV_n}{2}\|\mathbf{w}\|^2, \tag{8.4}$$

and attempt to minimize R_n with accuracy V_n. Since the regularization in (8.4) is of order V_n and (8.3) holds, the difference between $R_n(\mathbf{w}_n^*)$ and $L(\mathbf{w}^*)$ is also of order V_n – this is not immediate as it seems; see [106]. Thus, the variable $\mathbf{w}_n$ solves the ERM problem in (8.2) to within its statistical accuracy if it satisfies $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$. It follows that by solving the problem in (8.4) for $n = N$ we find $\mathbf{w}_N^*$ that solves the expected risk minimization in (**??**) up to the statistical accuracy V_N of the full training set $\mathcal{T}$. In the following section we introduce a class of methods that solve problem (8.4) up to its statistical accuracy faster than traditional deterministic and stochastic descent methods.

8.3 Adaptive sample size methods

The empirical risk minimization (ERM) problem in (8.4) can be solved using state-of-the-art methods for minimizing strongly convex functions. However, these methods never exploit the particular property of ERM that the functions are drawn from the same distribution. In this section, we propose an *adaptive sample size* scheme which exploits this property of ERM to improve the convergence guarantees for traditional optimization method to reach the statistical accuracy of the full training set. In the proposed adaptive sample size scheme, we start by a small number of samples and solve its corresponding ERM problem with a specific accuracy. Then, we double the size of the training set and use the solution of the previous ERM problem – with half samples – as a warm start for the new ERM problem. This procedure keeps going until the training set becomes identical to the given training set $\mathcal{T}$ which contains N samples.

Consider the training set $\mathcal{S}_m$ with m samples as a subset of the full training $\mathcal{T}$, i.e., $\mathcal{S}_m \subset \mathcal{T}$. Assume that we have solved the ERM problem corresponding to the set $\mathcal{S}_m$ such that the approximate solution $\mathbf{w}_m$ satisfies the condition $\mathbb{E}[R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*)] \leq \delta_m$. Now the next step in the proposed adaptive sample size scheme is to double the size of the current training set $\mathcal{S}_m$ and solve the ERM problem corresponding to the set $\mathcal{S}_n$ which has $n = 2m$ samples and contains the previous set, i.e., $\mathcal{S}_m \subset \mathcal{S}_n \subset \mathcal{T}$.

We use $\mathbf{w}_m$ which is a proper approximate for the optimal solution of R_m as the initial iterate for the optimization method that we use to minimize the risk R_n. This is a reasonable choice if the optimal arguments of R_m and R_n are close to each other, which is the case since samples are drawn from a fixed distribution $\mathcal{P}$. Starting with $\mathbf{w}_m$, we can use first-order descent methods to minimize the empirical risk R_n. Depending on the iterative method that we use for solving each ERM problem we might need different number of iterations to find an approximate solution $\mathbf{w}_n$ which satisfies the condition $\mathbb{E}[R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)] \leq \delta_n$. To design a comprehensive routine we need to come up with a proper condition for the required accuracy δ_n at each phase.

In the following proposition we derive an upper bound for the expected suboptimality of the variable $\mathbf{w}_m$ for the risk R_n based on the accuracy of $\mathbf{w}_m$ for the previous risk R_m associated with the training set $\mathcal{S}_m$. This upper bound allows us to choose the accuracy δ_m efficiently.

Proposition 9 *Consider the sets $\mathcal{S}_m$ and $\mathcal{S}_n$ as subsets of the training set $\mathcal{T}$ such that $\S_m \subset \mathcal{S}_n \subset \mathcal{T}$, where the number of samples in the sets $\mathcal{S}_m$ and $\mathcal{S}_n$ are m and n, respectively. Further, define $\mathbf{w}_m$ as an δ_m optimal solution of the risk R_m in expectation, i.e., $\mathbb{E}[R_m(\mathbf{w}_m) - R_m^*] \leq \delta_m$, and recall V_n as the statistical accuracy of the training set $\mathcal{S}_n$. Then the empirical risk error $R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)$ of the variable $\mathbf{w}_m$ corresponding to the*

Algorithm 10 Adaptive Sample Size Mechanism

Input: Initial sample size $n = m_0$ and argument $\mathbf{w}_n = \mathbf{w}_{m_0}$ with $\|\nabla R_n(\mathbf{w}_n)\| \leq (\sqrt{2c})V_n$
while $n \leq N$ **do**
    Update argument and index: $\mathbf{w}_m = \mathbf{w}_n$ and $m = n$.
    Increase sample size: $n = \min\{2m, N\}$.
    Set the initial variable: $\tilde{\mathbf{w}} = \mathbf{w}_m$.
    while $\|\nabla R_n(\tilde{\mathbf{w}})\| > (\sqrt{2c})V_n$ **do**
        Update the variable $\tilde{\mathbf{w}}$: Compute $\tilde{\mathbf{w}} = \text{Update}(\tilde{\mathbf{w}}, \nabla R_n(\tilde{\mathbf{w}}))$
    end while
    Set $\mathbf{w}_n = \tilde{\mathbf{w}}$.
end while

set $\S_n$ in expectation is bounded above by

$$\mathbb{E}[R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)] \leq \delta_m + \frac{2(n-m)}{n}\left(V_{n-m} + V_m\right) + 2\left(V_m - V_n\right) + \frac{c(V_m - V_n)}{2}\|\mathbf{w}^*\|^2. \tag{8.5}$$

Proof: See Appendix C.1. ■

The result in Proposition 9 characterizes the sub-optimality of the variable $\mathbf{w}_m$, which is an δ_m sub-optimal solution for the risk R_m, with respect to the empirical risk R_n associated with the set $\S_n$. If we assume that the statistical accuracy V_n is of the order $\mathcal{O}(1/n^\alpha)$ and we double the size of the training set at each step, i.e., $n = 2m$, then the inequality in (9.29) can be simplified to

$$\mathbb{E}[R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)] \leq \delta_m + \left[2 + \left(1 - \frac{1}{2^\alpha}\right)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\right] V_m. \tag{8.6}$$

The expression in (8.6) formalizes the reason that there is no need to solve the sub-problem R_m beyond its statistical accuracy V_m. In other words, even if δ_m is zero the expected sub-optimality will be of the order $\mathcal{O}(V_m)$, i.e., $\mathbb{E}[R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)] = \mathcal{O}(V_m)$. Based on this observation, The required precision δ_m for solving the sub-problem R_m should be of the order $\delta_m = \mathcal{O}(V_m)$.

The steps of the proposed adaptive sample size scheme is summarized in Algorithm 10. Note that since computation of the sub-optimality $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ requires access to the minimizer $\mathbf{w}_n^*$, we replace the condition $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$ by a bound on the norm of gradient $\|\nabla R_n(\mathbf{w}_n)\|^2$. The risk R_n is strongly convex, and we can bound the suboptimality $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ as

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq \frac{1}{2cV_n}\|\nabla R_n(\mathbf{w}_n)\|^2. \tag{8.7}$$

Hence, at each stage, we stop updating the variable if the condition $\|\nabla R_n(\mathbf{w}_n)\| \leq (\sqrt{2c})V_n$ holds which implies $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$. The intermediate variable $\tilde{\mathbf{w}}$ can be updated in Step 7 using any first-order method. We will discuss this procedure for accelerated gradient descent (AGD) and stochastic variance reduced gradient (SVRG) methods in Sections 8.4.1 and 8.4.2, respectively.

8.4 Complexity analysis

In this section, we aim to characterize the number of required iterations s_n at each stage to solve the subproblems within their statistical accuracy. We derive this result for all linearly convergent first-order deterministic and stochastic methods.

The inequality in (8.6) not only leads to an efficient policy for the required precision δ_m at each step, but also provides an upper bound for the sub-optimality of the initial iterate, i.e., $\mathbf{w}_m$, for minimizing the risk R_n. Using this upper bound, depending on the iterative method of choice, we can characterize the number of required iterations s_n to ensure that the updated variable is within the statistical accuracy of the risk R_n. To formally characterize the number of required iterations s_n, we first assume the following conditions are satisfied.

Assumption 17 *The loss functions $f(\mathbf{w}, \boldsymbol{\theta})$ are convex with respect to $\mathbf{w}$ for all values of $\boldsymbol{\theta}$. Moreover, their gradients $\nabla f(\mathbf{w}, \boldsymbol{\theta})$ are Lipschitz continuous with constant M*

$$\|\nabla f(\mathbf{w}, \boldsymbol{\theta}) - \nabla f(\mathbf{w}', \boldsymbol{\theta})\| \leq M\|\mathbf{w} - \mathbf{w}'\|, \qquad \textit{for all } \boldsymbol{\theta}. \tag{8.8}$$

The conditions in Assumption 17 imply that the average loss $L(\mathbf{w})$ and the empirical loss $L_n(\mathbf{w})$ are convex and their gradients are Lipschitz continuous with constant M. Thus, the empirical risk $R_n(\mathbf{w})$ is strongly convex with constant cV_n and its gradients $\nabla R_n(\mathbf{w})$ are Lipschitz continuous with parameter $M + cV_n$.

So far we have concluded that each subproblem should be solved up to its statistical accuracy. This observation leads to an upper bound for the number of iterations needed at each step to solve each subproblem. Indeed various descent methods can be executed for solving the sub-problem. Here we intend to come up with a general result that contains all descent methods that have a linear convergence rate when the objective function is strongly convex and smooth. In the following theorem, we derive a lower bound for the number of required iterations s_n to ensure that the variable $\mathbf{w}_n$, which is the outcome of updating $\mathbf{w}_m$ by s_n iterations of the method of interest, is within the statistical accuracy of the risk R_n for any linearly convergent method.

Theorem 13 *Consider the variable $\mathbf{w}_m$ as a V_m-suboptimal solution of the risk R_m in expectation, i.e., $\mathbb{E}[R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*)] \leq V_m$, where $V_m = \mathcal{O}(1/m^\alpha)$. Consider the sets*

$\mathcal{S}_m \subset \mathcal{S}_n \subset \mathcal{T}$ such that $n = 2m$, and suppose Assumption 17 holds. Further, define $0 \leq \rho_n < 1$ as the linear convergence factor of the descent method used for updating the iterates. Then, the variable $\mathbf{w}_n$ generated based on the adaptive sample size mechanism satisfies $\mathbb{E}[R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)] \leq V_n$ if the number of iterations s_n at the n-th stage is larger than

$$s_n \;\geq\; -\frac{\log\left[3 \times 2^{\alpha} + (2^{\alpha} - 1)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\right]}{\log \rho_n}. \tag{8.9}$$

Proof: According to the result in Proposition 13 and the condition that $\mathbb{E}[R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*)] \leq V_m$, we obtain that

$$\mathbb{E}[R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)] \leq \left[3 + \left(1 - \frac{1}{2^{\alpha}}\right)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\right] V_m. \tag{8.10}$$

If we assume that the first-order descent method that we use to update the iterates has a linear convergence rate, then there exists a constant $0 < \rho_n < 1$ we obtain that after s_n iterations the error is bounded above by

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq \rho_n^{s_n}(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)). \tag{8.11}$$

The result in (8.11) holds for deterministic methods. If we use a stochastic linearly convergent method such as SVRG, then the result holds in expectation and we can write

$$\mathbb{E}[R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)] \leq \rho^{s_n}(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)), \tag{8.12}$$

where the expectation is with respect to the index of randomly chosen functions.

It follows form computing the expected value of both sides in (8.11) with respect to the choice of training sets and using the upper bound in (8.10) for the expected difference $\mathbb{E}[R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)]$ that

$$\mathbb{E}[R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)] \leq \rho^{s_n}\left[3 + \left(1 - \frac{1}{2^{\alpha}}\right)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\right] V_m. \tag{8.13}$$

Note that the inequality in (8.13) also holds for stochastic methods. The difference is in stochastic methods the expectation is with respect to the choice of training sets and the index of random functions, while for deterministic methods it is only with respect to the choice of training sets.

To ensure that the suboptimality $\mathbb{E}[R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)]$ is smaller than V_n we need to guarantee that the right hand side in (8.13) is not larger than V_n, which is equivalent to the condition

$$\rho^{s_n}\left[3 + \left(1 - \frac{1}{2^{\alpha}}\right)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\right] \leq \frac{1}{2^{\alpha}}. \tag{8.14}$$

By regrouping the terms in (8.14) we obtain that

$$s_n \geq -\frac{\log\left[3 \times 2^{\alpha} + (2^{\alpha} - 1)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\right]}{\log(\rho_n)}, \tag{8.15}$$

and the claim in (8.9) follows. ■

The result in Theorem 13 characterizes the number of required iterations at each phase. Depending on the linear convergence factor ρ_n and the parameter α for the order of statistical accuracy, the number of required iterations might be different. Note that the parameter ρ_n might depend on the size of the training set directly or through the dependency of the problem condition number on n. It is worth mentioning that the result in (8.9) shows a lower bound for the number of required iteration which means that $s_n = \lfloor -(\log\left[3 \times 2^{\alpha} + (2^{\alpha} - 1)\left(2 + (c/2)\|\mathbf{w}^*\|^2\right)\right]/\log \rho_n)\rfloor + 1$ is the exact number of iterations needed when minimizing R_n, where $\lfloor a \rfloor$ indicates the floor of a. To characterize the overall computational complexity of the proposed adaptive sample size scheme, the exact expression for the linear convergence constant ρ_n is required. In the following section, we focus on two deterministic and stochastic methods and characterize their overall computational complexity to reach the statistical accuracy of the full training set $\mathcal{T}$.

8.4.1 Adaptive sample size accelerated gradient (Ada AGD)

The accelerated gradient descent (AGD) method, also called as Nesterov's method, is a long-established descent method which achieves the optimal convergence rate for first-order deterministic methods. In this section, we aim to combine the update of AGD with the adaptive sample size scheme in Section 8.3 to improve convergence guarantees of AGD for solving ERM problems. This can be done by using AGD for updating the iterates in step 7 of Algorithm 10. Given an iterate $\mathbf{w}_m$ within the statistical accuracy of the set $\mathcal{S}_m$, the adaptive sample size accelerated gradient descent method (Ada AGD) requires s_n iterations of AGD to ensure that the resulted iterate $\mathbf{w}_n$ lies in the statistical accuracy of $\mathcal{S}_n$. In particular, if we initialize the sequences $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{y}}$ as $\tilde{\mathbf{w}}_0 = \tilde{\mathbf{y}}_0 = \mathbf{w}_m$, the approximate solution $\mathbf{w}_n$ for the risk R_n is the outcome of the updates

$$\tilde{\mathbf{w}}_{k+1} = \tilde{\mathbf{y}}_k - \eta_n \nabla R_n(\tilde{\mathbf{y}}_k), \tag{8.16}$$

and

$$\tilde{\mathbf{y}}_{k+1} = \tilde{\mathbf{w}}_{k+1} + \beta_n(\tilde{\mathbf{w}}_{k+1} - \tilde{\mathbf{w}}_k) \tag{8.17}$$

after s_n iterations, i.e., $\mathbf{w}_n = \tilde{\mathbf{w}}_{s_n}$. The parameters η_n and β_n are indexed by n since they depend on the number of samples. We use the convergence rate of AGD to characterize the number of required iterations s_n to guarantee that the outcome of the recursive updates in

(8.16) and (8.17) is within the statistical accuracy of R_n.

Theorem 14 *Consider the variable $\mathbf{w}_m$ as a V_m-optimal solution of the risk R_m in expectation, i.e., $\mathbb{E}[R_m(\mathbf{w}_m)-R_m(\mathbf{w}_m^*)] \leq V_m$, where $V_m = \gamma/m^\alpha$. Consider the sets $\mathcal{S}_m \subset \mathcal{S}_n \subset \mathcal{T}$ such that $n = 2m$, and suppose Assumption 17 holds. Further, set the parameters η_n and β_n as*

$$\eta_n = \frac{1}{cV_n + M} \qquad \textit{and} \qquad \beta_n = \frac{\sqrt{cV_n + M} - \sqrt{cV_n}}{\sqrt{cV_n + M} + \sqrt{cV_n}}. \tag{8.18}$$

Then, the variable $\mathbf{w}_n$ generated based on the update of Ada AGD in (8.16)-(8.17) satisfies $\mathbb{E}[R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)] \leq V_n$ if the number of iterations s_n is larger than

$$s_n \geq \sqrt{\frac{n^\alpha M + c\gamma}{c\gamma}} \log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right]. \tag{8.19}$$

Moreover, if we define m_0 as the size of the first training set, to reach the statistical accuracy V_N of the full training set $\mathcal{T}$ the overall computational complexity of Ada GD is given by

$$N\left[1 + \log_2\left(\frac{N}{m_0}\right) + \left(\frac{\sqrt{2^\alpha}}{\sqrt{2^\alpha} - 1}\right)\sqrt{\frac{N^\alpha M}{c\gamma}}\right] \log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right]. \tag{8.20}$$

Proof: Note that according to the convergence result for accelerated gradient descent in [84], the sub-optimality of accelerated gradient descent method is linearly convergent with the constant $1 - 1/\sqrt{\kappa}$ where κ is the condition number of the objective function. In particular, the suboptimality after s_n iterations is bounded above by

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq \left(1 - \sqrt{\frac{1}{\kappa}}\right)^{s_n} \left(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*) + \frac{m}{2}\|\mathbf{w}_m - \mathbf{w}_n^*\|^2\right), \tag{8.21}$$

where m is the constant of strong convexity. Replacing $\frac{m}{2}\|\mathbf{w}_m - \mathbf{w}_n^*\|^2$ by its upper bound $R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)$ leads to the expression

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq 2\left(1 - \sqrt{\frac{1}{\kappa}}\right)^{s_n} \left(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)\right), \tag{8.22}$$

Hence, if we follow the steps of the proof of Theorem 2 we obtain that s_n should be larger than

$$s_n \geq -\frac{\log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right]}{\log(1 - 1/\sqrt{\kappa})}. \tag{8.23}$$

According to the inequality $-\log(1-x) > x$, we can replace $-\log(1-1/\sqrt{\kappa})$ by its lower bound $1/\sqrt{\kappa}$ to obtain

$$s_n \geq \sqrt{\kappa_n} \log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right]. \tag{8.24}$$

Note if the condition in (8.24) holds, then the inequality in (8.23) follows. The condition number of the risk R_n is given by $\kappa_n = (M+cV_n)/cV_n$. Further, as stated in the statement of the theorem, V_n can be written as $V_n = \gamma/n^\alpha$ where γ is a positive constant and $\alpha \in [0.5, 1]$. Based on these expressions, we can rewrite (8.24) as

$$s_n \geq \sqrt{\frac{n^\alpha M + c\gamma}{c\gamma}} \log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right], \tag{8.25}$$

which follows the claim in (8.19). If we assume that we start with m_0 samples such that $N/m_0 = 2^q$ where q is an integer then the total number of gradient computations to achieve V_N for the risk R_N is given by

$$\begin{aligned}
&\sum_{n=m_0,2m_0,\ldots,N} \sqrt{\frac{n^\alpha M + c\gamma}{c\gamma}} \ \log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right] \\
&\leq \log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right] \sum_{n=m_0,2m_0,\ldots,N} 1 + \sqrt{\frac{n^\alpha M}{c\gamma}} \\
&= \log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right] \left[(q+1) + \sqrt{\frac{m_0^\alpha M}{c\gamma}} \left(\frac{\sqrt{2^{(q+1)^\alpha}} - 1}{\sqrt{2^\alpha} - 1}\right)\right] \\
&\leq \log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right] \left[(q+1) + \sqrt{\frac{{m_0}^\alpha M}{c\gamma}} \left(\frac{\sqrt{2^{(q+1)^\alpha}}}{\sqrt{2^\alpha} - 1}\right)\right] \\
&= \log\left[6 \times 2^\alpha + (2^\alpha - 1)\left(4 + c\|\mathbf{w}^*\|^2\right)\right] \left[(q+1) + \sqrt{\frac{N^\alpha M}{c\gamma}} \left(\frac{\sqrt{2^\alpha}}{\sqrt{2^\alpha} - 1}\right)\right].
\end{aligned} \tag{8.26}$$

Replacing q by $\log_2(N/m_0)$ leads to the bound in (8.20). ■

The result in Theorem 14 characterizes the number of required iterations s_n to achieve the statistical accuracy of R_n. Moreover, it shows that to reach the accuracy $V_N = \mathcal{O}(1/N^\alpha)$ for the risk R_N accosiated to the full training set $\mathcal{T}$, the total computational complexity of Ada AGD is of the order $\mathcal{O}\left(N^{(1+\alpha/2)}\right)$. Indeed, this complexity is lower than the overall computational complexity of AGD for reaching the same target which is given by $\mathcal{O}\left(N\sqrt{\kappa_N}\log(N^\alpha)\right) = \mathcal{O}\left(N^{(1+\alpha/2)}\log(N^\alpha)\right)$. Note that this bound holds for AGD since the condition number $\kappa_N := (M+cV_N)/(cV_N)$ of the risk R_N is of the order $\mathcal{O}(1/V_N) = \mathcal{O}(N^\alpha)$.

8.4.2 Adaptive sample size SVRG (Ada SVRG)

For the adaptive sample size mechanism presented in Section 8.3, we can also use linearly convergent *stochastic* methods such as stochastic variance reduced gradient (SVRG) in [45] to update the iterates. The SVRG method succeeds in reducing the computational complexity of deterministic first-order methods by computing a single gradient per iteration and using a *delayed* version of the average gradient to update the iterates. Indeed, we can exploit the idea of SVRG to develop low computational complexity adaptive sample size methods to improve the performance of deterministic adaptive sample size algorithms. Moreover, the adaptive sample size variant of SVRG (Ada SVRG) enhances the proven bounds for SVRG to solve ERM problems.

We proceed to extend the idea of adaptive sample size scheme to the SVRG algorithm. To do so, consider $\mathbf{w}_m$ as an iterate within the statistical accuracy, $\mathbb{E}[R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*)] \leq V_m$, for a set $\mathcal{S}_m$ which contains m samples. Consider s_n and q_n as the numbers of outer and inner loops for the update of SVRG, respectively, when the size of the training set is n. Further, consider $\tilde{\mathbf{w}}$ and $\hat{\mathbf{w}}$ as the sequences of iterates for the outer and inner loops of SVRG, respectively. In the adaptive sample size SVRG (Ada SVRG) method to minimize the risk R_n, we set the approximate solution $\mathbf{w}_m$ for the previous ERM problem as the initial iterate for the outer loop, i.e., $\tilde{\mathbf{w}}_0 = \mathbf{w}_m$. Then, the outer loop update which contains gradient computation is defined as

$$\nabla R_n(\tilde{\mathbf{w}}_k) = \frac{1}{n}\sum_{i=1}^{n} \nabla f(\tilde{\mathbf{w}}_k, \boldsymbol{\theta}_i) + cV_n\tilde{\mathbf{w}}_k \qquad \text{for} \qquad k = 0, \ldots, s_n - 1, \tag{8.27}$$

and the inner loop for the k-th outer loop contains q_n iterations of the following update

$$\hat{\mathbf{w}}_{t+1,k} = \hat{\mathbf{w}}_{t,k} - \eta_n \left(\nabla f(\hat{\mathbf{w}}_{t,k}, z_{i_t}) + cV_n\hat{\mathbf{w}}_{t,k} - \nabla f(\tilde{\mathbf{w}}_k, z_{i_t}) - cV_n\tilde{\mathbf{w}}_k + \nabla R_n(\tilde{\mathbf{w}}_k)\right), \tag{8.28}$$

for $t = 0, \ldots, q_n - 1$, where the iterates for the inner loop at step k are initialized as $\hat{\mathbf{w}}_{0,k} = \tilde{\mathbf{w}}_k$, and i_t is index of the function which is chosen unfirmly at random from the set $\{1, \ldots, n\}$ at the inner iterate t. The outcome of each inner loop $\hat{\mathbf{w}}_{q_n,k}$ is used as the variable for the next outer loop, i.e., $\tilde{\mathbf{w}}_{k+1} = \hat{\mathbf{w}}_{q_n,k}$. We define the outcome of s_n outer loops $\tilde{\mathbf{w}}_{s_n}$ as the approximate solution for the risk R_n, i.e., $\mathbf{w}_n = \tilde{\mathbf{w}}_{s_n}$.

In the following theorem we derive a bound on the number of required outer loops s_n to ensure that the variable $\mathbf{w}_n$ generated by the updates in (8.27) and (8.28) will be in the statistical accuracy of R_n in expectation, i.e., $\mathbb{E}[R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)] \leq V_n$. To reach the smallest possible lower bound for s_n, we properly choose the number of inner loop iterations q_n and the learning rate η_n.

Theorem 15 *Consider the variable $\mathbf{w}_m$ as a V_m-optimal solution of the risk R_m, i.e., a*

solution such that $\mathbb{E}[R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*)] \leq V_m$*, where* $V_m = \mathcal{O}(1/m^\alpha)$*. Consider the sets* $\mathcal{S}_m \subset \mathcal{S}_n \subset \mathcal{T}$ *such that* $n = 2m$*, and suppose Assumption 17 holds. Further, set the number of inner loop iterations as* $q_n = n$ *and the learning rate as* $\eta_n = 0.1/(M + cV_n)$*. Then, the variable* $\mathbf{w}_n$ *generated based on the update of Ada SVRG in* (8.27)-(8.28) *satisfies* $\mathbb{E}[R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)] \leq V_n$ *if the number of iterations* s_n *is larger than*

$$s_n \ \geq \ \log_2 \Big[3 \times 2^\alpha + (2^\alpha - 1)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\Big]. \tag{8.29}$$

Moreover, to reach the statistical accuracy V_N *of the full training set* $\mathcal{T}$ *the overall computational complexity of Ada SVRG is given by*

$$4N \ \log_2 \Big[3 \times 2^\alpha + (2^\alpha - 1)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\Big]. \tag{8.30}$$

Proof: Let's recall the convergence result of SVRG after s outer loop where each inner loop contains r iterations. We can show that if $\mathbf{w}_m$ is the variable corresponding to m samples and n is the variable associated with n samples, then we have

$$\mathbb{E}_n[R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)] \leq \rho^s \left[R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)\right], \tag{8.31}$$

where the expectation is taken with respect to the indices chosen in the inner loops, and the constant ρ is defined as

$$\rho := \frac{1}{\gamma\eta(1 - 2L_0\eta)r} + \frac{2L_0\eta}{1 - 2L_0\eta} < 1 \tag{8.32}$$

where γ is the constant of strong convexity, L_0 is the constant for the Lipschitz continuity of gradients, q is the number of inner loop iterations, and η is the stepsize. If we assume that $V_n = \mathcal{O}(1/n^\alpha)$, then we obtain that $\gamma = c/n^\alpha$ and $L_0 = M + c/n^\alpha$. Further, if we set the number of inner loop iteration as $q = n$ and the stepsize as $\eta = 0.1/L_0$, the expression for ρ can be simplified as

$$\rho := \frac{Mn^\alpha + c}{0.08nc} + \frac{1}{4} < \frac{1}{2}, \tag{8.33}$$

where the inequality holds since the size of training set is such that $(Mn^\alpha + c)/(nc) \leq 0.02$. Considering the result in (8.15) and the upper bound for the linear factor ρ, to ensure that that outocme of the Ada SVRG is within the statistical accuracy of the risk R_n the number of outer loops s_n should be larger than

$$s_n \ \geq \ \log_2 \Big[3 \times 2^\alpha + (2^\alpha - 1)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\Big], \tag{8.34}$$

and the result in (8.29) follows.

Since each outer loop requires one full gradient computation and n inner loop iterations the total number of gradient computations (computational complexity) of Ada SVRG at the stage of minimizing R_n is given by $2ns_n$. Therefore, if we assume that we start with m_0 samples such that $N/m_0 = 2^q$ where q is an integer, then the total number of gradient computations to achieve V_N for the risk R_N is given by

$$\begin{aligned}
&\sum_{n=m_0,2m_0,\ldots,N} 2n \ \log_2 \left[3 \times 2^\alpha + (2^\alpha - 1)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\right] \\
&= 2m_0 \frac{2^{q+1}-1}{2-1} \ \log_2 \left[3 \times 2^\alpha + (2^\alpha - 1)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\right] \\
&\leq 4N \ \log_2 \left[3 \times 2^\alpha + (2^\alpha - 1)\left(2 + \frac{c}{2}\|\mathbf{w}^*\|^2\right)\right], \qquad (8.35)
\end{aligned}$$

which yields the claim in (8.30). ■

The result in (8.29) shows that the minimum number of outer loop iterations for Ada SVRG is equal to $s_n = \lfloor \log_2[3 \times 2^\alpha + (2^\alpha - 1)(2 + (c/2)\|\mathbf{w}^*\|^2)] \rfloor + 1$. This bound leads to the result in (8.30) which shows that the overall computational complexity of Ada SVRG to reach the statistical accuracy of the full training set $\mathcal{T}$ is of the order $\mathcal{O}(N)$. This bound not only improves the bound $\mathcal{O}(N^{1+\alpha/2})$ for Ada AGD, but also enhances the complexity of SVRG for reaching the same target accuracy which is given by $\mathcal{O}((N + \kappa)\log(N^\alpha)) = \mathcal{O}(N \log(N^\alpha))$.

8.5 Experiments

In this section, we compare the adaptive sample size versions of a group of first-order methods, including gradient descent (GD), accelerated gradient descent (AGD), and stochastic variance reduced gradient (SVRG) with their standard (fixed sample size) versions. In this section, we first compare the performance of these methods on the RCV1 dataset. We use $N = 10,000$ samples of the RCV1 dataset for the training set and the remaining $10,242$ as the test set. The number of features in each sample is $p = 47,236$. In our experiments, we use logistic loss. The constant c should be within the order of gradients Lipschitz continuity constant M, and, therefore, we set it as $c = 1$ since the samples are normalized and $M = 1$. The size of the initial training set for adaptive methods is $m_0 = 400$. In our experiments we assume $\alpha = 0.5$ and therefore the added regularization term is $(1/\sqrt{n})\|\mathbf{w}\|^2$.

The plots in Figure 8.1 compare the suboptimality of GD, AGD, and SVRG with their adaptive sample size versions. As our theoretical results suggested, we observe that the adaptive sample size scheme reduces the overall computational complexity of all of the considered linearly convergent first-order methods. If we compare the test errors of GD,

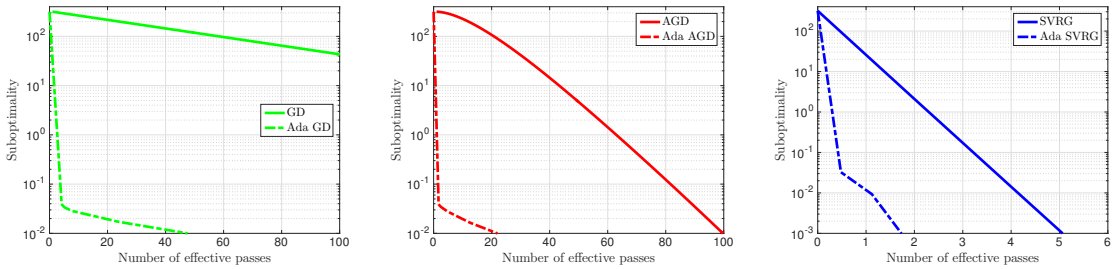


Figure 8.1: Comparison of GD, AGD, and SVRG with their adaptive sample size versions in terms of suboptimality vs. number of effective passes for RCV1 dataset with regularization of the order $\mathcal{O}(1/\sqrt{n})$.

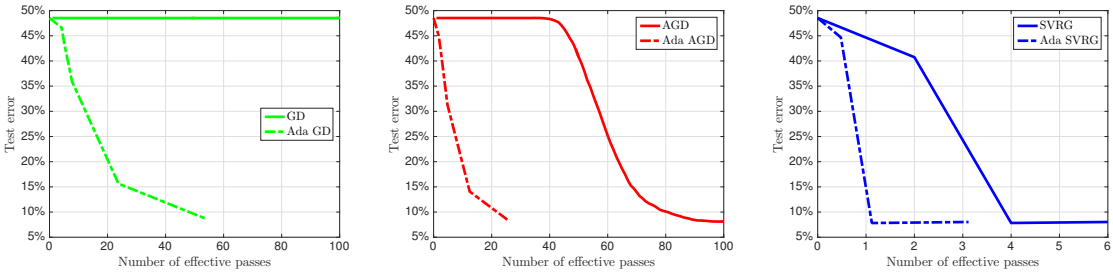


Figure 8.2: Comparison of GD, AGD, and SVRG with their adaptive sample size versions in terms of test error vs. number of effective passes for RCV1 dataset with regularization of the order $\mathcal{O}(1/\sqrt{n})$.

AGD, and SVRG with their adaptive sample size variants, we reach the same conclusion that the adaptive sample size scheme reduces the overall computational complexity to reach the statistical accuracy of the full training set. In particular, the left plot in Figure 8.2 shows that Ada GD approaches the minimum test error of 8% after 55 effective passes, while GD can not improve the test error even after 100 passes. Indeed, GD will reach lower test error if we run it for more iterations. The central plot in Figure 8.2 showcases that Ada AGD reaches 8% test error about 5 times faster than AGD. This is as predicted by $\log(N^{\alpha}) = \log(100) = 4.6$. The right plot in Figure 8.2 illustrates a similar improvement for Ada SVRG.

Now we proceed to compare these method using the MNIST dataset containing images of dimension $p = 784$. Since we are interested in a binary classification problem we only use the samples corresponding to digits 0 and 8, and, therefore, the number of samples is $11,774$. We choose $N = 6,000$ of these samples randomly and use them as the training set and use the remaining $5,774$ samples as the test set. We use the logistic loss to evaluate the performance of the classifier and normalize the samples to ensure that the constant for the Lipschitz continuity of the gradients is $M = 1$. In our experiments we consider two different scenarios. First we compare GD, AGD, and SVRG with their adaptive sample size versions when the additive regularization term is of order $1/\sqrt{n}$. Then, we redo the experiments for

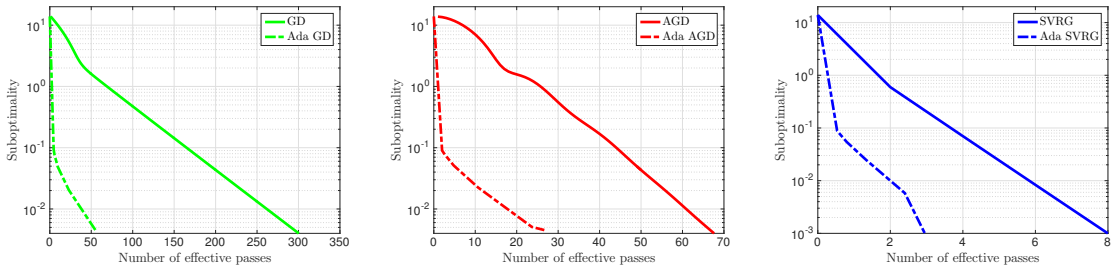


Figure 8.3: Comparison of GD, AGD, and SVRG with their adaptive sample size versions in terms of suboptimality vs. number of effective passes for MNIST dataset with regularization of the order $\mathcal{O}(1/\sqrt{n})$.

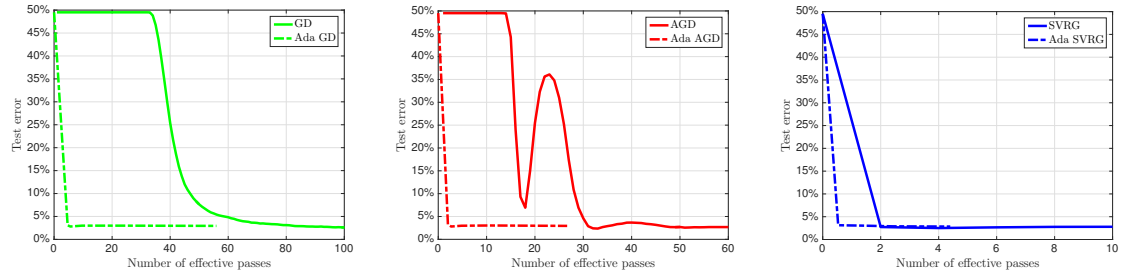


Figure 8.4: Comparison of GD, AGD, and SVRG with their adaptive sample size versions in terms of test error vs. number of effective passes for MNIST dataset with regularization of the order $\mathcal{O}(1/\sqrt{n})$.

a regularization term of order $1/n$.

The plots in Figure 8.3 compare the suboptimality of GD, AGD, and SVRG with Ada GD, Ada AGD, and Ada SVRG when the regularization term in $(1/\sqrt{n})\|\mathbf{w}\|^2$. Note that in this case the statistical accuracy should be order of $\mathcal{O}(1/\sqrt{n})$ and therefore we are interested in the number of required iterations to achieve the suboptimality of order 10^{-2}. As we observe Ada GD reach this target accuracy almost 6 times faster than GD. The improvement for Ada AGD and Ada SVRG is less significant, but they still reach the suboptimality of 10^{-2} significantly faster than their standard (fixed sample size) methods. Figure 8.4 illustrates the test error of GD, AGD, SVRG, Ada GD, Ada AGD, and Ada SVRG versus the number of effective passes over the dataset when the added regularization is of the order $\mathcal{O}(1/\sqrt{n})$. Comparison of these methods in terms of test error also support the gain in solving subproblems sequentially instead of minimizing the ERM corresponding to the full training set directly. In particular, for all three methods, the adaptive sample size version reaches the minimum test error of 2.5% faster than the fixed sample size version.

We also run the same experiments for the case that the regularization term is order $1/n$. Figure 8.5 shows the suboptimality of GD, AGD, and SVRG and their adaptive sample size version for the MNIST dataset when V_n is assumed to be $\mathcal{O}(1/n)$. We expect from

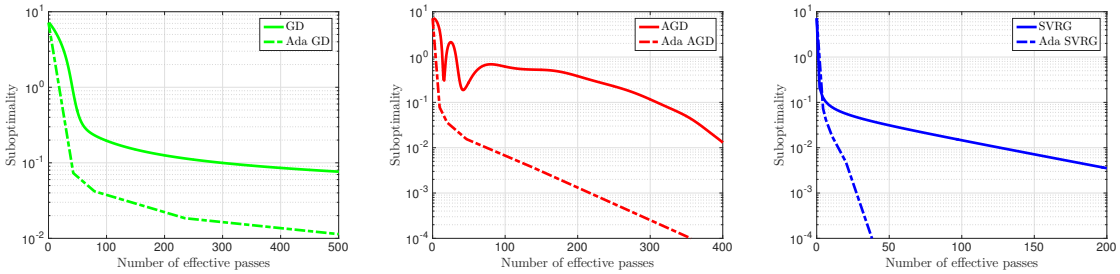


Figure 8.5: Comparison of GD, AGD, and SVRG with their adaptive sample size versions in terms of suboptimality vs. number of effective passes for MNIST dataset with regularization of the order $\mathcal{O}(1/n)$.

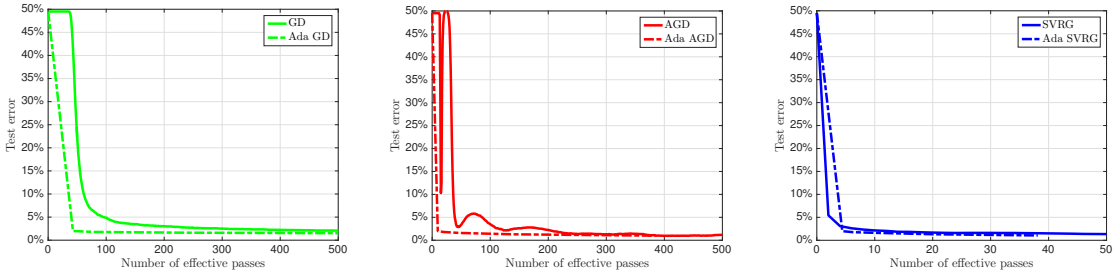


Figure 8.6: Comparison of GD, AGD, and SVRG with their adaptive sample size versions in terms of test error vs. number of effective passes for the MNIST dataset with regularization of the order $\mathcal{O}(1/n)$.

our theoretical achievements the advantage of using adaptive sample size scheme in this setting should be more significant, since $\log(N)$ is twice the value of $\log(\sqrt{N})$. Figure 8.5 fulfills this expectation by showing that Ada GD, Ada AGD, and Ada SVRG are almost 10 times faster than GD, AGD, and SVRG, respectively. Figure 8.6 demonstrates the test error of these methods versus the number of effective passes for a regularization of order $\mathcal{O}(1/n)$. In this case, this case all methods require more passes to achieve the minimum test error comparing to the case that regularization is of order $\mathcal{O}(1/n)$. Interestingly, the minimum accuracy in this case is equal to 1% which is lower than 2.5% for the previous setting. Indeed, the difference between the number of required passes to reach the minimum test error for adaptive sample size methods and their standard version is more significant since the factor $\log(N^{\alpha})$ is larger.

8.6 Discussions

We presented an adaptive sample size scheme to improve the convergence guarantees for a class of first-order methods which have linear convergence rates under strong convexity and smoothness assumptions. The logic behind the proposed adaptive sample size scheme is to

replace the solution of a relatively *hard* problem – the ERM problem for the full training set – by a sequence of relatively *easier* problems – ERM problems corresponding to a subset of samples. Indeed, whenever $m < n$, solving the ERM problems in (9.1) for loss R_m is simpler than the one for loss R_n because:

(i) The adaptive regularization term of order V_m makes the condition number of R_m smaller than the condition number of R_n – which uses a regularizer of order V_n.

(ii) The approximate solution $\mathbf{w}_m$ that we need to find for R_m is less accurate than the approximate solution $\mathbf{w}_n$ we need to find for R_n.

(iii) The computation cost of an iteration for R_m – e.g., the cost of evaluating a gradient – is lower than the cost of an iteration for R_n.

Properties (i)-(iii) combined with the ability to grow the sample size geometrically, reduce the overall computational complexity for reaching the statistical accuracy of the full training set. We particularized our results to develop adaptive (Ada) versions of AGD and SVRG. In both methods we found a computational complexity reduction of order $\mathcal{O}(\log(1/V_N)) = \mathcal{O}(\log(N^\alpha))$ which was corroborated in numerical experiments. The idea and analysis of adaptive first order methods apply generically to any other approach with linear convergence rate (Theorem 13). The development of sample size adaptation for sublinear methods is left for future research.

Chapter 9

Second-order adaptive sample size method

In Chapter 8 we studied adaptive sample size first order methods and showed that the idea of geometrically increasing the size of the training set leads to better convergence guarantees for solving ERM problems.

The goal of this chapter is to extend the adaptive sample size idea into second-order methods by introducing Ada Newton method. The main idea of Ada Newton is to increase the size of the training set by a factor larger than one in a way that the minimization variable for the current training set is in the local neighborhood of the optimal argument of the next training set. This allows to exploit the quadratic convergence property of Newton's method and reach the statistical accuracy of each training set with only one iteration of Newton's method. We discuss the details in the following sections.

9.1 Context and background

A hallmark of empirical risk minimization (ERM) on large datasets is that evaluating descent directions requires a complete pass over the dataset. Since this is undesirable due to the large number of training samples, stochastic optimization algorithms with descent directions estimated from a subset of samples are the method of choice. First order stochastic optimization has a long history [88, 98] but the last decade has seen fundamental progress in developing alternatives with faster convergence. A partial list of this consequential literature includes Nesterov acceleration [5, 85], stochastic averaging gradient [31, 49], variance reduction [45, 125], and dual coordinate methods [109, 110].

When it comes to stochastic second order methods the first challenge is that while *evaluation* of Hessians is as costly as evaluation of gradients, the stochastic *estimation* of Hessians has proven more challenging. This difficulty is addressed by incremental computa-

tions in [39] and subsampling in [34] or circumvented altogether in stochastic quasi-Newton methods [58, 70, 72, 78, 105]. Despite this incipient progress it is nonetheless fair to say that the striking success in developing stochastic first order methods is not matched by equal success in the development of stochastic second order methods. This is because even if the problem of estimating a Hessian is solved there are still four challenges left in the implementation of Newton-like methods in ERM:

(i) Global convergence of Newton's method requires implementation of a line search subroutine and line searches in ERM require a complete pass over the dataset.

(ii) The quadratic convergence advantage of Newton's method manifests close to the optimal solution but there is no point in solving ERM problems beyond their statistical accuracy.

(iii) Newton's method works for strongly convex functions but loss functions are not strongly convex for many ERM problems of practical importance.

(iv) Newton's method requires inversion of Hessians which is costly in large dimensional ERM.

Because stochastic Newton-like methods can't use line searches [cf. (i)], must work on problems that may be not strongly convex [cf. (iii)], and never operate very close to the optimal solution [cf (ii)], they never experience quadratic convergence. They do improve convergence constants and, if efforts are taken to mitigate the cost of inverting Hessians [cf. (iv)] as in [34, 70, 91, 105] they result in faster convergence. But since they still converge at linear rates they do not enjoy the foremost benefits of Newton's method.

In this paper we attempt to circumvent (i)-(iv) with the Ada Newton algorithm that combines the use of Newton iterations with adaptive sample sizes [29]. Say the total number of available samples is N, consider subsets of $n \leq N$ samples, and suppose the statistical accuracy of the ERM associated with n samples is V_n. In Ada Newton we add a quadratic regularization term of order V_n to the empirical risk – so that the regularized risk also has statistical accuracy V_n – and assume that for a certain initial sample size m_0, the problem has been solved to its statistical accuracy V_{m_0}. The sample size is then increased by a factor $\alpha > 1$ to $n = \alpha m_0$. We proceed to perform a single Newton iteration with unit stepsize and prove that the result of this update solves this extended ERM problem to its statistical accuracy (Section 9.2). This permits a second increase of the sample size by a factor α and a second Newton iteration that is likewise guaranteed to solve the problem to its statistical accuracy. Overall, this permits minimizing the empirical risk in $\alpha/(\alpha - 1)$ passes over the dataset and inverting $\log_\alpha N$ Hessians. Our theoretical results provide a characterization of the values of α that are admissible with respect to different problem

parameters (Theorem 16). In particular, we show that asymptotically on the number of samples n and with proper parameter selection we can set $\alpha = 2$ (Proposition 10). In such case we can optimize to within statistical accuracy in about 2 passes over the dataset and after inversion of about $3.32 \log_{10} N$ Hessians. Our numerical experiments verify that $\alpha = 2$ is a valid factor for increasing the size of the training set at each iteration while performing a single Newton iteration for each value of the sample size.

9.2 Ada Newton

Recall the problem formulation for ERM in Section 8.2. As we discussed previously, solving the optimization problem

$$\mathbf{w}_n^* := \operatorname*{argmin}_{\mathbf{w}} R_n(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} L_n(\mathbf{w}) + \frac{cV_n}{2}\|\mathbf{w}\|^2. \tag{9.1}$$

within the statistical accuracy V_n, i.e., $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$, leads to an V_n suboptimal solution for the original stochastic optimization problem in (8.1). Thus, we can say that a variable $\mathbf{w}_n$ satisfying $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$ solves the ERM problem to within its statistical accuracy. Therefore, our aim is to solve the problem in (9.1) for $n = N$ within its statistical accuracy.

To solve (9.1) suppose the problem has been solved to within its statistical accuracy for a set $\mathcal{S}_m \subset \mathcal{S}_n$ with $m = n/\alpha$ samples where $\alpha > 1$. Therefore, we have found a variable $\mathbf{w}_m$ for which $R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \leq V_m$. Our goal is to update $\mathbf{w}_m$ using the Newton step in a way that the updated variable $\mathbf{w}_n$ estimates $\mathbf{w}_n^*$ with accuracy V_n. To do so compute the gradient of the risk R_n evaluated at $\mathbf{w}_m$

$$\nabla R_n(\mathbf{w}_m) = \frac{1}{n}\sum_{k=1}^{n} \nabla f(\mathbf{w}_m, z_k) + cV_n\mathbf{w}_m, \tag{9.2}$$

as well as the Hessian $\mathbf{H}_n$ of R_n evaluated at $\mathbf{w}_m$

$$\mathbf{H}_n := \nabla^2 R_n(\mathbf{w}_m) = \frac{1}{n}\sum_{k=1}^{n} \nabla^2 f(\mathbf{w}_m, z_k) + cV_n\mathbf{I}, \tag{9.3}$$

and update $\mathbf{w}_m$ with the Newton step of the regularized risk R_n to compute

$$\mathbf{w}_n = \mathbf{w}_m - \mathbf{H}_n^{-1}\nabla R_n(\mathbf{w}_m). \tag{9.4}$$

Note that the stepsize of the Newton update in (9.4) is 1, which avoids line search algorithms requiring extra computation. The main contribution of this paper is to derive a condition

that guarantees that $\mathbf{w}_n$ solves R_n to within its statistical accuracy V_n. To do so, we first assume the following conditions are satisfied.

Assumption 18 *The loss functions $f(\mathbf{w}, \mathbf{z})$ are convex with respect to $\mathbf{w}$ for all values of $\mathbf{z}$. Moreover, their gradients $\nabla f(\mathbf{w}, \mathbf{z})$ are Lipschitz continuous with constant M*

$$\|\nabla f(\mathbf{w}, \mathbf{z}) - \nabla f(\mathbf{w}', \mathbf{z})\| \leq M\|\mathbf{w} - \mathbf{w}'\|, \qquad \text{for all } \mathbf{z}. \tag{9.5}$$

Assumption 19 *The loss functions $f(\mathbf{w}, \mathbf{z})$ are self-concordant with respect to $\mathbf{w}$ for all $\mathbf{z}$.*

Assumption 20 *The difference between the gradients of the empirical loss L_n and the statistical average loss L is bounded by $V_n^{1/2}$ for all $\mathbf{w}$ with high probability,*

$$\sup_{\mathbf{w}} \|\nabla L(\mathbf{w}) - \nabla L_n(\mathbf{w})\| \leq V_n^{1/2}, \qquad w.h.p. \tag{9.6}$$

The conditions in Assumption 18 imply that the average loss $L(\mathbf{w})$ and the empirical loss $L_n(\mathbf{w})$ are convex and their gradients are Lipschitz continuous with constant M. Thus, the empirical risk $R_n(\mathbf{w})$ is strongly convex with constant cV_n and its gradients $\nabla R_n(\mathbf{w})$ are Lipschitz continuous with parameter $M + cV_n$. Likewise, the condition in Assumption 19 implies that the average loss $L(\mathbf{w})$, the empirical loss $L_n(\mathbf{w})$, and the empirical risk $R_n(\mathbf{w})$ are also self-concordant. The condition in Assumption 20 says that the gradients of the empirical risk converge to their statistical average at a rate of order $V_n^{1/2}$. If the constant V_n is of order not faster than $O(1/n)$ the condition in Assumption 20 holds if the gradients converge to their statistical average at a rate of order $V_n^{1/2} = O(1/\sqrt{n})$. This is a conservative rate for the law of large numbers.

In the following theorem, given Assumptions 18-20, we state a condition that guarantees the variable $\mathbf{w}_n$ evaluated as in (9.4) solves R_n to within its statistical accuracy V_n.

Theorem 16 *Consider the variable $\mathbf{w}_m$ as a V_m-optimal solution of the risk R_m, i.e., a solution such that $R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \leq V_m$. Let $n = \alpha m > m$, consider the risk R_n associated with sample set $\mathcal{S}_n \supset \mathcal{S}_m$, and suppose assumptions 18 - 20 hold. If the sample size n is chosen such that*

$$\left(\frac{2(M + cV_m)V_m}{cV_n}\right)^{1/2} + \frac{2(n-m)}{nc^{1/2}} + \frac{\left((2+\sqrt{2})c^{1/2} + c\|\mathbf{w}^*\|\right)(V_m - V_n)}{(cV_n)^{1/2}} \leq \frac{1}{4} \tag{9.7}$$

and

$$144\left(V_m + \frac{2(n-m)}{n}\left(V_{n-m} + V_m\right) + 2\left(V_m - V_n\right) + \frac{c(V_m - V_n)}{2}\|\mathbf{w}^*\|^2\right)^2 \leq V_n \tag{9.8}$$

Algorithm 11 Ada Newton

Parameters: Sample size increase constants $\alpha_0 > 1$ and $0 < \beta < 1$.
Input: Initial sample size $n = m_0$ and argument $\mathbf{w}_n = \mathbf{w}_{m_0}$ with $\|\nabla R_n(\mathbf{w}_n)\| < (\sqrt{2c})V_n$
while [**do**main loop]$n \leq N$
    Update argument and index: $\mathbf{w}_m = \mathbf{w}_n$ and $m = n$. Reset factor $\alpha = \alpha_0$.
    repeat[sample size backtracking loop]
        Increase sample size: $n = \min\{\alpha m, N\}$.
        Compute gradient [cf. (9.2)]: $\nabla R_n(\mathbf{w}_m) = (1/n)\sum_{k=1}^{n} \nabla f(\mathbf{w}_m, z_k) + cV_n\mathbf{w}_m$
        Compute Hessian [cf. (9.3)]: $\mathbf{H}_n = (1/n)\sum_{k=1}^{n} \nabla^2 f(\mathbf{w}_m, z_k) + cV_n\mathbf{I}$
        Newton Update [cf. (9.4)]: $\mathbf{w}_n = \mathbf{w}_m - \mathbf{H}_n^{-1}\nabla R_n(\mathbf{w}_m)$
        Compute gradient [cf. (9.2)]: $\nabla R_n(\mathbf{w}_n) = (1/n)\sum_{k=1}^{n} \nabla f(\mathbf{w}_n, z_k) + cV_n\mathbf{w}_n$
        Backtrack sample size increase $\alpha = \beta\alpha$.
    until $\|\nabla R_n(\mathbf{w}_n)\| < (\sqrt{2c})V_n$
end while

are satisfied, then the variable $\mathbf{w}_n$*, which is the outcome of applying one Newton step on the variable* $\mathbf{w}_m$ *as in* (9.4)*, has sub-optimality error* V_n *with high probability, i.e.,*

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n, \qquad w.h.p. \tag{9.9}$$

Proof: See Section 9.3. ■

Theorem 16 states conditions under which we can iteratively increase the sample size while applying single Newton iterations without line search and staying within the statistical accuracy of the regularized empirical risk. The constants in (9.7) and (9.8) are not easy to parse but we can understand them qualitatively if we focus on large m. This results in a simpler condition that we state next.

Proposition 10 *Consider a learning problem in which the statistical accuracy satisfies* $V_m \leq \alpha V_n$ *for* $n = \alpha m$ *and* $\lim_{n\to\infty} V_n = 0$*. If the regularization constant* c *is chosen so that*

$$\left(\frac{2\alpha M}{c}\right)^{1/2} + \frac{2(\alpha - 1)}{\alpha c^{1/2}} < \frac{1}{4}, \tag{9.10}$$

then, there exists a sample size $\tilde{m}$ *such that* (9.7) *and* (9.8) *are satisfied for all* $m > \tilde{m}$ *and* $n = \alpha m$*. In particular, if* $\alpha = 2$ *we can satisfy* (9.7) *and* (9.8) *with* $c > 16(2\sqrt{M} + 1)^2$*.*

Proof: That the condition in (9.8) is satisfied for all $m > \tilde{m}$ follows simply because the left hand side is of order V_m^2 and the right hand side is of order V_n. To show that the condition in (9.7) is satisfied for sufficiently large m observe that the third summand in (9.7) is of order $O((V_m - V_n)/V_n^{1/2})$ and vanishes for large m. In the second summand of (9.7) we make $n = \alpha m$ to obtain the second summand in (9.10) and in the first summand replace

the ratio V_m/V_n by its bound α to obtain the first summand of (9.10). To conclude the proof just observe that the inequality in (9.10) is strict. ■

The condition $V_m \leq \alpha V_n$ is satisfied if $V_n = 1/n$ and is also satisfied if $V_n = 1/\sqrt{n}$ because $\sqrt{\alpha} < \alpha$. This means that for most ERM problems we can progress geometrically over the sample size and arrive at a solution $\mathbf{w}_N$ that solves the ERM problem R_N to its statistical accuracy V_N as long as (9.10) is satisfied .

The result in Theorem 16 motivates definition of the Ada Newton algorithm that we summarize in Algorithm 11. The core of the algorithm is in steps 6-9. Step 6 implements an increase in the sample size by a factor α and steps 7-9 implement the Newton iteration in (9.2)-(9.4). The required input to the algorithm is an initial sample size m_0 and a variable $\mathbf{w}_{m_0}$ that is known to solve the ERM problem with accuracy V_{m_0}. Observe that this initial iterate doesn't have to be computed with Newton iterations. The initial problem to be solved contains a moderate number of samples m_0, a mild condition number because it is regularized with constant cV_{m_0}, and is to be solved to a moderate accuracy V_{m_0} – recall that V_{m_0} is of order $V_{m_0} = O(1/m_0)$ or order $V_{m_0} = O(1/\sqrt{m_0})$ depending on regularity assumptions. Stochastic first order methods excel at solving problems with moderate number of samples m_0 and moderate condition to moderate accuracy.

We remark that the conditions in Theorem 16 and Proposition 10 are conceptual but that the constants involved are unknown in practice. In particular, this means that the allowed values of the factor α that controls the growth of the sample size are unknown a priori. We solve this problem in Algorithm 11 by backtracking the increase in the sample size until we guarantee that $\mathbf{w}_n$ minimizes the empirical risk $R_n(\mathbf{w}_n)$ to within its statistical accuracy. This backtracking of the sample size is implemented in Step 11 and the optimality condition of $\mathbf{w}_n$ is checked in Step 12. The condition in Step 12 is on the gradient norm that, because R_n is strongly convex, can be used to bound the suboptimality $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ as

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq \frac{1}{2cV_n} \|\nabla R_n(\mathbf{w}_n)\|^2. \tag{9.11}$$

Observe that checking this condition requires an extra gradient computation undertaken in Step 10. That computation can be reused in the computation of the gradient in Step 5 once we exit the backtracking loop. We emphasize that when the condition in (9.10) is satisfied, there exists a sufficiently large m for which the conditions in Theorem 16 are satisfied for $n = \alpha m$. This means that the backtracking condition in Step 12 is satisfied after one iteration and that, eventually, Ada Newton progresses by increasing the sample size by a factor α. This means that Algorithm 11 can be thought of as having a damped phase where the sample size increases by a factor smaller than ρ and a geometric phase where the sample size grows by a factor ρ in all subsequent iterations. The computational cost of this geometric phase is of not more than $\alpha/(\alpha - 1)$ passes over the dataset and

requires inverting not more than $\log_\alpha N$ Hessians. If $c > 16(2\sqrt{M}+1)^2$, we make $\alpha = 2$ for optimizing to within statistical accuracy in about 2 passes over the dataset and after inversion of about $3.32 \log_{10} N$ Hessians.

9.3 Convergence analysis

In this section we study the proof of Theorem 16. To do so, first we prove Lemmata 26 and 27 which are intermediate results that we use in proving the mentioned propositions. We start the analysis by providing an upper bound for the difference between the loss functions L_n and L_m.

Lemma 26 *Consider L_n and L_m as the empirical losses of the sets $\mathcal{S}_n$ and $\mathcal{S}_m$, respectively, where they are chosen such that $\mathcal{S}_m \subset \mathcal{S}_n$. If we define n and m as the number of samples in the training sets $\mathcal{S}_n$ and $\mathcal{S}_m$, respectively, then the absolute value of the difference between the empirical losses is bounded above by*

$$|L_n(\mathbf{w}) - L_m(\mathbf{w})| \leq \frac{n-m}{n} \left(V_{n-m} + V_m\right), \qquad w.h.p. \tag{9.12}$$

for any $\mathbf{w}$.

Proof: The proof is very similar to the proof of Lemma 30 in Appendix C.1 except a minor difference that the result in (9.12) holds with high probability instead of in expectation as in (C.5). This difference is the outcome of using the inequality $\sup_{\mathbf{w}} |L(\mathbf{w}) - L_n(\mathbf{w})| \leq V_n$ with high probability instead of the relation $\sup_{\mathbf{w}} \mathbb{E}\left[|L(\mathbf{w}) - L_n(\mathbf{w})|\right] \leq V_n$. The proof is omitted due to similarity of these analyses. ■

The result in Lemma 26 shows that the upper bound for the difference between the loss functions associated with the sets $\mathcal{S}_m$ and $\mathcal{S}_n$ where $\mathcal{S}_m \subset \mathcal{S}_n$ is proportional to the difference between the size of these two sets $n - m$. This result will help us later to understand how much we can increase the size of the training set at each iteration. In other words, how large the difference $n - m$ could be, while we have the statistical accuracy.

In the following lemma, we characterize an upper bound for the norm of the optimal argument $\mathbf{w}_n^*$ of the empirical risk $R_n(\mathbf{w})$ in terms of the norm of statistical average loss $L(\mathbf{w})$ optimal argument $\mathbf{w}^*$.

Lemma 27 *Consider L_n as the empirical loss of the set $\mathcal{S}_n$ and L as the statistical average loss. Moreover, recall $\mathbf{w}^*$ as the optimal argument of the statistical average loss L, i.e., $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} L(\mathbf{w})$. If Assumption 18 holds, then the norm of the optimal argument $\mathbf{w}_n^*$*

of the regularized empirical risk $R_n(\mathbf{w}) := L_n(\mathbf{w}) + cV_n\|\mathbf{w}\|^2$ *is bounded above by*

$$\|\mathbf{w}_n^*\|^2 \leq \frac{4}{c} + \|\mathbf{w}^*\|^2, \qquad w.h.p. \tag{9.13}$$

Proof: The proof is very similar to the proof of Lemma 31 in Appendix C.1 except a minor difference that the result in (9.13) holds with high probability instead of in expectation as in (C.6). This difference is the outcome of using the inequality $\sup_{\mathbf{w}} |L(\mathbf{w}) - L_n(\mathbf{w})| \leq V_n$ with high probability instead of the relation $\sup_{\mathbf{w}} \mathbb{E}\left[|L(\mathbf{w}) - L_n(\mathbf{w})|\right] \leq V_n$. The proof is omitted due to similarity of these analyses. ■

The main idea of the Ada Newton algorithm is introducing a policy for increasing the size of training set from m to n in a way that the current variable $\mathbf{w}_m$ is in the Newton quadratic convergence phase for the next regularized empirical risk R_n. In the following proposition, we characterize the required condition to guarantee staying in the local neighborhood of Newton's method.

Proposition 11 *Consider the sets* $\mathcal{S}_m$ *and* $\mathcal{S}_n$ *as subsets of the training set* $\mathcal{T}$ *such that* $\mathcal{S}_m \subset \mathcal{S}_n \subset \mathcal{T}$*. We assume that the number of samples in the sets* $\mathcal{S}_m$ *and* $\mathcal{S}_n$ *are* m *and* n*, respectively. Further, define* $\mathbf{w}_m$ *as an* V_m *optimal solution of the risk* R_m*, i.e.,* $R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \leq V_m$*. In addition, define* $\lambda_n(\mathbf{w}) := \left(\nabla R_n(\mathbf{w})^T \nabla^2 R_n(\mathbf{w})^{-1} \nabla R_n(\mathbf{w})\right)^{1/2}$ *as the Newton decrement of variable* $\mathbf{w}$ *associated with the risk* R_n*. If Assumption 18-20 hold, then Newton's method at point* $\mathbf{w}_m$ *is in the quadratic convergence phase for the objective function* R_n*, i.e.,* $\lambda_n(\mathbf{w}_m) < 1/4$*, if we have*

$$\left(\frac{2(M + cV_m)V_m}{cV_n}\right)^{1/2} + \frac{(2(n-m)/n)V_n^{1/2} + (\sqrt{2c} + 2\sqrt{c} + c\|\mathbf{w}^*\|)(V_m - V_n)}{(cV_n)^{1/2}} \leq \frac{1}{4} \quad w.h.p. \tag{9.14}$$

Proof: From the self-concordance analysis of Newton's method we know that the variable $\mathbf{w}_m$ is in the neighborhood that Newton's method has a quadratic convergence rate if $\lambda_n(\mathbf{w}_m) \leq 1/4$; see e.g., Chapter 9 of [20]. We proceed to come up with a condition for the quadratic convergence phase which guarantees that $\lambda_n(\mathbf{w}_m) < 1/4$ and $\mathbf{w}_m$ is in the local neighborhood of the optimal argument of R_n. Recall that we have a $\mathbf{w}_m$ which has sub-optimality V_m for R_m. We then proceed to enlarge the sample size to n and start from the observation that we can bound $\lambda_n(\mathbf{w}_m)$ as

$$\lambda_n(\mathbf{w}_m) = \|\nabla R_n(\mathbf{w}_m)\|_{\mathbf{H}_n^{-1}} \leq \|\nabla R_m(\mathbf{w}_m)\|_{\mathbf{H}_n^{-1}} + \|\nabla R_n(\mathbf{w}_m) - \nabla R_m(\mathbf{w}_m)\|_{\mathbf{H}_n^{-1}}, \tag{9.15}$$

where we have used the definition $\mathbf{H}_n = \nabla^2 R_n(\mathbf{w}_m)$. Note that the weighted norm $\|\mathbf{a}\|_{\mathbf{A}}$ for vector $\mathbf{a}$ and matrix $\mathbf{A}$ is equal to $\|\mathbf{a}\|_{\mathbf{A}} = (\mathbf{a}^T\mathbf{A}\mathbf{a})^{1/2}$. First, we bound the

norm $\|\nabla R_n(\mathbf{w}_m)\|_{\mathbf{H}_n^{-1}}$ in (9.15). Notice that the Hessian $\nabla^2 R_n(\mathbf{w}_m)$ can be written as $\nabla^2 L_n(\mathbf{w}_m) + cV_n\mathbf{I}$. Thus, the eigenvalues of the Hessian $\mathbf{H}_n = \nabla^2 R_n(\mathbf{w}_m)$ are bounded below by cV_n and consequently the eigenvalues of the Hessian inverse $\mathbf{H}_n^{-1} = \nabla^2 R_n(\mathbf{w}_m)^{-1}$ are upper bounded by $1/(cV_n)$. This bound implies that $\|\mathbf{H}_n^{-1}\| \leq 1/(cV_n)$. Moreover, from Theorem 2.1.5 of [83], we know that the Lipschitz continuity of the gradients $\nabla R_m(\mathbf{w})$ with constant $M + cV_m$ implies that

$$\|\nabla R_m(\mathbf{w}_m)\|^2 \leq 2(M + cV_m)(R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*)) \leq 2(M + cV_m)V_m, \tag{9.16}$$

where the last inequality holds comes from the condition that $R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \leq V_m$. Considering the upper bound for $\|\nabla R_m(\mathbf{w}_m)\|^2$ in (9.16) and the inequality $\|\nabla^2 R_n(\mathbf{w}_m)^{-1}\| \leq 1/(cV_n)$ we can write

$$\|\nabla R_m(\mathbf{w}_m)\|_{\mathbf{H}_n^{-1}} = \left[\nabla R_m(\mathbf{w}_m)^T \mathbf{H}_n^{-1} \nabla R_m(\mathbf{w}_m)\right]^{1/2} \leq \left(\frac{2(M + cV_m)V_m}{cV_n}\right)^{1/2}. \tag{9.17}$$

Now we proceed to bound the second the term in (9.15). The definition of the risk function the gradient can be written as $\nabla R_n(\mathbf{w}) = \nabla L_n(\mathbf{w}) + (cV_n)\mathbf{w}$. Thus, we can derive an upper bound for the difference $\|\nabla R_n(\mathbf{w}_m) - \nabla R_m(\mathbf{w}_m)\|$ as

$$\begin{aligned}
&\|\nabla R_n(\mathbf{w}_m) - \nabla R_m(\mathbf{w}_m)\| \\
&\quad \leq \|\nabla L_n(\mathbf{w}_m) - \nabla L_m(\mathbf{w}_m)\| + c(V_m - V_n)\|\mathbf{w}_m\| \\
&\quad \leq \|\nabla L_n(\mathbf{w}_m) - \nabla L_m(\mathbf{w}_m)\| + c(V_m - V_n)\|\mathbf{w}_m - \mathbf{w}_m^*\| + c(V_m - V_n)\|\mathbf{w}_m^*\|,
\end{aligned} \tag{9.18}$$

where in the second inequality we have used the triangle inequality and replaced $\|\mathbf{w}_m\|$ by its upper bound $\|\mathbf{w}_m - \mathbf{w}_m^*\| + \|\mathbf{w}_m^*\|$. By following the steps in (C.2)-(9.12) we can show that the difference $\|\nabla L_n(\mathbf{w}_m) - \nabla L_m(\mathbf{w}_m)\|$ is bounded above by

$$\begin{aligned}
\|\nabla L_n(\mathbf{w}) - \nabla L_m(\mathbf{w})\| &\leq \frac{n-m}{n}\|\nabla L_{n-m}(\mathbf{w}) - \nabla L(\mathbf{w})\| + \frac{n-m}{n}\|\nabla L_m(\mathbf{w}) - \nabla L(\mathbf{w})\| \\
&\leq \frac{2(n-m)}{n} V_n^{1/2},
\end{aligned} \tag{9.19}$$

where the second inequality uses the condition that $\|\nabla L_m(\mathbf{w}) - \nabla L(\mathbf{w})\| \leq V_m^{1/2}$ as in Assumption 3.

Note that the strong convexity of the risk R_m with parameter cV_m yields

$$\|\mathbf{w}_m - \mathbf{w}_m^*\|^2 \leq \frac{2}{cV_m}(R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*)) \leq \frac{2}{c}. \tag{9.20}$$

Thus, by considering the inequalities in (9.19) and (9.20) we can show that upper bound in (9.18) can be replaced by

$$\|\nabla R_n(\mathbf{w}_m) - \nabla R_m(\mathbf{w}_m)\| \leq \frac{2(n-m)}{n} V_n^{1/2} + (\sqrt{2c} + c\|\mathbf{w}_m^*\|)(V_m - V_n). \tag{9.21}$$

Substituting the upper bounds in (9.17) and (9.21) for the first and second summands in (9.15), respectively, follows the inequality

$$\lambda_n(\mathbf{w}_m) \leq \left(\frac{2(M + cV_m)V_m}{cV_n}\right)^{1/2} + \frac{(2(n-m)/n)V_n^{1/2} + (\sqrt{2c} + c\|\mathbf{w}_m^*\|)(V_m - V_n)}{(cV_n)^{1/2}}. \tag{9.22}$$

Note that the result in (9.13) shows that $\|\mathbf{w}_m^*\|^2 \leq (4/c) + \|\mathbf{w}^*\|^2$ with high probability. This observation implies that $\|\mathbf{w}_m^*\|$ is bounded above by $(2/\sqrt{c}) + \|\mathbf{w}^*\|$. Replacing the norm $\|\mathbf{w}_m^*\|$ in (9.22) by the upper bound $(2/\sqrt{c}) + \|\mathbf{w}^*\|$ yields

$$\lambda_n(\mathbf{w}_m) \leq \left(\frac{2(M + cV_m)V_m}{cV_n}\right)^{1/2} + \frac{(2(n-m)/n)V_n^{1/2} + (\sqrt{2c} + 2\sqrt{c} + c\|\mathbf{w}^*\|)(V_m - V_n)}{(cV_n)^{1/2}}. \tag{9.23}$$

As we mentioned previously, the variable $\mathbf{w}_m$ is in the neighborhood that Newton's method has a quadratic convergence rate for the function R_n if the condition $\lambda_n(\mathbf{w}_m) \leq 1/4$ holds. Hence, if the right hand side of (9.23) is bounded above by $1/4$ we can conclude that $\mathbf{w}_m$ is in the local neighborhood and the proof is complete. ■

From the analysis of Newton's method we know that if the Newton decrement $\lambda_n(\mathbf{w})$ is smaller than $1/4$, the variable $\mathbf{w}$ is in the local neighborhood of Newton's method; see e.g., Chapter 9 of [20]. From the result in Proposition 11, we obtain a sufficient condition to guarantee that $\lambda_n(\mathbf{w}_m) < 1/4$ which implies that $\mathbf{w}_m$, which is a V_m optimal solution for the regularized empirical loss R_m, i.e., $R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \leq V_m$, is in the local neighborhood of the optimal argument of R_n that Newton's method converges quadratically.

Unfortunately, the quadratic convergence of Newton's method for self-concordant functions is in terms of the Newton decrement $\lambda_n(\mathbf{w})$ and it does not necessary guarantee quadratic convergence in terms of objective function error. To be more precise, we can show that $\lambda_n(\mathbf{w}_n) \leq \gamma \lambda_n(\mathbf{w}_m)^2$; however, we can not conclude that the quadratic convergence of Newton's method implies $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq \gamma'(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*))^2$. In the following proposition we try to characterize an upper bound for the error $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ in terms of the squared error $(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*))^2$ using the quadratic convergence property of Newton decrement.

Proposition 12 *Consider* $\mathbf{w}_m$ *as a variable that is in the local neighborhood of the optimal argument of the risk* R_n *where Newton's method has a quadratic convergence rate, i.e.,* $\lambda_n(\mathbf{w}_m) \leq 1/4$. *Recall the definition of the variable* $\mathbf{w}_n$ *in* (9.4) *as the updated variable using Newton step. If Assumption 18 and 19 hold, then the difference* $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ *is upper bounded by*

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq 144(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*))^2. \tag{9.24}$$

Proof: To prove the result in (9.24) first we need to find upper and lower bounds for the difference $R_n(\mathbf{w}) - R_n(\mathbf{w}_n^*)$ in terms of the Newton decrement parameter $\lambda_n(\mathbf{w})$. To do so, we use the result in Theorem 4.1.11 of [83] which shows that

$$\lambda_n(\mathbf{w}) - \ln(1 + \lambda_n(\mathbf{w})) \leq R_n(\mathbf{w}) - R_n(\mathbf{w}_n^*) \leq -\lambda_n(\mathbf{w}) - \ln(1 - \lambda_n(\mathbf{w})). \tag{9.25}$$

Note that we assume that $0 < \lambda_n(\mathbf{w}) < 1/4$. Thus, we can use the Taylor's expansion of $\ln(1 + a)$ for $a = \lambda_n(\mathbf{w})$ to show that $\lambda_n(\mathbf{w}) - \ln(1 + \lambda_n(\mathbf{w}))$ is bounded below by $(1/2)\lambda_n(\mathbf{w})^2 - (1/3)\lambda_n(\mathbf{w})^3$. Since $0 < \lambda_n(\mathbf{w}) < 1/4$ we can show that $(1/6)\lambda_n(\mathbf{w})^2 \leq (1/2)\lambda_n(\mathbf{w})^2 - (1/3)\lambda_n(\mathbf{w})^3$. Thus, the term $\lambda_n(\mathbf{w}) - \ln(1 + \lambda_n(\mathbf{w}))$ is bounded below by $(1/6)\lambda^2$. Likewise, we use Taylor's expansion of $\ln(1 - a)$ for $a = \lambda_n(\mathbf{w})$ to show that $-\lambda_n(\mathbf{w}) - \ln(1 - \lambda_n(\mathbf{w}))$ is bounded above by $\lambda_n(\mathbf{w})^2$ for $\lambda_n(\mathbf{w}) < 1/4$; see e.g., Chapter 9 of [20]. Considering these bounds and the inequalities in (9.25) we can write

$$\frac{1}{6}\lambda_n(\mathbf{w})^2 \leq R_n(\mathbf{w}) - R_n(\mathbf{w}_n^*) \leq \lambda_n(\mathbf{w})^2. \tag{9.26}$$

Recall that the variable $\mathbf{w}_m$ satisfies the condition $\lambda_n(\mathbf{w}_m) \leq 1/4$. Thus, according to the quadratic convergence rate of Newton's method for self-concordant functions [20], we know that the Newton decrement has a quadratic convergence and we can write

$$\lambda_n(\mathbf{w}_n) \leq 2\lambda_n(\mathbf{w}_m)^2. \tag{9.27}$$

We use the result in (9.26) and (9.27) to show that the optimality error $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ has an upper bound which is proportional to $(R_n(w_m) - R_n(\mathbf{w}_n^*))^2$. In particular, we can write $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq \lambda_n(\mathbf{w}_n)^2$ based on the second inequality in (9.26). This observation in conjunction with the result in (9.27) implies that

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq 4\lambda_n(\mathbf{w}_m)^4. \tag{9.28}$$

The first inequality in (9.26) yields $\lambda_n(\mathbf{w}_m)^4 \leq 36(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*))^2$. Thus, we can substitute $\lambda_n(\mathbf{w}_m)^4$ in (9.28) by $36(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*))^2$ to obtain the result in (9.24). ■

The result in Proposition 12 provides an upper bound for the sub-optimality $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ in terms of the sub-optimality of variable $\mathbf{w}_m$ for the risk R_n, i.e., $R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)$. Recall that we know that $\mathbf{w}_m$ is in the statistical accuracy of R_m, i.e., $R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \leq V_m$, and we aim to show that the updated variable $\mathbf{w}_n$ stays in the statistical accuracy of R_n, i.e., $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$. This can be done by showing that the upper bound for $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ in (9.24) is smaller than V_n. We proceed to derive an upper bound for the sub-optimality $R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)$ in the following proposition.

Proposition 13 *Consider the sets $\mathcal{S}_m$ and $\mathcal{S}_n$ as subsets of the training set $\mathcal{T}$ such that $\mathcal{S}_m \subset \mathcal{S}_n \subset \mathcal{T}$. We assume that the number of samples in the sets $\mathcal{S}_m$ and $\mathcal{S}_n$ are m and n, respectively. Further, define $\mathbf{w}_m$ as an V_m optimal solution of the risk R_m, i.e., $R_m(\mathbf{w}_m) - R_m^* \leq V_m$. If Assumption 18-20 hold, then the empirical risk error $R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)$ of the variable $\mathbf{w}_m$ corresponding to the set $\mathcal{S}_n$ is bounded above by*

$$R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*) \leq V_m + \frac{2(n-m)}{n}\left(V_{n-m} + V_m\right) + 2\left(V_m - V_n\right) + \frac{c(V_m - V_n)}{2}\|\mathbf{w}^*\|^2 \quad w.h.p. \tag{9.29}$$

Proof: Note that the difference $R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)$ can be written as

$$\begin{aligned} R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*) = {} & R_n(\mathbf{w}_m) - R_m(\mathbf{w}_m) + R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \\ & + R_m(\mathbf{w}_m^*) - R_m(\mathbf{w}_n^*) + R_m(\mathbf{w}_n^*) - R_n(\mathbf{w}_n^*). \end{aligned} \tag{9.30}$$

We proceed to bound the differences in (C.11). To do so, note that the difference $R_n(\mathbf{w}_m) - R_m(\mathbf{w}_m)$ can be simplified as

$$\begin{aligned} R_n(\mathbf{w}_m) - R_m(\mathbf{w}_m) &= L_n(\mathbf{w}_m) - L_m(\mathbf{w}_m) + \frac{c(V_n - V_m)}{2}\|\mathbf{w}_m\|^2 \\ &\leq L_n(\mathbf{w}) - L_m(\mathbf{w}), \end{aligned} \tag{9.31}$$

where the inequality follows from the fact that $V_n < V_m$ and $V_n - V_m$ is negative. It follows from the result in Lemma 26 that the right hand side of (C.12) is bounded by $(n-m)/n\,(V_{n-m} + V_m)$. Therefore,

$$R_n(\mathbf{w}_m) - R_m(\mathbf{w}_m) \leq \frac{n-m}{n}\left(V_{n-m} + V_m\right). \tag{9.32}$$

According to the fact that $\mathbf{w}_m$ as an V_m optimal solution for the sub-optimality $R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*)$ we know that

$$R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \leq V_m. \tag{9.33}$$

Based on the definition of $\mathbf{w}_m^*$ which is the optimal solution of the risk R_m, the third difference in (C.11) which is $R_m(\mathbf{w}_m^*) - R_m(\mathbf{w}_n^*)$ is always negative. I.e.,

$$R_m(\mathbf{w}_m^*) - R_m(\mathbf{w}_n^*) \leq 0. \tag{9.34}$$

Moreover, we can use the triangle inequality to bound the difference $R_m(\mathbf{w}_n^*) - R_n(\mathbf{w}_n^*)$ in (C.11) as

$$\begin{aligned} R_m(\mathbf{w}_n^*) - R_n(\mathbf{w}_n^*) &= L_m(\mathbf{w}_n^*) - L_n(\mathbf{w}_n^*) + \frac{c(V_m - V_n)}{2}\|\mathbf{w}_n^*\|^2 \\ &\leq \frac{n-m}{n}\left(V_{n-m} + V_m\right) + \frac{c(V_m - V_n)}{2}\|\mathbf{w}_n^*\|^2. \end{aligned} \tag{9.35}$$

Replacing the differences in (C.11) by the upper bounds in (C.13)-(C.16) follows

$$R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*) \leq V_m + \frac{2(n-m)}{n}\left(V_{n-m} + V_m\right) + \frac{c(V_m - V_n)}{2}\|\mathbf{w}_n^*\|^2 \quad \text{w.h.p.} \tag{9.36}$$

Substitute $\|\mathbf{w}_n^*\|^2$ in (C.17) by the upper bound in (9.13) to obtain the result in (9.29). ■

The result in Proposition 13 characterizes the sub-optimality of the variable $\mathbf{w}_m$, which is an V_m sub-optimal solution for the risk R_m, with respect to the empirical risk R_n associated with the set $\mathcal{S}_n$.

The results in Proposition 11, 12, and 13 lead to the result in Theorem 16. To be more precise, from the result in Proposition 11 we obtain that the condition in (9.7) implies that $\mathbf{w}_m$ is in the local neighborhood of the optimal argument of R_n and $\lambda_n(\mathbf{w}_m) \leq 1/4$. Hence, the hypothesis of Proposition 12 is satisfied and we have $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq 144(R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*))^2$. This result paired with the result in Proposition 13 shows that if the condition in (9.8) is satisfied we can conclude that $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$ which completes the proof of Theorem 16.

9.4 Experiments

In this section, we study the performance of Ada Newton and compare it with state-of-the-art in solving a large-scale classification problem. In the main paper we only use the protein homology dataset provided on KDD cup 2004 website. Further numerical experiments on various datasets can be found in Section 7.4 in the supplementary material. The protein homology dataset contains $N = 145751$ samples and the dimension of each sample is $p = 74$. We consider three algorithms to compare with the proposed Ada Newton method. One of them is the classic Newton's method with backtracking line search. The second algorithm is Stochastic Gradient Descent (SGD) and the last one is the SAGA method introduced

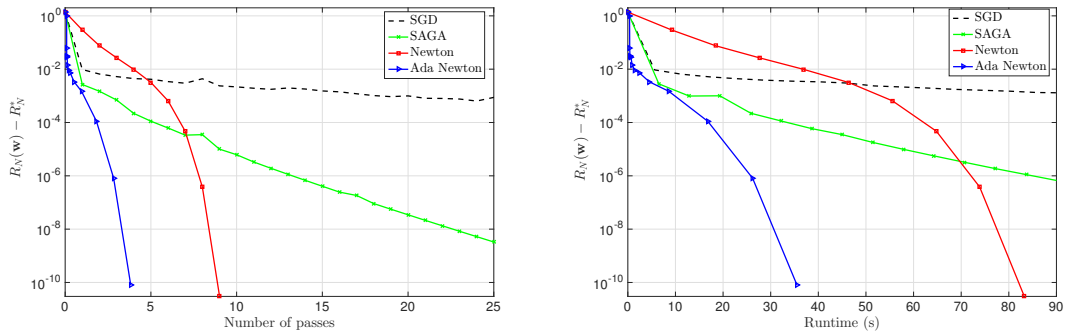


Figure 9.1: Comparison of SGD, SAGA, Newton, and Ada Newton in terms of number of effective passes over dataset (left) and runtime (right) for the protein homology dataset.

in [31]. In our experiments, we use logistic loss and set the regularization parameters as $c = 200$ and $V_n = 1/n$.

The stepsize of SGD in our experiments is 2×10^{-2}. Note that picking larger stepsize leads to faster but less accurate convergence and choosing smaller stepsize improves the accuracy convergence with the price of slower convergence rate. The stepsize for SAGA is hand-optimized and the best performance has been observed for $\alpha = 0.2$ which is the one that we use in the experiments. For Newton's method, the backtracking line search parameters are $\alpha = 0.4$ and $\beta = 0.5$. In the implementation of Ada Newton we increase the size of the training set by factor 2 at each iteration, i.e., $\alpha = 2$ and we observe that the condition $\|\nabla R_n(\mathbf{w}_n)\| < (\sqrt{2c})V_n$ is always satisfied and there is no need for reducing the factor α. Moreover, the size of initial training set is $m_0 = 124$. For the warmup step that we need to get into to the quadratic neighborhood of Newton's method we use the gradient descent method. In particular, we run gradient descent with stepsize 10^{-3} for 100 iterations. Note that since the number of samples is very small at the beginning, $m_0 = 124$, and the regularizer is very large, the condition number of problem is very small. Thus, gradient descent is able to converge to a good neighborhood of the optimal solution in a reasonable time. Notice that the computation of this warm up process is very low and is equal to 12400 gradient evaluations. This number of samples is less than 10% of the full training set. In other words, the cost is less than 10% of one pass over the dataset. Although, this cost is negligible, we consider it in comparison with SGD, SAGA, and Newton's method. We would like to mention that other algorithms such as Newton's method and stochastic algorithms can also be used for the warm up process; however, the gradient descent method sounds the best option since the gradient evaluation is not costly and the problem is well-conditioned for a small training set .

The left plot in Figure 9.1 illustrates the convergence path of SGD, SAGA, Newton, and Ada Newton for the protein homology dataset. Note that the x axis is the total number of samples used divided by the size of the training set $N = 145751$ which we call number of

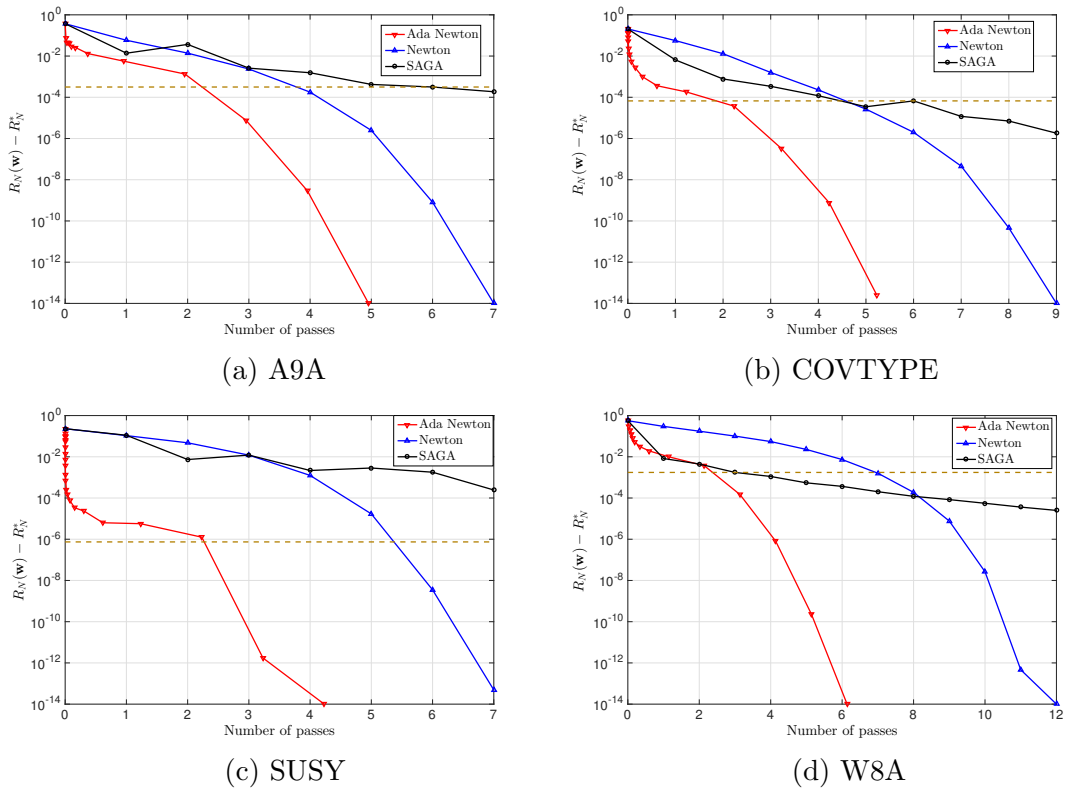
Table 9.1: Summary of the datasets

Dataset	Number of Samples	Number of Features
A9A	32561	123
W8A	49749	300
COVTYPE.BINARY	581012	54
SUSY	5000000	18

passes over the dataset. As we observe, The best performance among the four algorithms belongs to Ada Newton. In particular, Ada Newton is able to achieve the accuracy of $R_N(\mathbf{w}) - R_N^* < 1/N$ by 2.4 passes over the dataset which is very close to theoretical result in Theorem 1 that guarantees accuracy of order $O(1/N)$ after $\alpha/(\alpha - 1) = 2$ passes over the dataset. To achieve the same accuracy of $1/N$ Newton's method requires 7.5 passes over the dataset, while SAGA needs 10 passes. The SGD algorithm can not achieve the statistical accuracy of order $O(1/N)$ even after 25 passes over the dataset.

Although, Ada Newton and Newton outperform SAGA and SGD, their computational complexity are different. We address this concern by comparing the algorithms in terms of runtime. The right plot in Figure 9.1 demonstrates the convergence paths of the considered methods in terms of runtime. As we observe, Newton's method requires more time to achieve the statistical accuracy of $1/N$ relative to SAGA. This observation justifies the belief that Newton's method is not practical for large-scale optimization problems, since by enlarging p or making the initial solution worse the performance of Newton's method will be even worse than the ones in Figure 9.1. Ada Newton resolves this issue by starting from small sample size which is computationally less costly. Ada Newton also requires Hessian inverse evaluations, but the number of inversions is proportional to $\log_\alpha N$. Moreover, the performance of Ada Newton doesn't depend on the initial point and the warm up process is not costly as we described before. We observe that Ada Newton outperforms SAGA significantly. In particular it achieves the statistical accuracy of $1/N$ in less than 25 seconds, while SAGA achieves the same accuracy in 62 seconds. Note that since the variable $\mathbf{w}_N$ is in the quadratic neighborhood of Newton's method for R_N the convergence path of Ada Newton becomes quadratic eventually when the size of the training set becomes equal to the size of the full dataset. It follows that the advantage of Ada Newton with respect to SAGA is more significant if we look for a sub-optimality less than V_n.

We further compare the performances of these methods for other real datasets such as A9A, W8A, COVTYPE, and SUSY. These datasets have different size and dimensionality as stated in Table 9.1. In these experiments, we use 90% of samples of the data points as the training set and the remaining 10% as the test set. The stepsize for SAGA is set as $1/L$ as suggested in [31].



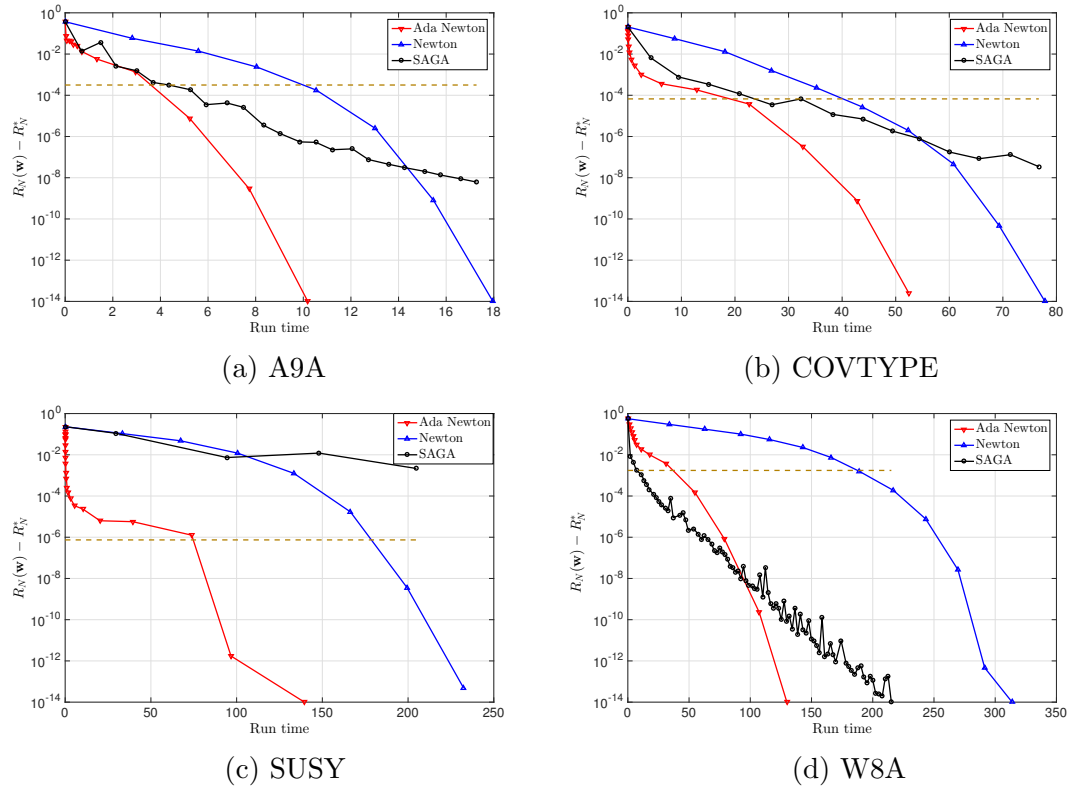
(a) A9A (b) COVTYPE

(c) SUSY (d) W8A

Figure 9.2: Comparison of the sub-optimality of SAGA, Newton, and Ada Newton in terms of number of effective passes over dataset for four datasets. The horizontal axis represents the number of effective passes over the training set and the vertical axis shows the sub-optimality error $R_N(\mathbf{w}) - R_N^*$ where N is the size of training set. The dotted horizontal line refers to statistical accuracy.

Figure 9.2 illustrates the sub-optimality $R_N(\mathbf{w}) - R_N^*$ of these methods versus the number of passes over the datasets. In order to connect convergence on the empirical and expected risks, we plot the a horizontal dotted green line that shows the iteration at which Ada Newton reached convergence on the test set. As we observe, Ada Newton achieves statistical accuracy (the green line) after almost 2 passes over the training set for all the considered datasets. This observation matches the expectation from the theoretical guarantees in this chapter that Ada Newton should achieve the statistical accuracy of the full training set after almost two passes over the dataset.

Since the computational complexity of SAGA is lower than the ones for Newton's method and Ada Newton, we also compare these methods in terms of runtime. Figure 9.3 demonstrates the sub-optimality of these methods versus their runtimes. This comparison justifies that the Newton's method is impractical for large scale ERM minimization, and Ada Newton significantly improves the performance of Newton's method. Indeed, the gap between Ada Newton and SAGA is less significant comparing to the plots in Figure 9.2, but still the



(a) A9A (b) COVTYPE

(c) SUSY (d) W8A

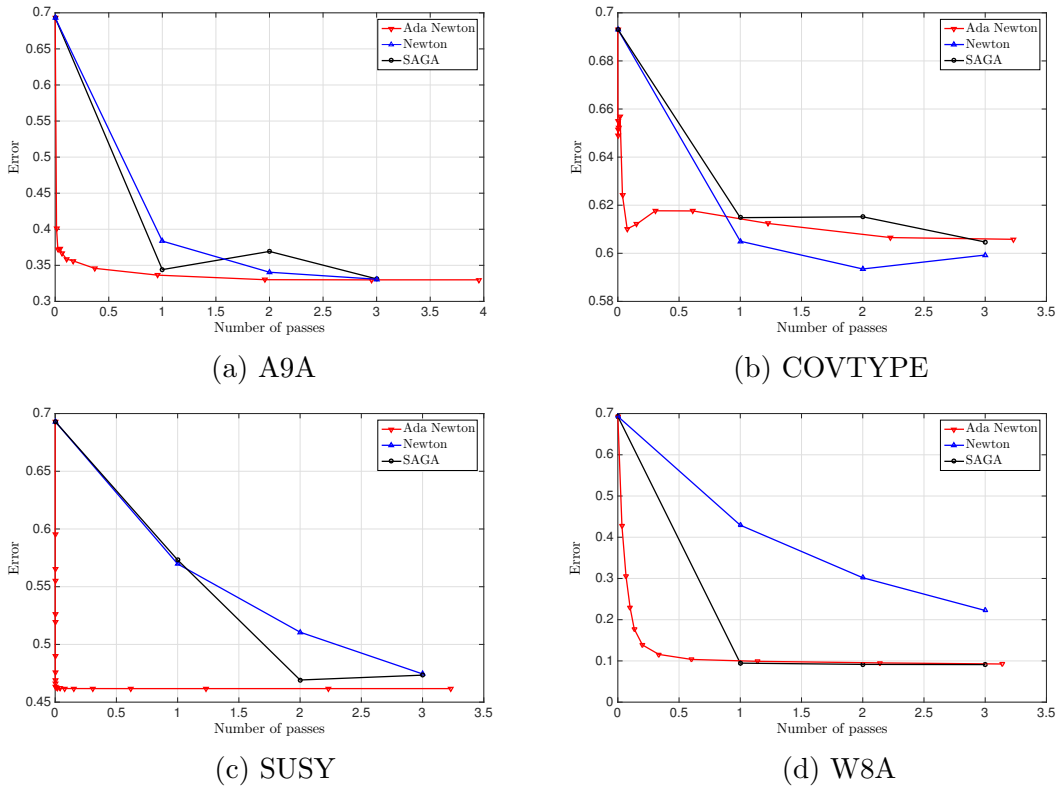
Figure 9.3: Comparison of the sub-optimality of SAGA, Newton, and Ada Newton in terms of run time for four datasets. The horizontal axis represents runtime and the vertical axis shows the sub-optimality error $R_N(\mathbf{w}) - R_N^*$ where N is the size of training set. The dotted horizontal line refers to statistical accuracy.

advantage of Ada Newton relative to SAGA is clear.

We further present the expected error of classifiers trained by SAGA, Newton, and Ada Newton on the test set of each of the considered datasets in Figure 9.4. The results showcase that in all experiments Ada Newton achieves a target test error faster than Newton's method and SAGA.

9.5 Discussions

As explained in Section 9.3, Theorem 16 holds because condition (9.7) makes $\mathbf{w}_m$ part of the quadratic convergence region of R_n. From this fact, it follows that the Newton iteration makes the suboptimality gap $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ the square of the suboptimality gap $R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)$. This yields condition (9.8) and is the fact that makes Newton steps valuable in increasing the sample size. If we replace Newton iterations by any method with linear convergence rate, the orders of both sides on condition (9.8) are the same. This would make aggressive increase of the sample size unlikely.



(a) A9A (b) COVTYPE

(c) SUSY (d) W8A

Figure 9.4: Comparison of the test error of SAGA, Newton, and Ada Newton in terms of number of effective passes over the dataset for four datasets. The horizontal axis represents the number of effective passes over the training set and the vertical axis shows the error on the test set.

In Section 9.1 we pointed out four reasons that challenge the development of stochastic Newton methods. It would not be entirely accurate to call Ada Newton a stochastic method because it doesn't rely on stochastic descent directions. It is, nonetheless, a method for ERM that makes pithy use of the dataset. The challenges listed in Section **??** are overcome by Ada Newton because:

(i) Ada Newton does not use line searches. Optimality improvement is guaranteed by increasing the sample size.

(ii) The advantages of Newton's method are exploited by increasing the sample size at a rate that keeps the solution for sample size m in the quadratic convergence region of the risk associated with sample size $n = \alpha m$. This allows aggressive growth of the sample size.

(iii) The ERM problem is not necessarily strongly convex. A regularization of order V_n is added to construct the empirical risk R_n

(iv) Ada Newton inverts approximately $\log_\alpha N$ Hessians. To be more precise, the total number of inversion could be larger than $\log_\alpha N$ because of the backtracking step. However, the backtracking step is bypassed when the number of samples is sufficiently large.

It is fair to point out that items (ii) and (iv) are true only to the extent that the damped phase in Algorithm 11 is not significant. Our numerical experiments indicate that this is true but the conclusion is not warranted by out theoretical bounds except when the dataset is very large. This suggests the bounds are loose and that further research is warranted to develop tighter bounds.

Chapter 10

Conclusions

In the first part of the thesis, which contains Chapters 2-4, we focused on the use of stochastic methods, which operate on a subset of samples at each iteration, to solve-large scale empirical risk minimization problems. In particular, we focused on the application of quasi-Newton methods in stochastic settings for accelerating the convergence rate of state-of-the-art first-order stochastic methods.

In Chapter 2, we studied the reasons that stochastic gradient descent methods are slow in ill-condition problems. Further, in detail, we explained the challenges in designing stochastic quasi-Newton methods and, in particular, the issue of Hessian approximation matrices singularity. To overcome these challenges RES, a stochastic implementation of a regularized version of the Broyden-Fletcher-Goldfarb-Shanno quasi-Newton method was introduced to find corresponding optimal arguments. RES resolve the singularity issue by modifying the proximity condition in the BFGS update for Hessian approximation matrices such that the matrices always stay positive definite, while they satisfy secant condition which is the fundamental property of quasi-Newton Hessian approximation matrices. Almost sure convergence of the sequence generated by RES was established under the assumption that sample functions have well behaved Hessians. A sublinear convergence rate in expectation was further proven. Numerical results showed that RES affords important reductions in terms of convergence time relative to stochastic gradient descent. These reductions are of particular significance for problems with large condition numbers or large dimensionality since RES exhibits remarkable stability in terms of the total number of iterations required to achieve target accuracies. An application of RES to support vector machines was also developed. In this particular case the advantages of RES manifest in improvements of classification accuracies for training sets of fixed cardinality.

In Chapter 3, we turned our attention to a limited memory version of stochastic (online) BFGS method to reduce the high computational complexity of RES which is the outcome of inverting the Hessian approximation matrices at each iteration. In particular, an on-

line limited memory version of the (oL)BFGS algorithm was studied for solving strongly convex optimization problems with stochastic objectives. Almost sure convergence was established by bounding the traces and determinants of curvature estimation matrices under the assumption that sample functions have well behaved Hessians. The convergence rate of oLBFGS was further determined to be at least linear in expectation. This rate is customary of stochastic optimization algorithms which are limited by their ability to smooth out the noise in stochastic gradient estimates. The application of oLBFGS to support vector machines was also developed and numerical tests on synthetic data were provided. The numerical results show that oLBFGS affords important reductions with respect to stochastic gradient descent (SGD) in terms of the number of feature vectors that need to be processed to achieve a target accuracy as well as in the associated execution time. Moreover, oLBFGS also exhibits a significant execution time reduction when compared to other stochastic quasi-Newton methods. These reductions increase with the problem dimension and can become arbitrarily large. A detailed comparison between oLBFGS and SGD for training a logistic regressor in a large scale search engine advertising problem was also presented. The numerical tests show that oLBFGS trains the regressor using less than 1% of the data required by SGD to obtain similar classification accuracy.

Chapter 4, completed the work in Chapters 2 and 3 by introducing an incremental quasi-Newton BFGS method to solve a large scale optimization problem, in which an aggregate cost function is minimized while computing only a single gradient and Hessian approximation per iteration. The presented IQN method has three fundamental properties which makes it distinct from state-of-the-art incremental (stochastic) quasi-Newton methods. First, IQN uses the aggregated information of variables, gradients, and Hessian approximation matrices to reduce the noise of approximation for both gradients and Hessian approximation matrices. Second, in IQN the index of the updated function is chosen in a cyclic fashion, rather than the random selection scheme used in known incremental methods. Third, IQN utilizes a consistent Taylor series which yields a more involved update. These three properties together lead to an incremental quasi-Newton method with a local superlinear convergence rate. In particular, the convergence analysis of IQN indicates the local superlinear convergence of the sequence of residuals with respect to the average sequence. Moreover, it was shown that there exists a superlinearly convergent sequence that is an upper bound for the original sequence of errors, which implies superlinear convergence of a subsequence of residuals generated by IQN iterates. Numerical experiments on synthetic and real datasets verified superior performance of IQN relative to first-order incremental methods.

The focus of the second part of the thesis, which includes Chapters 5-7, was on the idea of distributing samples over multiple processors to reduce the computational burden at each

processor for solving big-data empirical risk minimization problems. We explored this approach, which we referred to as decentralized optimization, by presenting efficient methods that solve this distributed optimization problem in a communication efficient manner.

In Chapter 5, we presented the network Newton method as an approximate Newton method for solving consensus optimization problems. The algorithm builds on a reinterpretation of distributed gradient descent as a penalty method and relies on an approximation of the Newton step of the corresponding penalized objective function. To approximate the Newton direction we truncate the Taylor series of the exact Newton step. This leads to a family of methods defined by the number K of Taylor series terms kept in the approximation. When we keep K terms of the Taylor series, the method is called NN-K and can be implemented through the aggregation of information in K-hop neighborhoods. We showed that NN converges at least linearly to the solution of the penalized objective, and, consequently, to a neighborhood of the optimal argument for the original optimization problem. We completed the convergence analysis of NN-K by showing that the sequence of iterates generated by NN-K converges at a quadratic rate in a specific interval. Numerical analyses compared the performances of NN-K with different choices of K for minimizing quadratic objectives. We observed that all NN-K methods work faster than distributed gradient descent in terms of number of iterations and number of communications. We also analyzed a tradeoff on the choice of a penalty parameter that controls both, the accuracy of the optimal objective computed by network Newton methods and the rate of convergence. We proposed an adaptive version of network Newton (ANN) that achieves exact convergence by executing network Newton with an increasing sequence of penalty coefficients. Numerical analyses of ANN show that it is best to initialize penalty coefficients at moderate values and decrease them through moderate factors.

In Chapter 6, we proposed an Exact Second-Order Method (ESOM) that converges to the optimal argument of the global objective function at a linear rate. We developed the update of ESOM by substituting the primal update of Proximal Method of Multipliers (PMM) with its second order approximation. Moreover, we approximated the Hessian inverse of the proximal augmented Lagrangian by truncating its Taylor's series. This approximation leads to a class of algorithms ESOM-K where $K+1$ indicates the number of Taylor's series terms that are used for Hessian inverse approximation. Convergence analysis of ESOM-K shows that the sequence of iterates converges to the optimal argument linearly irrespective to the choice of K. We showed that the linear convergence factor of ESOM-K is a function of time and the choice of K. The linear convergence factor of ESOM approaches the linear convergence factor of PMM as time passes. Moreover, larger choice of K makes the factor of linear convergence for ESOM closer to the one for PMM. Numerical results verify the theoretical linear convergence and the relation between the linear convergence factor

of ESOM-K and PMM. Further, we observed that larger choice of K for ESOM-K leads to faster convergence in terms of number of iterations, while the most efficient version of ESOM-K in terms of communication cost is ESOM-0.

In Chapter 7, the decentralized double stochastic averaging gradient (DSA) was presented as an algorithm for solving decentralized optimization problems where the local functions can be written as an average of a set of local instantaneous functions. DSA exploits stochastic averaging gradients in lieu of gradients and mixes information of two consecutive iterates to determine the descent direction. By assuming strongly convex local instantaneous functions with Lipschitz continuous gradients, the DSA algorithm converges linearly to the optimal arguments in expectation. In addition, the sequence of local iterates $\mathbf{w}_v^t$ for each node in the network almost surely converges to the optimal argument $\tilde{\mathbf{w}}^*$. A comparison between the DSA algorithm and a group of stochastic and deterministic alternatives are provided for solving a logistic regression problem. The numerical results show DSA is the only stochastic decentralized algorithm to reach linear convergence. DSA outperforms decentralized stochastic alternatives in terms of number of required iteration for convergence, and exhibits faster convergence relative to deterministic alternatives in terms of number feature vectors processed until convergence. Effect of number of samples, number of nodes in the network, condition of the objective function, and condition number of graph on the convergence rate of DSA were also studied this chapter.

The third and last part of the thesis, which includes Chapters 8 and 9, is on solving empirical risk minimization via an adaptive sample size scheme. The main idea of the proposed adaptive sample size mechanism is to start with a small number of samples and increase the size of the training set geometrically at each step. Since the functions are driven from a common distribution the solution for the smaller empirical risk minimization problems is a good estimate for the empirical risk minimization problem corresponding to the enlarged training set.

In Chapter 8, we first explained the two fundamental properties of ERM. The first property is that since the empirical risk and the statistical loss have different minimizers, there is no reason to solve ERM beyond the expected difference between the two objectives. The second important property of ERM is that the component functions are drawn from a common distribution. This implies that if we consider subsets of the training set, the respective empirical risk functions are not that different from each other and, indeed, their differences are related to the statistical accuracy of the subset. We presented adaptive sample size scheme and highlighted how this approach exploits these two peculiar features of ERM and leads to better convergence guarantees. In particular, we showed that to reach the statistical accuracy of the full training set the adaptive sample size scheme reduces the overall computational complexity of a broad range of first-order methods by a logarithmic factor

of the inverse of statistical accuracy. This improvement led to the best known convergence complexity for solving ERM problems among first-order methods, which was achieved by adaptive sample size SVRG algorithm.

In Chapter 9, we extended the idea of adaptive sample size methods to Newton's method which enjoys from a local quadratic convergence rate. In the presented adaptive sample size Newton method (Ada Newton) the sample size is increased geometrically by a factor $\alpha > 1$. The main advantage of Ada Newton relative to adaptive sample size first-order methods is that at each step it only requires a single Newton iteration with unit stepsize to solve the extended ERM problem to its statistical accuracy. As we highlighted in the convergence analysis of Ada Newton, this fascinating behavior happens by increasing the size of the training in a way that the minimization variable for the current training set is in the local neighborhood of the optimal argument of the next training set. This allows to exploit the quadratic convergence property of Newton's method and reach the statistical accuracy of each training set with only one iteration of Newton's method. We showed in this chapter both theoretically that we can iteratively increase the sample size while applying single Newton iterations without line search and staying within the statistical accuracy of the regularized empirical risk. In particular, we can double the size of the training set in each iteration when the number of samples is sufficiently large. Numerical experiments on various datasets confirm the possibility of increasing the sample size by factor 2 at each iteration which implies that Ada Newton achieves the statistical accuracy of the full training set with about two passes over the dataset.

Appendix A

Appendix

A.1 Proof of Lemma 3

Proof: We prove (2.65) using induction. To prove the claim for $t = 0$ simply observe that the definition of Q in (2.66) implies that

$$Q := \max\left[\frac{b}{c-1},\ t_0 u_0\right] \geq\ t_0 u_0, \tag{A.1}$$

because the maximum of two numbers is at least equal to both of them. By rearranging the terms in (A.1) we can conclude that

$$u_0\ \leq\ Q/t_0. \tag{A.2}$$

Comparing (A.2) and (2.65) it follows that the latter inequality is true for $t = 0$.

Introduce now the induction hypothesis that (2.65) is true for $t = s$. To show that this implies that (2.65) is also true for $t = s + 1$ substitute the induction hypothesis $u_s \leq Q/(s + t_0)$ into the recursive relationship in (2.64). This substitution shows that u_{s+1} is bounded as

$$u_{s+1} \leq \left(1 - \frac{c}{s+t_0}\right)\frac{Q}{s+t_0} + \frac{b}{(s+t_0)^2}\ . \tag{A.3}$$

Observe now that according to the definition of Q in (2.66), we know that $b/(c-1) \leq Q$ because Q is the maximum of $b/(c-1)$ and $t_0 u_0$. Reorder this bound to show that $b \leq Q(c-1)$ and substitute into (A.3) to write

$$u_{s+1} \leq \left(1 - \frac{c}{s+t_0}\right)\frac{Q}{s+t_0} + \frac{(c-1)Q}{(s+t_0)^2}\ . \tag{A.4}$$

Pulling out $Q/(s+t_0)^2$ as a common factor and simplifying and reordering terms it follows that (A.4) is equivalent to

$$u_{s+1}\ \leq\ \frac{Q\big[s + t_0 - c + (c-1)\big]}{(s+t_0)^2}\ =\ \frac{s+t_0-1}{(s+t_0)^2}Q. \tag{A.5}$$

To complete the induction step use the difference of squares formula for $(s+t_0)^2 - 1$ to conclude that

$$\big[(s+t_0)-1\big]\big[(s+t_0)+1\big]\ =\ (s+t_0)^2 - 1\ \leq\ (s+t_0)^2. \tag{A.6}$$

Reordering terms in (A.6) it follows that $\big[(s+t_0)-1\big]/(s+t_0)^2 \leq 1/\big[(s+t_0)+1\big]$, which

upon substitution into (A.5) leads to the conclusion that

$$u_{s+1} \leq \frac{Q}{s + t_0 + 1}. \tag{A.7}$$

Eq. (A.7) implies that the assumed validity of (2.65) for $t = s$ implies the validity of (2.65) for $t = s + 1$. Combined with the validity of (2.65) for $t = 0$, which was already proved, it follows that (2.65) is true for all times $t \geq 0$. ■

Appendix B

Appendix

B.1 Proof of Theorem 7

To prove global convergence of the Network Newton method we first introduce two technical lemmas. In the first lemma, we develop an upper bound for the objective function value $F(\mathbf{y})$ using the first three terms of its Taylor expansion. In the second lemma, we construct an upper bound for the error $F(\mathbf{y}_{t+1}) - F(\mathbf{y}^*)$ in terms of $F(\mathbf{y}_t) - F(\mathbf{y}^*)$.

Lemma 28 *Consider the function $F(\mathbf{y})$ defined in* (5.6)*. If Assumptions 10 and 11 hold, then for any $\mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^{np}$*

$$F(\hat{\mathbf{y}}) \leq F(\mathbf{y}) + \nabla F(\mathbf{y})^T(\hat{\mathbf{y}} - \mathbf{y}) + \frac{1}{2}(\hat{\mathbf{y}} - \mathbf{y})^T \nabla^2 F(\mathbf{y})(\hat{\mathbf{y}} - \mathbf{y}) + \frac{\alpha L}{6}\|\hat{\mathbf{y}} - \mathbf{y}\|^3. \tag{B.1}$$

Proof: The claim follows from the Lipschitz continuity of the Hessian with constant αL and Theorem 7.7 in [2] which characterizes the error of taylor's expansion. ■

In the following lemma, we use the result in Lemma 28 to establish an upper bound for the error $F(\mathbf{y}_{t+1}) - F(\mathbf{y}^*)$.

Lemma 29 *Consider the NN-K method as defined in* (5.12)-(5.17)*. Further, recall the definition of $\mathbf{y}^*$ as the optimal argument of the objective function $F(\mathbf{y})$. If Assumptions 9-11 hold, then*

$$F(\mathbf{y}_{t+1}) - F(\mathbf{y}^*) \leq \left[1 - \left(2\epsilon - \epsilon^2\right)\alpha m \lambda\right][F(\mathbf{y}_t) - F(\mathbf{y}^*)] + \frac{\alpha L \epsilon^3 \Lambda^3}{6\lambda^{\frac{3}{2}}}[F(\mathbf{y}_t) - F(\mathbf{y}^*)]^{\frac{3}{2}}. \tag{B.2}$$

Proof: By setting $\hat{\mathbf{y}} := \mathbf{y}_{t+1}$ and $\mathbf{y} := \mathbf{y}_t$ in (B.1) we obtain

$$F(\mathbf{y}_{t+1}) \leq F(\mathbf{y}_t) + \mathbf{g}_t^T(\mathbf{y}_{t+1} - \mathbf{y}_t) + \frac{1}{2}(\mathbf{y}_{t+1} - \mathbf{y}_t)^T \mathbf{H}_t(\mathbf{y}_{t+1} - \mathbf{y}_t) + \frac{\alpha L}{6}\|\mathbf{y}_{t+1} - \mathbf{y}_t\|^3, \tag{B.3}$$

where $\mathbf{g}_t := \nabla F(\mathbf{y}_t)$ and $\mathbf{H}_t := \nabla^2 F(\mathbf{y}_t)$. From the definition of the NN-K update in (5.17) we can write the difference of two consecutive variables as $\mathbf{y}_{t+1} - \mathbf{y}_t = -\epsilon \hat{\mathbf{H}}_t^{-1}\mathbf{g}_t$. Making this substitution into (B.3) implies

$$F(\mathbf{y}_{t+1}) \leq F(\mathbf{y}_t) - \epsilon \mathbf{g}_t^T \hat{\mathbf{H}}_t^{-1}\mathbf{g}_t + \frac{\epsilon^2}{2}\mathbf{g}_t^T \hat{\mathbf{H}}_t^{-1}\mathbf{H}_t\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t + \frac{\alpha L \epsilon^3}{6}\|\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\|^3. \tag{B.4}$$

According to (5.46), we can substitute $\hat{\mathbf{H}}_t^{-1/2}\mathbf{H}_t\hat{\mathbf{H}}_t^{-1/2}$ in (B.4) by $\mathbf{I} - \mathbf{E}_t$ which leads to

$$F(\mathbf{y}_{t+1}) \leq F(\mathbf{y}_t) - \epsilon \mathbf{g}_t^T \hat{\mathbf{H}}_t^{-1}\mathbf{g}_t + \frac{\epsilon^2}{2}\mathbf{g}_t^T \hat{\mathbf{H}}_t^{-\frac{1}{2}}(\mathbf{I} - \mathbf{E}_t)\hat{\mathbf{H}}_t^{-\frac{1}{2}}\mathbf{g}_t + \frac{\alpha L \epsilon^3}{6}\|\hat{\mathbf{H}}_t^{-1}\mathbf{g}_t\|^3. \tag{B.5}$$

Proposition 6 shows that $\mathbf{E}_t$ is positive semidefinite, and, therefore, the quadratic form $\mathbf{g}_t^T \hat{\mathbf{H}}_t^{-1/2} \mathbf{E}_t \hat{\mathbf{H}}_t^{-1/2} \mathbf{g}_t$ is nonnegative. Considering this lower bound we can simplify (B.5) to

$$F(\mathbf{y}_{t+1}) \leq F(\mathbf{y}_t) - \frac{\left(2\epsilon - \epsilon^2\right)}{2} \mathbf{g}_t^T \hat{\mathbf{H}}_t^{-1} \mathbf{g}_t + \frac{\alpha L \epsilon^3}{6} \|\hat{\mathbf{H}}_t^{-1} \mathbf{g}_t\|^3. \tag{B.6}$$

Since $\epsilon < 1$, we obtain that $2\epsilon - \epsilon^2$ is positive. Moreover, recall the result of Lemma 16 that all the eigenvalues of the Hessian inverse approximation $\hat{\mathbf{H}}_t^{-1}$ are lower and upper bounded by λ and Λ, respectively. These two observations imply that we can replace the term $\mathbf{g}_t^T \hat{\mathbf{H}}_t^{-1} \mathbf{g}_t$ by its lower bound $\lambda \|\mathbf{g}_t\|^2$. Moreover, existence of upper bound Λ for the eigenvalues of Hessian inverse approximation $\hat{\mathbf{H}}_t^{-1}$ implies that the term $\|\hat{\mathbf{H}}_t^{-1} \mathbf{g}_t\|^3$ is upper bounded by $\Lambda^3 \|\mathbf{g}_t\|^3$. Substituting these bounds for the second and third terms of (B.6) and subtracting $F(\mathbf{y}^*)$ from both sides of inequality (B.6) leads to

$$F(\mathbf{y}_{t+1}) - F(\mathbf{y}^*) \leq F(\mathbf{y}_t) - F(\mathbf{y}^*) - \frac{\left(2\epsilon - \epsilon^2\right) \lambda}{2} \|\mathbf{g}_t\|^2 + \frac{\alpha L \epsilon^3 \Lambda^3}{6} \|\mathbf{g}_t\|^3. \tag{B.7}$$

Since F is strongly convex with constant αm we can write [see Eq. (9.9) in [20]],

$$F(\mathbf{y}^*) \geq F(\mathbf{y}_t) - \frac{1}{2\alpha m} \|\nabla F(\mathbf{y}_t)\|^2. \tag{B.8}$$

Rearrange terms in (B.8) to obtain $2\alpha m(F(\mathbf{y}_t) - F(\mathbf{y}^*))$ as a lower bound for $\|\nabla F(\mathbf{y}_t)\|^2 = \|\mathbf{g}_t\|^2$. Now substitute the lower bound $2\alpha m(F(\mathbf{y}_t) - F(\mathbf{y}^*))$ for squared norm of gradient $\|\mathbf{g}_t\|^2$ in the second summand of (B.7) to obtain

$$F(\mathbf{y}_{t+1}) - F(\mathbf{y}^*) \leq \left[1 - \left(2\epsilon - \epsilon^2\right) \alpha m \lambda\right] \left(F(\mathbf{y}_t) - F(\mathbf{y}^*)\right) + \frac{\alpha L \epsilon^3 \Lambda^3}{6} \|\mathbf{g}_t\|^3. \tag{B.9}$$

Since the eigenvalues of the Hessian are upper bounded by $2(1-\delta)+\alpha M$, for any vectors $\hat{\mathbf{y}}$ and $\mathbf{y}$ in $\mathbb{R}^{np}$ we can write

$$F(\mathbf{y}) \leq F(\hat{\mathbf{y}}) + \nabla F(\hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) + \frac{2(1-\delta) + \alpha M}{2} \|\mathbf{y} - \hat{\mathbf{y}}\|^2. \tag{B.10}$$

According to the definition of λ in (5.52), we can substitute $2(1-\delta) + \alpha M$ by $1/\lambda$. Implementing this substitution and minimizing both sides of the equality with respect to $\mathbf{y}$ yields

$$F(\mathbf{y}^*) \leq \; F(\hat{\mathbf{y}}) - \lambda \|\nabla F(\hat{\mathbf{y}})\|^2. \tag{B.11}$$

Setting $\hat{\mathbf{y}} = \mathbf{y}_t$, replacing $\nabla F(\mathbf{y}_t)$ by $\mathbf{g}_t$, and taking the square root of both sides of the resulting inequality yields

$$\|\mathbf{g}_t\| \leq \left[\lambda^{-1} \left[F(\mathbf{y}_t) - F(\mathbf{y}^*)\right]\right]^{1/2}. \tag{B.12}$$

Replace the upper bound in (B.12) for the norm of the gradient $\|\mathbf{g}_t\|$ in the last term of (B.9) to obtain (B.2). ■

Proof of Theorem 7: To simplify upcoming derivations define the sequence β_t as

$$\beta_t := (2-\epsilon)\epsilon\alpha m\lambda - \frac{\epsilon^3 \alpha L \Lambda^3 \left[F(\mathbf{y}_t) - F(\mathbf{y}^*)\right]^{\frac{1}{2}}}{6\lambda^{\frac{3}{2}}}. \tag{B.13}$$

Recall the result of Lemma 29. Factorizing $F(\mathbf{y}_t) - F(\mathbf{y}^*)$ from the terms of the right hand side of (B.2) in association with the definition of β_t in (B.13) implies that we can simplify (B.2) as

$$F(\mathbf{y}_{t+1}) - F(\mathbf{y}^*) \leq (1-\beta_t)(F(\mathbf{y}_t) - F(\mathbf{y}^*)). \tag{B.14}$$

It remains to show that for all time steps t, the constants β_t satisfy $0 < \beta_t < 1$. We first show that $\beta_t < 1$ for all $t \geq 0$. Based on (B.13) we can write

$$\beta_t \leq (2-\epsilon)\epsilon\alpha m\lambda. \tag{B.15}$$

Considering $(\epsilon-1)^2 \geq 0$ we have $\epsilon(2-\epsilon) \leq 1$. Further, by inequalities $m < M$ and $1-\delta > 0$, we obtain $\alpha m < \alpha M + (1-\delta)$. Thus, $\alpha m/(\alpha M + 2(1-\delta)) < 1$ which is equivalent to $\alpha m\lambda < 1$. It follows from these inequalities that

$$(2-\epsilon)\epsilon\alpha m\lambda < 1. \tag{B.16}$$

That $\beta_t < 1$ follows by combining (B.15) with (B.16).

To prove that $0 < \beta_t$ for all $t \geq 0$ we prove that this is true for $t = 0$ and then prove that the β_t sequence is increasing. According to (5.59), we can write

$$\epsilon \leq \left[\frac{3m\lambda^{\frac{5}{2}}}{L\Lambda^3 (F(\mathbf{y}_0) - F(\mathbf{y}^*))^{\frac{1}{2}}}\right]^{\frac{1}{2}}, \tag{B.17}$$

By computing the squares of both sides of (B.17), multiplying the right hand side of the resulting inequality by 2 to make the inequality strict, and factorizing $\alpha m\lambda$ we obtain

$$\epsilon^2 < \frac{6\lambda^{\frac{3}{2}}}{\alpha L \Lambda^3 [F(\mathbf{y}_0) - F(\mathbf{y}^*)]^{\frac{1}{2}}} \times \alpha m\lambda. \tag{B.18}$$

If we now divide both sides of the inequality in (B.18) by the first multiplicand in the right hand side of (B.18) we obtain

$$\frac{\epsilon^2 \alpha L \Lambda^3 [F(\mathbf{y}_0) - F(\mathbf{y}^*)]^{\frac{1}{2}}}{6\lambda^{\frac{3}{2}}} < \alpha m\lambda. \tag{B.19}$$

Observe that based on the hypothesis in (5.59) the step size ϵ is smaller than 1 and it is then trivially true that $2-\epsilon \geq 1$. This observation shows that if we multiply the right hand side of (B.19) by $2(1-\epsilon/2)$ the inequality still holds,

$$\frac{\epsilon^2 \alpha L \Lambda^3 (F(\mathbf{y}_0) - F(\mathbf{y}^*))^{\frac{1}{2}}}{6\lambda^{\frac{3}{2}}} < \alpha m (2-\epsilon)\lambda. \tag{B.20}$$

Multiply both sides of (B.20) by ϵ and rearrange terms to obtain

$$\alpha m \epsilon (2-\epsilon)\lambda - \frac{\epsilon^3 \alpha L \Lambda^3 [F(\mathbf{y}_0) - F(\mathbf{y}^*)]^{\frac{1}{2}}}{6\lambda^{\frac{3}{2}}} > 0. \tag{B.21}$$

Based on (B.13), the result in (B.21) yields $\beta_0 > 0$. Observing that β_0 is positive, to show that for all t the sequence of β_t is positive it is sufficient to prove that the sequence β_t is increasing. We use strong induction to prove $\beta_t < \beta_{t+1}$ for all $t \geq 0$. By setting $t = 0$ in (B.14) we obtain

$$F(\mathbf{y}_1) - F(\mathbf{y}^*) \leq (1-\beta_0)(F(\mathbf{y}_0) - F(\mathbf{y}^*)). \tag{B.22}$$

Considering the result in (B.22) and the fact that $0 < \beta_0 < 1$, we obtain that the objective function error at time $t = 1$ is strictly smaller than the error at time $t = 0$, i.e.

$$F(\mathbf{y}_1) - F(\mathbf{y}^*) < F(\mathbf{y}_0) - F(\mathbf{y}^*). \tag{B.23}$$

According to (B.13), a smaller objective function error $F(\mathbf{y}_t) - F(\mathbf{y}^*)$ leads to a larger coefficient β_t. This observation combined with the result in (B.23) leads to

$$\beta_0 < \beta_1. \tag{B.24}$$

To complete the strong induction argument assume now that $\beta_0 < \beta_1 < \cdots < \beta_{t-1} < \beta_t$ and proceed to prove that if this is true we must have $\beta_t < \beta_{t+1}$. Begin by observing that since $0 < \beta_0$ the induction hypothesis implies that for all $u \in \{0, \ldots, t\}$ the constant β_u is also positive, i.e., $0 < \beta_u$. Further recall that for all t the sequence β_t is also smaller than 1 as already proved. Combining these two observations we have $0 < \beta_u < 1$ for all $u \in \{0, \ldots, t\}$. Consider now the inequality in (B.14) and utilize the fact that $0 < \beta_u < 1$ for all $u \in \{0, \ldots, t\}$ to conclude that

$$F(\mathbf{y}_{u+1}) - F(\mathbf{y}^*) < F(\mathbf{y}_u) - F(\mathbf{y}^*), \tag{B.25}$$

for all $u \in \{0, \ldots, t\}$. Setting $u = t$ in (B.25) we conclude that $F(\mathbf{y}_{t+1}) - F(\mathbf{y}^*) < F(\mathbf{y}_t) - F(\mathbf{y}^*)$. By further repeating the argument leading from (B.24) to (B.23) we can conclude

that

$$\beta_t < \beta_{t+1}. \tag{B.26}$$

The strong induction proof is complete and we can claim that

$$0 < \beta_0 < \beta_1 < \cdots < \beta_t < 1, \tag{B.27}$$

for all times t. The results in (B.14) and (B.27) imply $\lim_{t\to\infty} F(\mathbf{y}_t) - F(\mathbf{y}^*) = 0$. To conclude that the rate is at least linear simply observe that if the sequence β_t is increasing as per (B.27), the sequence $1 - \beta_t$ is decreasing and satisfies

$$0 < 1 - \beta_t < 1 - \beta_0 < 1, \tag{B.28}$$

for all time steps t. Applying the inequality in (B.14) recursively and considering the inequality in (B.28) yields

$$F(\mathbf{y}_t) - F(\mathbf{y}^*) \leq (1 - \beta_0)^t (F(\mathbf{y}_0) - F(\mathbf{y}^*)). \tag{B.29}$$

Considering $\zeta = \beta_0$, the claim in (5.60) follows.

Appendix C

Appendix

C.1 Proof of Proposition 9

The steps of the proof for Proposition 13 are adopted from the analysis in [59]. We start the proof by providing an upper bound for the difference between the loss functions L_n and L_m. The upper bound is studied in the following lemma which uses the condition in (**??**).

Lemma 30 *Consider L_n and L_m as the empirical losses of the sets $\mathcal{S}_n$ and $\mathcal{S}_m$, respectively, where they are chosen such that $\mathcal{S}_m \subset \mathcal{S}_n$. If we define n and m as the number of samples in the training sets $\mathcal{S}_n$ and $\mathcal{S}_m$, respectively, then the expected absolute value of the difference between the empirical losses is bounded above by*

$$\mathbb{E}\left[\,|L_n(\mathbf{w}) - L_m(\mathbf{w})|\,\right] \leq \frac{n-m}{n}\left(V_{n-m} + V_m\right), \tag{C.1}$$

for any $\mathbf{w}$.

Proof: First we characterize the difference between the difference of the loss functions associated with the sets $\mathcal{S}_m$ and $\mathcal{S}_n$. To do so, consider the difference

$$L_n(\mathbf{w}) - L_m(\mathbf{w}) = \frac{1}{n}\sum_{i\in\mathcal{S}_n} f_i(\mathbf{w}) - \frac{1}{m}\sum_{i\in\mathcal{S}_m} f_i(\mathbf{w}). \tag{C.2}$$

Notice that the set $\mathcal{S}_m$ is a subset of the set $\mathcal{S}_n$ and we can write $\mathcal{S}_n = \mathcal{S}_m \cup \S_{n-m}$. Thus, we can rewrite the right hand side of (C.2) as

$$\begin{aligned} L_n(\mathbf{w}) - L_m(\mathbf{w}) &= \frac{1}{n}\left[\sum_{i\in\mathcal{S}_m} f_i(\mathbf{w}) + \sum_{i\in\S_{n-m}} f_i(\mathbf{w})\right] - \frac{1}{m}\sum_{i\in\mathcal{S}_m} f_i(\mathbf{w}) \\ &= \frac{1}{n}\sum_{i\in\S_{n-m}} f_i(\mathbf{w}) - \frac{n-m}{mn}\sum_{i\in\mathcal{S}_m} f_i(\mathbf{w}). \end{aligned} \tag{C.3}$$

Factoring $(n-m)/n$ from the terms in the right hand side of (C.3) follows

$$L_n(\mathbf{w}) - L_m(\mathbf{w}) = \frac{n-m}{n}\left[\frac{1}{n-m}\sum_{i\in\S_{n-m}} f_i(\mathbf{w}) - \frac{1}{m}\sum_{i\in\mathcal{S}_m} f_i(\mathbf{w})\right]. \tag{C.4}$$

Now add and subtract the statistical loss $L(\mathbf{w})$ and compute the expected value to obtain

$$\begin{aligned} \mathbb{E}[|L_n(\mathbf{w}) - L_m(\mathbf{w})|] &= \frac{n-m}{n}\mathbb{E}\left[\left|\frac{1}{n-m}\sum_{i\in\S_{n-m}} f_i(\mathbf{w}) - L(\mathbf{w}) + L(\mathbf{w}) - \frac{1}{m}\sum_{i\in\mathcal{S}_m} f_i(\mathbf{w})\right|\right] \\ &\leq \frac{n-m}{n}\left(V_{n-m} + V_m\right), \end{aligned} \tag{C.5}$$

where the last inequality follows by using the triangle inequality and the upper bound in (**??**). ■

The result in Lemma 30 shows that the upper bound for the difference between the loss functions associated with the sets $\mathcal{S}_m$ and $\mathcal{S}_n$ where $\mathcal{S}_m \subset \mathcal{S}_n$ is proportional to the difference between the size of these two sets $n - m$.

In the following lemma, we characterize an upper bound for the norm of the optimal argument $\mathbf{w}_n^*$ of the empirical risk $R_n(\mathbf{w})$ in terms of the norm of statistical average loss $L(\mathbf{w})$ optimal argument $\mathbf{w}^*$.

Lemma 31 *Consider L_n as the empirical loss of the set $\mathcal{S}_n$ and L as the statistical average loss. Moreover, recall $\mathbf{w}^*$ as the optimal argument of the statistical average loss L, i.e., $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} L(\mathbf{w})$. If Assumption 18 holds, then the norm of the optimal argument $\mathbf{w}_n^*$ of the regularized empirical risk $R_n(\mathbf{w}) := L_n(\mathbf{w}) + cV_n\|\mathbf{w}\|^2$ is bounded above by*

$$\mathbb{E}[\|\mathbf{w}_n^*\|^2] \leq \frac{4}{c} + \|\mathbf{w}^*\|^2 \tag{C.6}$$

Proof: The optimality condition of $\mathbf{w}_n^*$ for the the regularized empirical risk $R_n(\mathbf{w}) = L_n(\mathbf{w}) + (cV_n)/2\|\mathbf{w}\|^2$ implies that

$$L_n(\mathbf{w}_n^*) + \frac{cV_n}{2}\|\mathbf{w}_n^*\|^2 \leq L_n(\mathbf{w}^*) + \frac{cV_n}{2}\|\mathbf{w}^*\|^2. \tag{C.7}$$

By regrouping the terms and computing the expectation we can show that $\mathbb{E}[\|\mathbf{w}_n^*\|^2]$ is bonded above by

$$\mathbb{E}[\|\mathbf{w}_n^*\|^2] \leq \frac{2}{cV_n}\mathbb{E}[(L_n(\mathbf{w}^*) - L_n(\mathbf{w}_n^*))] + \|\mathbf{w}^*\|^2. \tag{C.8}$$

We proceed to bound the difference $L_n(\mathbf{w}^*) - L_n(\mathbf{w}_n^*)$. By adding and subtracting the terms $L(\mathbf{w}^*)$ and $L(\mathbf{w}_n^*)$ we obtain that

$$L_n(\mathbf{w}^*) - L_n(\mathbf{w}_n^*) = \big[L_n(\mathbf{w}^*) - L(\mathbf{w}^*)\big] + \big[L(\mathbf{w}^*) - L(\mathbf{w}_n^*)\big] + \big[L(\mathbf{w}_n^*) - L_n(\mathbf{w}_n^*)\big]. \tag{C.9}$$

Notice that the second bracket in (C.9) is non-positive since $L(\mathbf{w}^*) \leq L(\mathbf{w}_n^*)$. Therefore, it is bounded by 0. According to (**??**), the first and third brackets in (C.9) are bounded above by V_n in expectation. Replacing these upper bounds by the brackets in (C.9) yields

$$\mathbb{E}[L_n(\mathbf{w}^*) - L_n(\mathbf{w}_n^*)] \leq 2V_n. \tag{C.10}$$

Substituting the upper bound in (C.10) into (C.8) implies the claim in (C.6). ■

Note that the difference $R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)$ can be written as

$$\begin{aligned} R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*) = R_n(\mathbf{w}_m) - R_m(\mathbf{w}_m) + R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \\ + R_m(\mathbf{w}_m^*) - R_m(\mathbf{w}_n^*) + R_m(\mathbf{w}_n^*) - R_n(\mathbf{w}_n^*). \end{aligned} \tag{C.11}$$

We proceed to bound the differences in (C.11). To do so, note that the difference $R_n(\mathbf{w}_m) - R_m(\mathbf{w}_m)$ can be simplified as

$$\begin{aligned} R_n(\mathbf{w}_m) - R_m(\mathbf{w}_m) &= L_n(\mathbf{w}_m) - L_m(\mathbf{w}_m) + \frac{c(V_n - V_m)}{2}\|\mathbf{w}_m\|^2 \\ &\leq L_n(\mathbf{w}) - L_m(\mathbf{w}), \end{aligned} \tag{C.12}$$

where the inequality follows from the fact that $V_n < V_m$ and $V_n - V_m$ is negative. It follows from the result in Lemma 30 that the right hand side of (C.12) is bounded by $(n-m)/n\,(V_{n-m} + V_m)$. Therefore,

$$\mathbb{E}\left[|R_n(\mathbf{w}_m) - R_m(\mathbf{w}_m)|\right] \leq \frac{n-m}{n}\left(V_{n-m} + V_m\right). \tag{C.13}$$

Since $\mathbf{w}_m$ is an δ_m sub-optimal solution for R_m we know that

$$\mathbb{E}[R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*)] \leq \delta_m. \tag{C.14}$$

Based on the definition of $\mathbf{w}_m^*$ which is the optimal solution of the risk R_m, the third difference in (C.11) which is $R_m(\mathbf{w}_m^*) - R_m(\mathbf{w}_n^*)$ is always negative. I.e.,

$$R_m(\mathbf{w}_m^*) - R_m(\mathbf{w}_n^*) \leq 0. \tag{C.15}$$

Moreover, we can use the triangle inequality to bound the difference $R_m(\mathbf{w}_n^*) - R_n(\mathbf{w}_n^*)$ in (C.11) as

$$\begin{aligned} \mathbb{E}[R_m(\mathbf{w}_n^*) - R_n(\mathbf{w}_n^*)] &= \mathbb{E}[L_m(\mathbf{w}_n^*) - L_n(\mathbf{w}_n^*)] + \frac{c(V_m - V_n)}{2}\mathbb{E}[\|\mathbf{w}_n^*\|^2] \\ &\leq \frac{n-m}{n}\left(V_{n-m} + V_m\right) + \frac{c(V_m - V_n)}{2}\mathbb{E}[\|\mathbf{w}_n^*\|^2]. \end{aligned} \tag{C.16}$$

Replacing the differences in (C.11) by the upper bounds in (C.13)-(C.16) leads to

$$\mathbb{E}[R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*)] \leq \delta_m + \frac{2(n-m)}{n}\left(V_{n-m} + V_m\right) + \frac{c(V_m - V_n)}{2}\mathbb{E}[\|\mathbf{w}_n^*\|^2] \tag{C.17}$$

Substitute $\mathbb{E}[\|\mathbf{w}_n^*\|^2]$ in (C.17) by the upper bound in (C.6) to obtain the result in (9.29).

Bibliography

[1] Z. Allen Zhu, "Katyusha: the first direct acceleration of stochastic gradient methods," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, 2017, pp. 1200–1205.

[2] T. M. Apostol, *Calculus, volume I.* John Wiley & Sons, 2007, vol. 1.

[3] D. Bajovic, D. Jakovetic, N. Krejic, and N. K. Jerinkic, "Newton-like method with diagonal correction for distributed optimization," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 1171–1203, 2017.

[4] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.

[5] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[6] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches.* Cambridge University Press, 2011.

[7] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods.* Academic press, 2014.

[8] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods.* Prentice-Hall, Inc., 1989.

[9] A. Bijral, A. D. Sarwate, and N. Srebro, "Data dependent convergence for consensus stochastic optimization," *IEEE Transactions on Automatic Control*, 2017.

[10] J. R. Birge, X. Chen, L. Qi, and Z. Wei, "A stochastic newton method for stochastic quadratic programs with resource," *Technical report*, University of Michigan, Ann Arbor, MI 1995.

[11] D. Blatt, A. O. Hero, and H. Gauchman, "A convergent incremental gradient method with a constant step size," *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 29–51, 2007.

[12] A. Bordes, L. Bottou, and P. Gallinari, "SGD-QN: Careful quasi-Newton stochastic gradient descent," *The Journal of Machine Learning Research*, vol. 10, pp. 1737–1754, 2009.

[13] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[14] L. Bottou and Y. L. Cun, "On-line learning for very large datasets," in *Applied Stochastic Models in Business and Industry*, vol. 21. pp. 137-151, 2005.

[15] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[16] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems 20, Vancouver, British Columbia, Canada, December 3-6, 2007*, 2007, pp. 161–168.

[17] O. Bousquet, "Concentration inequalities and empirical processes theory applied to the analysis of learning algorithms," *PhD thesis, Ecole Polytechnique*, 2002.

[18] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM review*, vol. 46, no. 4, pp. 667–689, 2004.

[19] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[21] C. G. Broyden, J. E. D. Jr., Wang, and J. J. More, "On the local and superlinear convergence of quasi-Newton methods," *IMA J. Appl. Math*, vol. 12, no. 3, pp. 223–245, June 1973.

[22] F. Bullo, J. Cortes, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

[23] R. H. Byrd, S. 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.

[24] R. H. Byrd, J. Nocedal, and Y.-X. Yuan, "Global convergence of a class of quasi-Newton methods on convex problems," *SIAM Journal on Numerical Analysis*, vol. 24, no. 5, pp. 1171–1190, 1987.

[25] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 427–438, 2013.

[26] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 32–43, 2014.

[27] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus admm," *Signal Processing, IEEE Transactions on*, vol. 63, no. 2, pp. 482–497, 2015.

[28] N. Chatzipanagiotis, D. Dentcheva, and M. M. Zavlanos, "An augmented Lagrangian method for distributed optimization," *Mathematical Programming*, vol. 152, no. 1-2, pp. 405–434, 08 2015.

[29] H. Daneshmand, A. Lucchi, and T. Hofmann, "Starting small - learning with adaptive sample sizes," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA*, 2016, pp. 1463–1471.

[30] A. Defazio, "A simple practical accelerated method for finite sums," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 676–684.

[31] A. Defazio, F. R. 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 27, Montreal, Quebec, Canada*, 2014, pp. 1646–1654.

[32] 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.

[33] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: convergence analysis and network scaling," *Automatic control, IEEE Transactions on*, vol. 57, no. 3, pp. 592–606, 2012.

[34] M. A. Erdogdu and A. Montanari, "Convergence rates of sub-sampled Newton methods," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, Montreal, Quebec, Canada*, 2015, pp. 3052–3060.

[35] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[36] R. Frostig, R. Ge, S. M. Kakade, and A. Sidford, "Competing with the empirical risk minimizer in a single pass," in *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, 2015, pp. 728–763.

[37] R. M. Gower, D. Goldfarb, and P. Richtárik, "Stochastic block BFGS: squeezing more curvature out of data," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA*, 2016, pp. 1869–1878.

[38] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo, "On the convergence rate of incremental aggregated gradient algorithms," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 1035–1048, 2017.

[39] ——, "A globally convergent incremental Newton method," *Mathematical Programming*, vol. 151, no. 1, pp. 283–313, 2015.

[40] M. R. Hestenes, "Multiplier and gradient methods," *Journal of optimization theory and applications*, vol. 4, no. 5, pp. 303–320, 1969.

[41] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Explicit convergence rate of a distributed alternating direction method of multipliers," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 892–904, 2016.

[42] D. Jakovetic, J. M. Moura, and J. Xavier, "Distributed nesterov-like gradient algorithms," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 5459–5464.

[43] D. Jakovetic, J. Xavier, and J. M. Moura, "Cooperative convex optimization in networked systems: Augmented Lagrangian algorithms with directed gossip communication," *IEEE Transactions on Signal Processing*, vol. 59, no. 8, pp. 3889–3902, 2011.

[44] ——, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[45] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems 26, Lake Tahoe, Nevada, United States*, 2013, pp. 315–323.

[46] U. A. Khan, S. Kar, and J. M. Moura, "Diland: An algorithm for distributed sensor localization with noisy distance measurements," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1940–1947, 2010.

[47] J. Konecnỳ and P. Richtárik, "Semi-stochastic gradient descent methods," *arXiv preprint arXiv:1312.1666*, vol. 2, no. 2.1, p. 3, 2013.

[48] G. Lan, S. Lee, and Y. Zhou, "Communication-efficient algorithms for decentralized and stochastic optimization," *arXiv preprint arXiv:1701.03961*, 2017.

[49] N. Le Roux, M. W. 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 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, 2012, pp. 2672–2680.

[50] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

[51] D.-H. Li and M. Fukushima, "A modified BFGS method and its global convergence in nonconvex minimization," *Journal of Computational and Applied Mathematics*, vol. 129, no. 1, pp. 15–35, 2001.

[52] H. Lin, J. Mairal, and Z. Harchaoui, "A universal catalyst for first-order optimization," in *Advances in Neural Information Processing Systems*, 2015, pp. 3366–3374.

[53] Q. Ling and A. Ribeiro, "Decentralized linearized alternating direction method of multipliers," *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 5447–5451, 2014.

[54] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized linearized alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015.

[55] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[56] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, July 2008.

[57] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.

[58] A. Lucchi, B. McWilliams, and T. Hofmann, "A variance reduced stochastic Newton method," *arXiv preprint arXiv:1503.08316*, 2015.

[59] A. Mokhtari, H. Daneshmand, A. Lucchi, T. Hofmann, and A. Ribeiro, "Adaptive Newton method for empirical risk minimization to statistical accuracy," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 4062–4070.

[60] A. Mokhtari, M. Eisen, and A. Ribeiro, "An incremental quasi-Newton method with a local superlinear convergence rate," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4039–4043.

[61] ——, "IQN: An incremental quasi-Newton method with local superlinear convergence rate," *arXiv preprint arXiv:1702.00709*, 2017.

[62] A. Mokhtari, M. Gürbüzbalaban, and A. Ribeiro, "Surpassing gradient descent provably: A cyclic incremental method with linear convergence rate," *arXiv preprint arXiv:1611.00347*, 2016.

[63] A. Mokhtari, Q. Ling, and A. Ribeiro, "An approximate Newton method for distributed optimization," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2959–2963.

[64] ——, "Network Newton-Part I: Algorithm and convergence," *arXiv preprint arXiv:1504.06017*, 2015.

[65] ——, "Network Newton-Part II: Convergence rate and implementation," *arXiv preprint arXiv:1504.06020*, 2015.

[66] ——, "Network newton distributed optimization methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 146–161, 2017.

[67] A. Mokhtari and A. Ribeiro, "A dual stochastic DFP algorithm for optimal resource allocation in wireless systems," in *Signal Processing Advances in Wireless Communications (SPAWC), 2013 IEEE 14th Workshop on*. IEEE, 2013, pp. 21–25.

[68] ——, "Regularized stochastic BFGS algorithm," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 1109–1112.

[69] ——, "A quasi-Newton method for large scale support vector machines," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 8302–8306.

[70] ——, "RES: Regularized stochastic BFGS algorithm," *IEEE Transactions on Signal Processing*, vol. 62, no. 23, pp. 6089–6104, 2014.

[71] ——, "Decentralized double stochastic averaging gradient," in *Signals, Systems and Computers, 2015 49th Asilomar Conference on*. IEEE, 2015, pp. 406–410.

[72] ——, "Global convergence of online limited memory BFGS," *Journal of Machine Learning Research*, vol. 16, pp. 3151–3181, 2015.

[73] ——, "DSA: decentralized double stochastic averaging gradient algorithm," *Journal of Machine Learning Research*, vol. 17, no. 61, pp. 1–35, 2016.

[74] A. Mokhtari, W. Shi, and Q. Ling, "ESOM: Exact second-order method for consensus optimization," in *Signals, Systems and Computers, 2016 50th Asilomar Conference on*. IEEE, 2016, pp. 783–787.

[75] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "Decentralized quadratically approximated alternating direction method of multipliers," in *Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on*. IEEE, 2015, pp. 795–799.

[76] ——, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

[77] ——, "DQM: Decentralized quadratically approximated alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5158–5173, 2016.

[78] P. Moritz, R. Nishihara, and M. I. Jordan, "A linearly-convergent stochastic L-BFGS algorithm," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, 2016, pp. 249–258.

[79] J. M. Mulvey and A. Ruszczyn, "A diagonal quadratic approximation method for large scale linear programs," *Operations Research Letters*, vol. 12, no. 4, pp. 205–215, 1992.

[80] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[81] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[82] A. Nemirovski, A. Juditsky, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[83] Y. Nesterov, "Introductory lectures on convex programming volume i: Basic course," 1998.

[84] ——, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013, vol. 87.

[85] Y. Nesterov *et al.*, "Gradient methods for minimizing composite objective function," UCL, Tech. Rep., 2007.

[86] A. Y. Ng, "Feature selection, L1 vs. L2 regularization, and rotational invariance," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 78–.

[87] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed. New York, NY: Springer-Verlag, 1999.

[88] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[89] M. J. Powell, "Some global convergence properties of a variable metric algorithm for minimization without exact line searches," *Nonlinear programming*, vol. 9, no. 1, pp. 53–72, 1976.

[90] G. Qu and N. Li, "Accelerated distributed nesterov gradient descent," *arXiv preprint arXiv:1705.07176*, 2017.

[91] Z. Qu, P. Richtárik, M. Takác, and O. Fercoq, "SDNA: stochastic dual Newton ascent for empirical risk minimization," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 1823–1832.

[92] M. Rabbat, "Multi-agent mirror descent for decentralized stochastic optimization," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE 6th International Workshop on*. IEEE, 2015, pp. 517–520.

[93] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM, 2004, pp. 20–27.

[94] M. G. Rabbat and R. D. Nowak, "Decentralized source localization and tracking [wireless sensor networks]," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 3. IEEE, 2004, pp. iii–921.

[95] M. G. Rabbat, R. D. Nowak, J. Bucklew *et al.*, "Generalized consensus computation in networked systems with erasure links," in *Signal Processing Advances in Wireless Communications, 2005 IEEE 6th Workshop on*. IEEE, 2005, pp. 1088–1092.

[96] A. Ribeiro, "Ergodic stochastic optimization algorithms for wireless communication and networking," *IEEE Transactions on Signal Processing*, vol. 58, no. 12, pp. 6369–6386, 2010.

[97] ——, "Optimal resource allocation in wireless communication and networking," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–19, 2012.

[98] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[99] A. Rodomanov and D. Kropotov, "A superlinearly-convergent proximal newton-type method for the optimization of finite sums," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 2597–2605.

[100] A. Ruszczyński, "On convergence of an augmented Lagrangian decomposition method for sparse convex optimization," *Mathematics of Operations Research*, vol. 20, no. 3, pp. 634–656, 1995.

[101] A. Ruszczynski and W. Syski, "Stochastic approximation method with gradient averaging for unconstrained problems," *IEEE Transactions on Automatic Control*, vol. 28, no. 12, pp. 1097–1105, 1983.

[102] A. P. Ruszczyński, *Nonlinear optimization*. Princeton university press, 2006, vol. 13.

[103] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links–Part I: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.

[104] M. W. Schmidt, N. Le Roux, and F. R. Bach, "Minimizing finite sums with the stochastic average gradient," *Math. Program.*, vol. 162, no. 1-2, pp. 83–112, 2017.

[105] N. N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-Newton method for online convex optimization," in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS 2007, San Juan, Puerto Rico, March 21-24, 2007*, 2007, pp. 436–443.

[106] S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan, "Learnability, stability and uniform convergence," *The Journal of Machine Learning Research*, vol. 11, pp. 2635–2670, 2010.

[107] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for SVM," *Math. Program.*, vol. 127, no. 1, pp. 3–30, 2011.

[108] S. Shalev-Shwartz and N. Srebro, "SVM optimization: inverse dependence on training set size," in *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, 2008, pp. 928–935.

[109] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss," *The Journal of Machine Learning Research*, vol. 14, pp. 567–599, 2013.

[110] ——, "Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization," *Mathematical Programming*, vol. 155, no. 1-2, pp. 105–145, 2016.

[111] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: an exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[112] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization." *IEEE Trans. Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[113] B. Sirb and X. Ye, "Consensus optimization with delayed and stochastic gradients on decentralized networks," in *Big Data (Big Data), 2016 IEEE International Conference on.* IEEE, 2016, pp. 76–85.

[114] V. Solo and X. Kong, *Adaptive Signal Processing Algorithms: Stability and Performance.* Englewood Cliffs: NJ: Prentice-Hall, 1995.

[115] G. Stephanopoulos and A. W. Westerberg, "The use of hestenes' method of multipliers to resolve dual gaps in engineering system optimization," *Journal of Optimization Theory and Applications*, vol. 15, no. 3, pp. 285–309, 1975.

[116] G. Sun, "Kdd cup track 2 soso.com ads prediction challenge, 2012," Accessed August 1, 2012.

[117] R. Tappenden, P. Richtárik, and B. Büke, "Separable approximations and decomposition methods for the augmented Lagrangian," *Optimization Methods and Software*, no. ahead-of-print, pp. 1–26, 2014.

[118] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on.* IEEE, 2012, pp. 1543–1550.

[119] ——, "Push-sum distributed dual averaging for convex optimization," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on.* IEEE, 2012, pp. 5453–5458.

[120] V. Vapnik, *The nature of statistical learning theory*, 2nd ed. springer, 1999.

[121] ——, *The nature of statistical learning theory.* Springer Science & Business Media, 2013.

[122] N. Watanabe, Y. Nishimura, and M. Matsubara, "Decomposition in large system optimization using the method of multipliers," *Journal of Optimization Theory and Applications*, vol. 25, no. 2, pp. 181–193, 1978.

[123] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed Newton method for network utility maximization–I: Algorithm," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2162–2175, 2013.

[124] B. E. Woodworth and N. Srebro, "Tight complexity bounds for optimizing composite objectives," in *Advances in Neural Information Processing Systems*, 2016, pp. 3639–3647.

[125] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 2057–2075, 2014.

[126] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[127] M. Zargham, A. Ribeiro, and A. Jadbabaie, "A distributed line search for network optimization," in *American Control Conference (ACC), 2012*. IEEE, 2012, pp. 472–477.

[128] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, "Accelerated dual descent for network flow optimization," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 905–920, 2014.

[129] L. Zhang, M. Mahdavi, and R. Jin, "Linear convergence with condition number independent access of full gradients," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, 2013, pp. 980–988.

[130] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, 2004.