DAve-QN: A Distributed Averaged Quasi-Newton Method with Local Superlinear Convergence Rate

Saeed Soori
CS Department
University of Toronto
sasoori@cs.toronto.edu

Konstantin Mischenko
CS Department
KAUST University
konstantin.mishchenko@kaust.edu.sa

Aryan Mokhtari
ECE Department
University of Texas at Austin
mokhtari@austin.utexas.edu

Maryam Mehri Dehnavi
CS Department
University of Toronto
mmehride@cs.toronto.edu

Mert Gurbuzbalaban
MSIS Department
Rutgers University
mert.gurbuzbalaban@rutgers.edu

Abstract

In this paper, we consider distributed algorithms for solving the empirical risk minimization problem under the master/worker communication model. We develop a distributed asynchronous quasi-Newton algorithm that can achieve superlinear convergence. To our knowledge, this is the first distributed asynchronous algorithm with superlinear convergence guarantees. Our algorithm is communication-efficient in the sense that at every iteration the master node and workers communicate vectors of size $O(p)$, where p is the dimension of the decision variable. The proposed method is based on a distributed asynchronous averaging scheme of decision vectors and gradients in a way to effectively capture the local Hessian information of the objective function. Our convergence theory supports asynchronous computations subject to both bounded delays and unbounded delays with a bounded time-average. Unlike in the majority of asynchronous optimization literature, we do not require choosing smaller stepsize when delays are huge. We provide numerical experiments that match our theoretical results and showcase significant improvement comparing to state-of-the-art distributed algorithms.

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

1 Introduction

Many optimization problems in machine learning including *empirical risk minimization* are based on processing large amounts of data as an input. Due to the advances in sensing technologies and storage capabilities the size of the data we can collect and store increases at an exponential manner. As a consequence, a single machine (processor) is typically not capable of processing and storing all the samples of a dataset. To solve such "big data" problems, we typically rely on distributed architectures where the data is distributed over several machines that reside on a communication network (Bertsekas and Tsitsiklis (1989); Recht et al. (2011)). In such modern architectures, the cost of communication is typically orders of magnitude larger than the cost of floating point operation costs and the gap is increasing (Dongarra et al. (2014)). This requires development of distributed optimization algorithms that can find the right trade-off between the cost of local computations and that of communication.

In this paper, we focus on distributed algorithms for empirical risk minimization problems. The setting is as follows: Given n machines, each machine has access to m_i samples $\{\xi_{i,j}\}_{j=1}^{m_i}$ for $i = 1, 2, \ldots, n$. The samples $\xi_{i,j}$ are random variables supported on a set $\mathcal{P} \subset \mathbb{R}^d$. Each machine has a loss function that is averaged over the local dataset:

$$f_i(\mathbf{x}) = \frac{1}{m_i}\sum_{j=1}^{m_i} \phi(\mathbf{x}, \xi_{i,j}) + \frac{\lambda}{2}\|\mathbf{x}\|_2^2$$

where the function $\phi : \mathbb{R}^p \times \mathbb{R}^d \to \mathbb{R}$ is convex in $\mathbf{x}$ for each $\xi \in \mathbb{R}^d$ fixed and $\lambda \geq 0$ is a regularization parameter. The goal is to develop communication-efficient distributed algorithms to minimize the overall empirical loss defined by

$$\mathbf{x}^* := \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}). \qquad (1)$$

The communication model we consider is the *centralized communication* model, also known as the *master/worker* model (Xiao et al. (2019)). In this model, the master machine possesses a copy of the global decision variable $\mathbf{x}$ which is shared with the worker machines. Each worker performs local computations on its local data which is then communicated to the master node to update the decision variable. Communications can be synchronous or asynchronous, resulting in different types of optimization algorithms and convergence guarantees. The merit of synchronization is that it prevents workers from using obsolete information and, thereby, from submitting a low quality update of parameters to the master. However all the nodes have to wait for the slowest worker, which leads to unnecessary overheads. Asynchronous algorithms do not suffer from this issue, maximizing the efficiency of the workers while minimizing the system overheads. Asynchronous algorithms are particularly preferable over networks with heterogeneous machines with different memory capacities, work overloads, and processing capabilities.

There has been a number of distributed algorithms suggested in the literature to solve the empirical risk minimization problem (1) based on primal first-order methods (Vanli et al. (2016); Gürbüzbalaban et al. (2017); Cutkosky and Busa-Fekete (2018)), their accelerated or variance-reduced versions (Lee et al. (2017); Wai et al. (2018a,b); Leblond et al. (2017); Pedregosa et al. (2017)), lock-free parallel methods (Recht et al. (2011); Peng et al. (2016)), coordinate descent-based approaches (Xiao et al. (2019); Takáč et al. (2015); Yang (2013); Bianchi et al. (2015)), dual methods (Agarwal and Duchi (2011); Yang (2013)), primal-dual methods (Xiao et al. (2019); Smith et al. (2016); Ma et al. (2015); Bianchi et al. (2015); Chen et al. (2018)), distributed ADMM-like methods (Zhang and Kwok (2014)) as well as quasi-Newton approaches (Eisen et al. (2017); Lee et al. (2018)), inexact second-order methods (Shamir et al. (2014); Reddi et al. (2016); Zhang and Lin (2015); Wang et al. (2017); Dünner et al. (2018); Gürbüzbalaban et al. (2015)) and general-purpose frameworks for distributed computing environments (Smith et al. (2016); Ma et al. (2015)) both in the asynchronous and synchronous setting. The efficiency of these algorithms is typically measured by the *communication complexity* which is defined as the equivalent number of vectors in $\mathbb{R}^p$ sent or received across all the machines until the optimization algorithm converges to an ε-neighborhood of the optimum value. Lower bounds on the communication complexity have been derived by Arjevani and Shamir (2015) as well as some linearly convergent algorithms achieving these lower bounds (Zhang and Lin (2015); Lee et al. (2017)). However, in an analogy to the lower bounds obtained by Nemirovskii et al. (1983) for first-order centralized algorithms, the lower bounds for the communication complexity are only effective if the dimension p of the problem is allowed to be larger than the number of iterations. This assumption is perhaps reasonable for very large scale problems where p can be billions, however it is clearly conservative for moderate to large-scale problems where p is not as large.

Contributions: Most existing state-of-the-art communication-efficient algorithms for strongly convex problems share vectors of size $O(p)$ at every iteration while having linear convergence guarantees. In this work, we propose the first communication-efficient asynchronous optimization algorithm that can achieve local superlinear convergence for solving the empirical risk minimization problem under the master/worker communication model. Our algorithm is communication-efficient in the sense that it also shares vectors of size $O(p)$. Our theory supports asynchronous computations subject to both bounded delays and unbounded delays with a bounded time-average. We provide numerical experiments that illustrate our theory and practical performance. The proposed method is based on a distributed asynchronous averaging scheme of decision vectors and gradients in a way to effectively capture the local Hessian information. Our proposed algorithm, Distributed Averaged Quasi-Newton (DAve-QN) is inspired by the Incremental Quasi-Newton (IQN) method proposed by Mokhtari et al. (2018a) which is a deterministic incremental algorithm based on the BFGS method. In contrast to the IQN method which is designed for centralized computation, our proposed scheme can be implemented in asynchronous master/worker distributed settings; allowing better scalability properties with parallelization, while being robust to delays of the workers.

Related work. Although the setup that we consider in this paper is an asynchronous master/worker distributed setting, it also relates to incremental aggregated algorithms (Roux et al. (2012); Defazio et al. (2014a,b); Mairal (2015); Gürbüzbalaban et al. (2017); Mokhtari et al. (2018b); Vanli et al. (2018)), as at each iteration the information corresponding to one of the machines, i.e., functions, is evaluated while the variable is updated by aggregating the most recent information of all the machines. In fact, our method is inspired by an incremental quasi-Newton method proposed by Mokhtari et al. (2018a) and a delay-tolerant method by Mishchenko et al. (2018). However, in the IQN method, the update at iteration t is a function of the last n iterates $\{x^{t-1}, \dots, x^{t-n}\}$, while in our asynchronous distributed scheme the updates are performed on delayed iterates $\{x^{t-d_1^t-1}, \ldots, x^{t-d_n^t-n}\}$. This major difference between the updates of these two algorithms requires a challenging different analysis. Further, our algorithm can be considered as an asynchronous distributed variant of traditional quasi-Newton methods that have been heavily studied in the numerical optimization community (Goldfarb (1970); Broyden et al. (1973a); Dennis and Moré (1974); Powell (1976)). Also, there have been some works on decentralized variants of quasi-Newton methods for consensus optimization where communications are performed over a fixed arbitrary graph where a master node is impractical or does not exist, this setup is also known as the *multi-agent setting* (Nedic and Ozdaglar (2009); Mansoori and Wei (2017)). The work proposed by Eisen et al. (2017) introduces a linearly convergent decentralized quasi-Newton method for decentralized settings. Mansoori and Wei (2017) propose an asynchronous Newton-based approach that solves a penalized version of the problem whose solution lies in a $O(\alpha)$ neighborhood of the optimal solution where α is a penalty parameter. Their algorithm enjoys local superlinear convergence guarantees to this neighborhood. We emphasize that our setup is different in the sense that we have a star network topology obeying the master/slave hierarchy. Furthermore, our asymptotic convergence results are stronger than those available in the multi-agent setting as we establish a superlinear convergence rate for the proposed method to the global minimum x^* of the problem (1). There has also been recent progress in solving distributed non-convex problems with second-order methods. Among these the most relevant to our paper are Şimşekli et al. (2018) which proposes an asynchronous-parallel stochastic L-BFGS method and the DINGO method (Crane and Roosta (2019)). DINGO is a Newton-type method that optimizes the gradient's norm as a surrogate function with linear convergence guarantees to a local minimum for non-convex objectives that satisfy an invexity property.

Outline. In Section 2.1, we review the update of the BFGS algorithm that we build on our distributed quasi-Newton algorithm. We formally present our proposed DAve-QN algorithm in Section 2.2. We then provide our theoretical convergence results for the proposed DAve-QN method in Section 3. Numerical results are presented in Section 4. Finally, we give a summary of our results and discuss future work in Section 5.

2 Algorithm

2.1 Preliminaries: The BFGS algorithm

The update of the BFGS algorithm for minimizing a convex smooth function $f : \mathbb{R}^p \to \mathbb{R}$ is given by

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta_t (\mathbf{B}^{t+1})^{-1} \nabla f(\mathbf{x}^t), \tag{2}$$

where $\mathbf{B}^{t+1}$ is an estimate of the Hessian $\nabla^2 f(\mathbf{x}^t)$ at time t and η_t is the stepsize (see e.g. Nocedal and Wright (2006)). The idea behind the BFGS (and, more generally, behind quasi-Newton) methods is to compute the Hessian approximation $\mathbf{B}^{t+1}$ using only first-order information. Like Newton methods, BFGS methods work with stepsize $\eta_t = 1$ when the iterates are close to the optimum. However, at the initial stages of the algorithm, the stepsize is typically determined by a line search for avoiding the method to diverge.

A common rule for the Hessian approximation is to choose it to satisfy the secant condition $\mathbf{B}^{t+1}\mathbf{s}^{t+1} = \mathbf{y}^{t+1}$,where $\mathbf{s}^{t+1} = \mathbf{x}^t - \mathbf{x}^{t-1}$, and $\mathbf{y}^{t+1} = \nabla f(\mathbf{x}^t) - \nabla f(\mathbf{x}^{t-1})$ are called the *variable variation* and *gradient variation* vectors, respectively. The Hessian approximation update of BFGS which satisfies the secant condition can be written as a rank-two update

$$\begin{aligned} \mathbf{B}^{t+1} &= \mathbf{B}^t + \mathbf{U}^{t+1} + \mathbf{V}^{t+1}, \\ \mathbf{U}^{t+1} &= \frac{\mathbf{y}^{t+1}(\mathbf{y}^{t+1})^T}{(\mathbf{y}^{t+1})^T\mathbf{s}^{t+1}}, \\ \mathbf{V}^{t+1} &= -\frac{\mathbf{B}^t\mathbf{s}^{t+1}(\mathbf{s}^{t+1})^T\mathbf{B}^t}{(\mathbf{s}^{t+1})^T\mathbf{B}^t\mathbf{s}^{t+1}}. \end{aligned} \tag{3}$$

Note that both matrices $\mathbf{U}^t$ and $\mathbf{V}^t$ are rank-one. Therefore, the update (3) is rank two. Owing to this property, the inverse of the Hessian approximation $\mathbf{B}^{t+1}$ can be computed at a low cost of $\mathcal{O}(p^2)$ arithmetic iterations based on the Woodbury-Morrison formula, instead of computing the inverse matrix directly with a complexity of $\mathcal{O}(p^3)$. For a strongly convex function f with the global minimum $\mathbf{x}^*$, a classical convergence result for the BFGS method shows that the iterates generated by BFGS are *superlinearly* convergent (Broyden et al. (1973b)), i.e. $\lim_{t\to\infty} \frac{\|\mathbf{x}^{t+1}-\mathbf{x}^*\|}{\|\mathbf{x}^t-\mathbf{x}^*\|} = 0$. There are also limited-memory BFGS (L-BFGS) methods that require less memory ($\mathcal{O}(p)$) at the expense of having a linear (but not superlinear) convergence (Nocedal and Wright (2006)). Our main goal in this paper is to design a BFGS-type method that can solve problem (1) efficiently with superlinear convergence in an asynchronous setting under the master/slave communication model. We introduce our proposed algorithm in the following section.

2.2 A Distributed Averaged Quasi-Newton Method (DAve-QN)

In this section, we introduce a BFGS-type method that can be implemented in a distributed setting (master/slave) without any central coordination between the nodes, i.e., asynchronously. To do so, we consider a setting where n worker nodes (machines) are connected to a master node. Each worker node i has access to a component of the global objective function, i.e., node i

has access only to the function f_i. The decision variable stored at the master node is denoted by x^t at time t. At each moment t, d_i^t denotes the delay in communication with the i-th worker, i.e., the last exchange with this worker was at time $t-d_i^t$. For convenience, if the last communication was performed exactly at moment t, then we set $d_i^t = 0$. In addition, D_i^t denotes the double delay in communication, which relates to the penultimate communication and can be expressed as follows:

$$D_i^t = d_i^t + d_i^{t-d_i^t-1} + 1.$$

Note that the time index t increases if one of the workers performs an update. For example, if worker i communicates with the master at iterations 5, 8 and 13, then we have $d_i^9 = 1$, $d_i^7 = 2$, $D_i^9 = 4$.

Every worker node i has two copies of the decision variable corresponding to the last two communications with the master, i.e. node i possesses $\mathbf{x}^{t-d_i^t}$ and $\mathbf{z}_i^t := \mathbf{x}^{t-D_i^t}$. Since there has been no communication after $t-d_i^t$, we will clearly have

$$\mathbf{z}_i^{t-d_i^t} = \mathbf{z}_i^t = \mathbf{x}^{t-D_i^t}. \tag{4}$$

We are interested in designing a distributed version of the BFGS method described in Section 2.1, where each node at time t has an approximation $\mathbf{B}_i^t$ to the local Hessian (Hessian matrix of f_i) where $\mathbf{B}_i^t$ is constructed based on the local delayed decision variables $\mathbf{x}^{t-d_i^t}$ and $\mathbf{z}_i^t$, and therefore the local Hessian approximation $\mathbf{B}_i^t$ will also be outdated, i.e. it will satisfy

$$\mathbf{B}_i^t = \mathbf{B}_i^{t-d_i^t}. \tag{5}$$

An instance of the setting that we consider in this paper is illustrated in Figure 1. At time t, one of the workers, say i_t, finishes its task and sends a group of vectors and scalars (that we will precise later) to the master node, avoiding communication of any $p \times p$ matrices as it is assumed that this would be prohibitively expensive communication-wise. Then, the master node uses this information to update the decision variable $\mathbf{x}^t$ using the new information of node i_t and the old information of the remaining workers. After this process, master sends the updated information to node i_t.

We define the *aggregate Hessian approximation* as

$$\mathbf{B}^t := \sum_{i=1}^n \mathbf{B}_i^t = \sum_{i=1}^n \mathbf{B}_i^{t-d_i^t} \tag{6}$$

where we used (5). In addition, we introduce

$$\mathbf{u}^t := \sum_{i=1}^n \mathbf{B}_i^t \mathbf{z}_i^t = \sum_{i=1}^n \mathbf{B}_i^{t-d_i^t} \mathbf{z}_i^{t-d_i^t},$$
$$\mathbf{g}^t := \sum_{i=1}^n \nabla f_i(\mathbf{z}_i^t) = \sum_{i=1}^n \nabla f_i(\mathbf{z}_i^{t-d_i^t}), \tag{7}$$

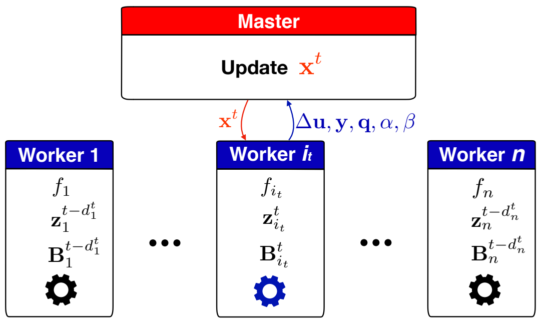


Figure 1: Asynchronous communication scheme used by the proposed algorithm.

as the *aggregate Hessian-variable product* and *aggregate gradient* respectively where we made use of the identities (4)–(5). All these vectors and matrices are only available at the master node since it requires access to the information of all the workers.

Given that at step $t+1$ only a single index i_t is updated, using the identities (4)–(7), it follows that the master has the update rules

$$\mathbf{B}^{t+1} = \mathbf{B}^t + (\mathbf{B}_{i_t}^{t+1} - \mathbf{B}_{i_t}^t) = \mathbf{B}^t + \left(\mathbf{B}_{i_t}^{t+1} - \mathbf{B}_{i_t}^{t-d_i^t}\right), \tag{8}$$

$$\begin{aligned}\mathbf{u}^{t+1} &= \mathbf{u}^t + (\mathbf{B}_{i_t}^{t+1}\mathbf{z}_{i_t}^{t+1} - \mathbf{B}_{i_t}^t \mathbf{z}_{i_t}^t)\\ &= \mathbf{u}^t + \left(\mathbf{B}_{i_t}^{t+1}\mathbf{x}^{t-d_{i_t}^t} - \mathbf{B}_{i_t}^{t-d_{i_t}^t}\mathbf{x}^{t-D_{i_t}^t}\right),\end{aligned} \tag{9}$$

$$\begin{aligned}\mathbf{g}^{t+1} &= \mathbf{g}^t + (\nabla f_{i_t}(\mathbf{z}_{i_t}^{t+1}) - \nabla f_{i_t}(\mathbf{z}_{i_t}^t))\\ &= \mathbf{g}^t + \left(\nabla f_{i_t}(\mathbf{x}^{t-d_{i_t}^t}) - \nabla f_{i_t}(\mathbf{x}^{t-D_{i_t}^t})\right).\end{aligned} \tag{10}$$

We observe that, only $\mathbf{B}_{i_t}^{t+1}$ and $\nabla f_{i_t}(\mathbf{z}_{i_t}^{t+1}) = \nabla f_{i_t}(\mathbf{x}^{t-d_{i_t}^t})$ are required to be computed at step $t+1$. The former is obtained by the standard BFGS rule applied to f_i carried out by the worker i_t:

$$\mathbf{B}_{i_t}^{t+1} = \mathbf{B}_{i_t}^t + \frac{\mathbf{y}_{i_t}^{t+1}(\mathbf{y}_{i_t}^{t+1})^\top}{\alpha^{t+1}} - \frac{\mathbf{q}_{i_t}^{t+1}(\mathbf{q}_{i_t}^{t+1})^\top}{\beta^{t+1}} \tag{11}$$

with

$$\alpha^{t+1} := (\mathbf{y}_{i_t}^{t+1})^\top \mathbf{s}_{i_t}^{t+1},$$
$$\mathbf{y}_{i_t}^{t+1} := \mathbf{z}_{i_t}^{t+1} - \mathbf{z}_{i_t}^t = \mathbf{x}^{t-d_{i_t}^t} - \mathbf{x}^{t-D_{i_t}^t}, \tag{12}$$
$$\mathbf{q}_{i_t}^{t+1} := \mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^{t+1},$$
$$\beta^{t+1} := (\mathbf{s}_{i_t}^{t+1})^\top \mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^{t+1} = (\mathbf{s}_{i_t}^{t+1})^\top \mathbf{q}_{i_t}^{t+1}. \tag{13}$$

Then, the master computes the new iterate as $\mathbf{x}^{t+1} = (\mathbf{B}^{t+1})^{-1}\left(\mathbf{u}^{t+1} - \mathbf{g}^{t+1}\right)$ and sends it to worker i_t. For the rest of the workers, we update the time counter without changing the variables, so $\mathbf{z}_i^{t+1} = \mathbf{z}_i^t$ and $\mathbf{B}_i^{t+1} = \mathbf{B}_i^t$

for $i \neq i_t$. Although, updating the inverse $(\mathbf{B}^t)^{-1}$ may seem costly at first glance, in fact it can be computed efficiently in $\mathcal{O}(p^2)$ iterations, similar to standard implementations of the BFGS methods. More specifically, if we introduce a new matrix

$$\mathbf{U}^{t+1} := (\mathbf{B}^t)^{-1} - \frac{(\mathbf{B}^t)^{-1}\mathbf{y}_{i_t}^{t+1}(\mathbf{y}_{i_t}^{t+1})^\top(\mathbf{B}^t)^{-1}}{(\mathbf{y}_{i_t}^{t+1})^\top\mathbf{s}_{i_t}^{t+1} + (\mathbf{y}_{i_t}^{t})^\top(\mathbf{B}^t)^{-1}\mathbf{y}_{i_t}^{t+1}}, \tag{14}$$

then, by the Sherman-Morrison-Woodbury formula, we have the identity

$$(\mathbf{B}^{t+1})^{-1} = \mathbf{U}^{t+1} + \frac{\mathbf{U}^{t+1}(\mathbf{B}_{i_t}^{t-d_{i_t}^t}\mathbf{s}_{i_t}^{t+1})(\mathbf{B}_{i_t}^{t-d_{i_t}^t}\mathbf{s}_{i_t}^{t+1})^T\mathbf{U}^{t+1}}{(\mathbf{s}_{i_t}^{t+1})^T\mathbf{B}_{i_t}^{t-d_{i_t}^t}\mathbf{s}_{i_t}^{t+1} - (\mathbf{B}_{i_t}^{t-d_{i_t}^t}\mathbf{s}_{i_t}^{t+1})^T\mathbf{U}^{t+1}(\mathbf{B}_{i_t}^{t-d_{i_t}^t}\mathbf{s}_{i_t}^{t+1})}. \tag{15}$$

Therefore, if we already have $(\mathbf{B}^t)^{-1}$, it suffices to have only matrix vector products. If we denote $\mathbf{v}^{t+1} = (\mathbf{B}^t)^{-1}\mathbf{y}_{i_t}^{t+1}$ and $\mathbf{w}^{t+1} := \mathbf{U}^{t+1}\mathbf{q}_{i_t}^{t+1}$, then these equations can be simplified as

$$\mathbf{U}^{t+1} = (\mathbf{B}^t)^{-1} - \frac{\mathbf{v}^{t+1}(\mathbf{v}^{t+1})^\top}{\alpha^{t+1} + (\mathbf{v}^{t+1})^\top\mathbf{y}_{i_t}^{t+1}},$$
$$\mathbf{v}^{t+1} = (\mathbf{B}^t)^{-1}\mathbf{y}_{i_t}^{t+1}, \tag{16}$$
$$(\mathbf{B}^{t+1})^{-1} = \mathbf{U}^{t+1} + \frac{\mathbf{w}^{t+1}(\mathbf{w}^{t+1})^\top}{\beta^{t+1} - (\mathbf{q}^{t+1})^\top\mathbf{w}^{t+1}},$$
$$\mathbf{w}^{t+1} := \mathbf{U}^{t+1}\mathbf{q}_{i_t}^{t+1}, \tag{17}$$

where $\alpha^{t+1}, \beta^{t+1}, \mathbf{q}_{i_t}^{t+1}$ and $\mathbf{y}_{i_t}^{t+1}$ are defined by (12)–(13).

The steps of the DAve-QN at the master node and the workers are summarized in Algorithm 1. Note that steps at worker i is devoted to performing the update in (11) based on the update formula (16)–(17). In the supplementary material, we also provide a simplified version of the DAve-QN algorithm that illustrates the update rules from both the master's side and the workers' side further.

Next, we define *epochs* $\{T_m\}_m$ by setting $T_1 = 0$ and the following recursion:

$$\begin{aligned} T_{m+1} \quad &:= \quad \min\big\{t : \text{ each machine made at least two updates on the interval } [T_m, t]\big\} \\ &= \quad \min\{t : t - D_i^t \geq T_m \text{ for all } i = 1, .., M\}. \end{aligned}$$

The proof of the following simple lemma is provided in the supplementary material.

Lemma 1. *Algorithm 1 iterates satisfy*

$$\mathbf{x}^t = \left(\frac{1}{n}\sum_{i=1}^n \mathbf{B}_i^t\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^n \mathbf{B}_i^t\mathbf{z}_i^t - \frac{1}{n}\sum_{i=1}^n \nabla f_i(\mathbf{z}_i^t)\right).$$

The result in Lemma 1 shows that explicit relationship between the updated variable $\mathbf{x}^t$ based on the proposed DAve-QN and the local information at the workers. We will use this update to analyze DAve-QN.

Proposition 1 (Epochs' properties)**.** *The following relations between epochs and delays hold:*

- *For any $t \in [T_{m+1}, T_{m+2})$ and any $i = 1, 2, \ldots, n$ one has $t - D_i^t \in [T_m, t)$.*
- *If delays are uniformly bounded, i.e. there exists a constant d such that $d_i^t \leq d$ for all i and t, then for all m we have $T_{m+1} - T_m \leq D := 2d + 1$ and $T_m \leq Dm$.*
- *If we define average delays as $\overline{d^t} := \frac{1}{n}\sum_{i=1}^n d_i^t$, then $\overline{d^t} \geq (n-1)/2$. Moreover, assuming that $\overline{d^t} \leq (n-1)/2 + \overline{d}$ for all t, we get $T_m \leq 4n(\overline{d}+1)m$.*

Clearly, without visiting every function we can not converge to $\mathbf{x}^*$. Therefore, it is more convenient to measure performance in terms of number of passed epochs, which can be considered as our alternative counter for time. Proposition 1 explains how one can get back to the iterations time counter assuming that delays are bounded uniformly or on average. However, uniform upper bounds are rather pessimistic which motivates the convergence in epochs that we consider.

3 Convergence Analysis

In this section, we study the convergence properties of the proposed distributed asynchronous quasi-Newton method. To do so, we first assume that the following conditions are satisfied.

Assumption 1. *The component functions f_i are L-smooth and μ-strongly convex, i.e., there exist positive constants $0 < \mu \leq L$ such that, for all i and $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^p$*

$$\mu\|\mathbf{x} - \hat{\mathbf{x}}\|^2 \leq (\nabla f_i(\mathbf{x}) - \nabla f_i(\hat{\mathbf{x}}))^T(\mathbf{x} - \hat{\mathbf{x}}) \leq L\|\mathbf{x} - \hat{\mathbf{x}}\|^2. \tag{18}$$

Assumption 2. *The Hessians $\nabla^2 f_i$ are Lipschitz continuous, i.e., there exists a positive constant $\tilde{L}$ such that, for all i and $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^p$, we can write $\|\nabla^2 f_i(\mathbf{x}) - \nabla^2 f_i(\hat{\mathbf{x}})\| \leq \tilde{L}\|\mathbf{x} - \hat{\mathbf{x}}\|$.*

It is well-known and widely used in the literature on Newton's and quasi-Newton methods (Nesterov (2013); Broyden et al. (1973b); Powell (1971); J. E. Dennis and More (1974)) that if the function f_i has Lipschitz continuous Hessian $x \mapsto \nabla^2 f_i(x)$ with parameter $\tilde{L}$ then

$$\begin{aligned} &\left\|\nabla^2 f_i(\tilde{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}}) - (\nabla f_i(\mathbf{x}) - \nabla f_i(\hat{\mathbf{x}}))\right\| \\ &\leq \tilde{L}\|\mathbf{x} - \hat{\mathbf{x}}\| \max\{\|\mathbf{x} - \tilde{\mathbf{x}}\|, \|\hat{\mathbf{x}} - \tilde{\mathbf{x}}\|\}, \end{aligned} \tag{19}$$

Algorithm 1 (DAve-QN)

Master:

```
Initialize $\mathbf{x}^0$, $\mathbf{B}_i^0$, $(\mathbf{B}^0)^{-1} = (\sum_{i=1}^n \mathbf{B}_i^0)^{-1}$, $\mathbf{u}^0 = \sum_{i=1}^n \mathbf{B}_i^0\mathbf{x}^0$, $\mathbf{g}^0 = \sum_{i=1}^n \nabla f_i(\mathbf{x}^0)$
for $t = 1$ to $T$-$1$ do
    If a worker sends an update:
    Receive $\Delta\mathbf{u}^t$, $\mathbf{y}$, $\mathbf{q}$, $\alpha^t$, $\beta^t$ from it
    $\mathbf{u}^t = \mathbf{u}^{t-1} + \Delta\mathbf{u}^t$
    $\mathbf{g}^t = \mathbf{g}^{t-1} + \mathbf{y}$
    $\mathbf{v}^t = (\mathbf{B}^{t-1})^{-1}\mathbf{y}$
    $\mathbf{U}^t = (\mathbf{B}^{t-1})^{-1} - \frac{\mathbf{v}^t\mathbf{v}^{t\top}}{\alpha^t + \mathbf{v}t^\top\mathbf{y}}$
    $\mathbf{w}^t = \mathbf{U}^t\mathbf{q}$
    $(\mathbf{B}^t)^{-1} = \mathbf{U}^t + \frac{\mathbf{w}^t\mathbf{w}^{t\top}}{\beta^t - \mathbf{q}^T\mathbf{w}^t}$
    $\mathbf{x}^t = (\mathbf{B}^t)^{-1}(\mathbf{u}^t - \mathbf{g}^t)$
    Send $\mathbf{x}^t$ to the slave in return
end
Interrupt all slaves
Output $\mathbf{x}^T$
```

Slave i:

```
Initialize $\mathbf{x}_i^0 = \mathbf{x}^0$, $\mathbf{B}_i^0$
while not interrupted by master do
    Receive $\mathbf{x}^{t-D_i^t}$ at moment $t - D_i^t$
    Perform below steps by moment $t - d_i^t$
    $\mathbf{z}_i^t = \mathbf{z}_i^{t-d_i^t} = \mathbf{x}^{t-D_i^t}$
    $\mathbf{s}_i^t = \mathbf{s}_i^{t-d_i^t} = \mathbf{z}^{t-d_i^t} - \mathbf{z}_i^{t-D_i^t}$
    $\mathbf{y}_i^t = \mathbf{y}_i^{t-d_i^t} = \nabla f_i(\mathbf{x}^{t-d_i^t}) - \nabla f_i(\mathbf{z}_i^t)$
    $\mathbf{q}_i^t = \mathbf{q}_i^{t-d_i^t} = \mathbf{B}_i^{t-D_i^t}\mathbf{s}_i^{t-d_i^t}$
    $\alpha^{t-d_i^t} = \mathbf{y}_i^{tT}\mathbf{s}_i^t$
    $\beta^{t-d_i^t} = (\mathbf{s}_i^{t-d_i^t})^T\mathbf{B}_i^{t-D_i^t}\mathbf{s}_i^t$
    $\mathbf{B}_i^{t-d_i^t} = \mathbf{B}_i^{t-D_i^t} + \frac{\mathbf{y}_i^t\mathbf{y}_i^{tT}}{\alpha^{t-d_i^t}} - \frac{\mathbf{q}_i^t\mathbf{q}_i^{tT}}{\beta^{t-d_i^t}}$
    $\Delta\mathbf{u}^{t-d_i^t} = \mathbf{B}_i^{t-d_i^t}\mathbf{z}_i^{t-d_i^t} - \mathbf{B}_i^{t-D_i^t}\mathbf{z}_i^{t-D_i^t}$
    Send $\Delta\mathbf{u}^{t-d_i^t}, \mathbf{y}_i^{t-d_i^t}, \mathbf{q}_i^{t-d_i^t}, \alpha^{t-d_i^t}, \beta^{t-d_i^t}$ to the master at moment $t - d_i^t$
end
```

for any arbitrary $\mathbf{x}, \tilde{\mathbf{x}}, \hat{\mathbf{x}} \in \mathbb{R}^p$. See, for instance, Lemma 3.1 in Broyden et al. (1973b).

Lemma 2. *Consider the DAve-QN algorithm summarized in Algorithm 1. For any i, define the residual sequence for function f_i as $\sigma_i^t := \max\{\|\mathbf{z}_i^t - \mathbf{x}^*\|, \|\mathbf{z}_i^{t-D_i^t} - \mathbf{x}^*\|\}$ and set $\mathbf{M}_i = \nabla^2 f_i(\mathbf{x}^*)^{-1/2}$. If Assumptions 1 and 2 hold and the condition $\sigma_i^t < \mu/(3\tilde{L})$ is satisfied then a Hessian approximation matrix $\mathbf{B}_i^t$ and its last updated version $\mathbf{B}_i^{t-D_i^t}$ satisfy*

$$\left\|\mathbf{B}_i^t - \nabla^2 f_i(\mathbf{x}^*)\right\|_{\mathbf{M}_i} \leq \left[\left[1 - \alpha\theta_i^{t-D_i^t\,2}\right]^{\frac{1}{2}} + \alpha_3\sigma_i^{t-D_i^t}\right]. \left\|\mathbf{B}_i^{t-D_i^t} - \nabla^2 f_i(\mathbf{x}^*)\right\|_{\mathbf{M}_i} + \alpha_4\sigma_i^{t-D_i^t}, \tag{20}$$

where α, α_3, and α_4 are some positive constants and $\theta_i^{t-D_i^t} = \frac{\|\mathbf{M}_i(\mathbf{B}_i^{t-D_i^t} - \nabla^2 f_i(\mathbf{x}^*))\mathbf{s}_i^{t-D_i^t}\|}{\|\mathbf{B}_i^{t-D_i^t} - \nabla^2 f_i(\mathbf{x}^*)\|_{\mathbf{M}_i}\|\mathbf{M}_i^{-1}\mathbf{s}_i^{t-D_i^t}\|}$ with the convention that $\theta_i^{t-D_i^t} = 0$ in the special case $\mathbf{B}_i^{t-D_i^t} = \nabla^2 f_i(\mathbf{x}^*)$.

Lemma 2 shows that, if we neglect the additive term $\alpha_4\sigma_i^{t-D_i^t}$ in (20), the difference between the Hessian approximation matrix $\mathbf{B}_i^t$ for the function f_i and its corresponding Hessian at the optimal point $\nabla^2 f_i(\mathbf{x}^*)$ decreases by following the update of Algorithm 1. To formalize this claim and show that the additive term is negligible, we prove in the following lemma that the sequence of errors $\|\mathbf{x}^t - \mathbf{x}^*\|$ converges to zero R-linearly which also implies linear convergence of the sequence σ_i^t.

Lemma 3. *Consider the DAve-QN method outlined in Algorithm 1. Further assume that the conditions in Assumptions 1 and 2 are satisfied. Then, for any $r \in (0, 1)$ there exist positive constants $\epsilon(r)$ and $\delta(r)$ such that if $\|\mathbf{x}^0 - \mathbf{x}^*\| < \epsilon(r)$ and $\|\mathbf{B}_i^0 - \nabla^2 f_i(\mathbf{x}^*)\|_{\mathbf{M}} < \delta(r)$ for $\mathbf{M} = \nabla^2 f_i(\mathbf{x}^*)^{-1/2}$ and $i = 1, 2, \ldots, n$, the sequence of iterates generated by DAve-QN satisfy*

$$\|\mathbf{x}^t - \mathbf{x}^*\| \leq r^m\|\mathbf{x}^0 - \mathbf{x}^*\|$$

for all $t \in [T_m, T_{m+1})$.

The result in Lemma 3 shows that the error for the sequence of iterates generated by the DAve-QN method converge to zero at least linearly in a neighborhood of the optimal solution. Using this result, in the following theorem we prove our main result, which shows a specific form of superlinear convergence.

Remark. The convergence results in Lemma 3 are in terms of the epochs T_m. This can be stated in terms of upper bounds on the delays, d_i^t or their average value as they are controlled by the quantity T_m.

Theorem 1. *Consider the proposed method outlined in Algorithm 1. Suppose that Assumptions 1 and 2 hold. Further, assume that the required conditions for the results in Lemma 2 and Lemma 3 are satisfied. Then,* *the sequence of residuals* $\|\mathbf{x}^t - \mathbf{x}^*\|$ *satisfies*

$$\lim_{t\to\infty} \frac{\max_{t\in[T_{m+1},T_{m+2})} \|\mathbf{x}^t - \mathbf{x}^*\|}{\max_{t\in[T_m,T_{m+1})} \|\mathbf{x}^t - \mathbf{x}^*\|} = 0.$$

The result in Theorem 1 shows that the maximum residual in an epoch divided by the the maximum residual for the previous epoch converges to zero. This observation shows that there exists a subsequence of residuals $\|\mathbf{x}^t - \mathbf{x}^*\|$ that converges to zero superlinearly.

Extension to non-strongly convex objectives: For convex objectives f that are not strongly convex, the Hessian matrix $\nabla^2 f(x)$ can be singular which causes stability issues with the Hessian approximation of BFGS-type methods. A popular approach to solve this issue is to add a regularizing $\varepsilon\|x\|^2$ term to the objective for an appropriately chosen $\varepsilon > 0$ that is small enough. This leads to a strongly convex approximation of the objective and regularizes the Hessian. With this approach, our algorithm and analysis can be applied to convex functions that are not strongly convex if the domain is compact, following similar ideas to (Lessard et al., 2016, Section 5.4).

4 Experiments

We conduct our experiments on four datasets (`covtype`, `SUSY`, `mnist8m`, `cifar10`) from the LIBSVM library (Chang and Lin (2011)).[1] For the first two datasets, the objective considered is a binary logistic regression problem $f(x) = \frac{1}{n}\sum_{i=1}^n \log(1+\exp(-b_i a_i^T x) + \frac{\lambda}{2}\|x\|^2$ where $a_i \in \mathbb{R}^p$ are the feature vectors and $b_i \in \{-1,+1\}$ are the labels. The other two datasets are about multi-class classification instead of binary classification. For comparison, we used two other algorithms designed for distributed optimization:

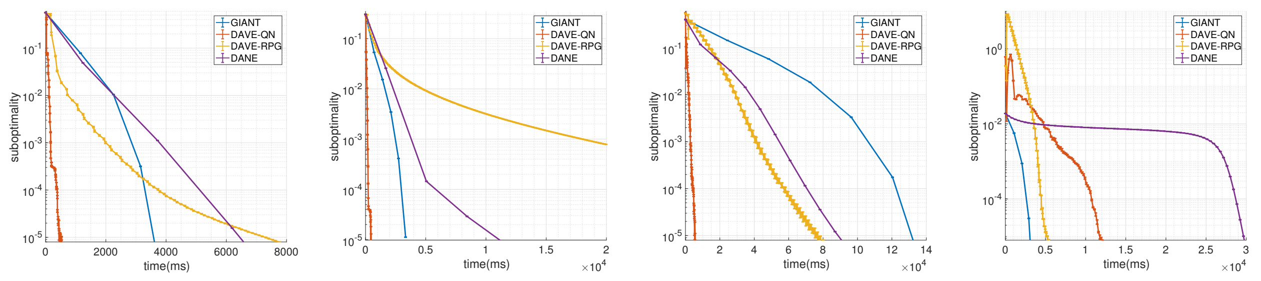
- Distributed Average Repeated Proximal Gradient (**DAve-RPG**, Mishchenko et al. (2018)) . It is a recently proposed competitive state-of-the-art asynchronous method for first-order distributed optimization, numerically demonstrated to outperform incremental aggregated gradient methods (Gürbüzbalaban et al. (2017); Vanli et al. (2018)) and synchronous proximal gradient methods (Mishchenko et al. (2018)).
- Distributed Approximate Newton (**DANE**, Shamir et al. (2014)). This is a well-known Newton-like method that does not require a parameter server node, but performs reduce operations at every step.
- Globally Improved Approximate Newton (**GIANT**, Wang et al. (2018)). It is a distributed synchronous communication-efficient Newton-type method that reduces to the DANE method for quadratic objectives. For more general smooth objectives with a Lipschitz Hessian, it enjoys a local linear-quadratic convergence rate. It has also been shown in Wang et al. (2018) to numerically outperform the DANE (Shamir et al. (2014)), L-BFGS (Liu and Nocedal (1989)) and Accelerated Gradient Descent (Nesterov (2013)) methods on a number of problems.

In our experiments, we did not implement algorithms that require shared memory (such as ASAGA , Leblond et al. (2017) or Hogwild!, Recht et al. (2011)) because in our setting of master/worker communication model, the memory is not shared. Since the focus of this paper is mainly on asynchronous algorithms where the communication delays is the main bottleneck, for fairness reasons, we are also not comparing our method with some other synchronous algorithms such as DISCO (Zhang and Lin (2015)) that would not support asynchronous computations. Our code is publicly available at `https://github.com/DAve-QN/source`.

The experiments are conducted on XSEDE Comet CPUs (Intel Xeon E5-2680v3 2.5 GHz)Towns et al. (2014). For DAve-QN and DAve-RPG we build a cluster of 17 processes in which 16 of the processes are workers and one is the master. The DANE method does not require a master so we use 16 workers for its experiments. We split the data randomly among the processes so that each has the same amount of samples. In our experiments, Intel MKL 11.1.2 and MVAPICH2/2.1 are used for the BLAS (sparse/dense) operations and we use MPI programming compiled with mpicc 14.0.2. Each experiment is repeated thirty times and the average is reported.

For the methods' parameters the best options provided by the method authors are used. For DAve-QN, we use an unit step size which is asymptotically optimal. We observed that DAve-QN converged to the optimal point for stepsize=1 (although sometimes progress was a bit slower in the beginning) and therefore we used it in all experiments. For DAve-RPG the stepsize $\frac{1}{L}$ is used where L is found by a standard backtracking line search similar to Schmidt et al. (2015). DANE has two parameters, η and μ. As recommended by the authors, we use $\eta = 1$ and $\mu = 3\lambda$. We tuned λ to the dataset, choosing $\lambda = 1$ for the `mnist8m` and `cifar10` datasets, $\lambda = 0.001$ for `SUSY` and $\lambda = 0.1$ for the `covtype`. The results we obtained were also qualitatively similar, when we used the common choice $\lambda = 1/n$ for all the methods. Since DANE requires a local optimization problem to be solved, we use SVRG

[1]We use all the datasets without any pre-processing except for the smaller-scale covtype dataset, which we enlarged 5 times for bigger scale experiments using the approach in Wang et al. (2017).



(a) covtype$(54, \approx 2.9M)$ (b) SUSY$(18, 5M)$ (c) mnist8m$(784, 8.1M)$ (d) cifar10$(3072, 60K)$

Figure 2: Expected suboptimality versus time. The first and second numbers indicated next to the name of the datasets are the variables p and n respectively.

(Johnson and Zhang (2013)) as its local solver where its parameters are selected based on the experiments in Shamir et al. (2014).

We note that DAve-QN has a computation complexity of $O(p^2)$ per iteration which is due to the matrix-vector multiplications, a memory complexity of $O(p^2)$ because of storing a $p \times p$ matrix on the workers and master and a communication complexity of $O(p)$ as it only requires to send some scalars and vectors in $\mathbb{R}^p$. More specifically, DAve-QN exchanges $3p + 2$ numbers (slave to master) and p numbers (master to slave), whereas DAve-RPG exchanges $2p$ numbers and DANE exchanges $4p$ numbers (counting both slave and master side) which are all $O(p)$.

Our results are summarized in Figure 2 where we report the average suboptimality versus time in a logarithmic y-axis. For linearly convergent algorithms, the slope of the performance curves determines the convergence rate. The plots contain the error bars based on 5 runs. Since we are using silent machines from XSEDE/Comet that run an experiment in isolation, we do not see much variability in the runs. DANE method is the slowest on these datasets, but it does not need a master, therefore it can apply to multi-agent applications (Nedic and Ozdaglar (2009)) where master nodes are often not available. We observe that DAve-QN performs significantly better on all the datasets except `cifar10`, illustrating the superlinear convergence behavior provided by our theory compared to other methods. For the `cifar10` dataset, p is the largest and GIANT is the fastest. Although DAve-QN starts faster than DAve-RPG, DAve-RPG has a cheaper iteration complexity ($O(p)$ compared to $O(p^2)$ of DAve-QN) and becomes eventually faster.

5 Conclusion and Future Work

In this paper, we focus on the problem of minimizing a large-scale empirical risk minimization in a distributed manner. We used an asynchronous architecture which requires no global coordination between the master node and the workers. Unlike distributed first-order methods that follow the gradient direction to update the iterates, we proposed a distributed averaged quasi-Newton (DAve-QN) algorithm that uses a quasi-Newton approximate Hessian of the workers' local objective function to update the decision variable. In contrast to second-order methods that require computation of the local functions Hessians, the proposed DAve-QN only uses gradient information to improve the convergence of first-order methods in ill-conditioned settings. Therefore, the computational cost of each iteration of DAve-QN is $\mathcal{O}(p^2)$, while the size of the vectors that are communicated between the master and workers is $\mathcal{O}(p)$. Our theoretical results show that the sequence of iterates generated at the master node by following the update of DAve-QN converges superlinearly to the optimal solution when the objective functions at the workers are smooth and strongly convex. Our results hold for both bounded delays and unbounded delays with a bounded time-average. Numerical experiments illustrate the performance of our method.

The choice of the stepsize in the initial stages of the algorithm is the key to get good overall iteration complexity for second-order methods. Investigating several line search techniques developed for BFGS and adapting it to the distributed asynchronous setting is a future research direction of interest. Another promising direction would be developing Newton-like methods that can go beyond superlinear convergence while preserving communication complexity. Finally, investigating the dependence of the convergence properties on the sample size m_i of each machine i would be interesting, in particular one would expect the performance in terms of communication complexity to improve if the sample size of each machine is increased.

Acknowledgements

This work was supported in part by the U.S. NSF awards NSF DMS-1723085, NSF CCF-1814888, NSF CCF-1657175, NSERC Discovery Grants (RGPIN-

06516, DGECR-00303), and the Canada Research Chairs program. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) Towns et al. (2014), which is supported by National Science Foundation grant number ACI-1548562.

References

Agarwal, A. and Duchi, J. C. (2011). Distributed delayed stochastic optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 873–881. Curran Associates, Inc.

Arjevani, Y. and Shamir, O. (2015). Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems*, pages 1756–1764.

Bertsekas, D. P. and Tsitsiklis, J. N. (1989). *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc.

Bianchi, P., Hachem, W., and Iutzeler, F. (2015). A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization. *IEEE Transactions on Automatic Control*, 61(10):2947–2957.

Broyden, C. G., Dennis Jr, J., and Moré, J. J. (1973a). On the local and superlinear convergence of quasi-newton methods. *IMA Journal of Applied Mathematics*, 12(3):223–245.

Broyden, C. G., Jr., J. E. D., Wang, and More, J. J. (1973b). On the local and superlinear convergence of quasi-Newton methods. *IMA J. Appl. Math*, 12(3):223–245.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.

Chen, T., Giannakis, G., Sun, T., and Yin, W. (2018). Lag: Lazily aggregated gradient for communication-efficient distributed learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 5050–5060. Curran Associates, Inc.

Crane, R. and Roosta, F. (2019). Dingo: Distributed newton-type method for gradient-norm optimization. *arXiv preprint arXiv:1901.05134*.

Cutkosky, A. and Busa-Fekete, R. (2018). Distributed stochastic optimization via adaptive sgd. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 1910–1919. Curran Associates, Inc.

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014a). Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing systems*, pages 1646–1654.

Defazio, A., Domke, J., and Caetano, T. (2014b). Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the 31st international conference on machine learning (ICML-14)*, pages 1125–1133.

Dennis, J. E. and Moré, J. J. (1974). A characterization of superlinear convergence and its application to quasi-newton methods. *Mathematics of computation*, 28(126):549–560.

Dongarra, J., Hittinger, J., Bell, J., Chacon, L., Falgout, R., Heroux, M., Hovland, P., Ng, E., Webster, C., and Wild, S. (2014). Applied mathematics research for exascale computing. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

Dünner, C., Lucchi, A., Gargiani, M., Bian, A., Hofmann, T., and Jaggi, M. (2018). A Distributed Second-Order Algorithm You Can Trust. *arXiv e-prints*, page arXiv:1806.07569.

Eisen, M., Mokhtari, A., and Ribeiro, A. (2017). Decentralized quasi-Newton methods. *IEEE Transactions on Signal Processing*, 65(10):2613–2628.

Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26.

Gürbüzbalaban, M., Ozdaglar, A., and Parrilo, P. (2015). A globally convergent incremental Newton method. *Mathematical Programming*, 151(1):283–313.

Gürbüzbalaban, M., Ozdaglar, A., and Parrilo, P. A. (2017). On the convergence rate of incremental aggregated gradient algorithms. *SIAM Journal on Optimization*, 27(2):1035–1048.

J. E. Dennis, J. and More, J. J. (1974). A characterization of super linear convergence and its application to quasi-Newton methods. *Mathematics of computation*, 28(126):549–560.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26, Lake Tahoe, Nevada, United States*, pages 315–323.

Leblond, R., Pedregosa, F., and Lacoste-Julien, S. (2017). ASAGA: Asynchronous Parallel SAGA. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine*

Learning Research, pages 46–54, Fort Lauderdale, FL, USA. PMLR.

Lee, C.-p., Lim, C. H., and Wright, S. J. (2018). A distributed quasi-Newton algorithm for empirical risk minimization with nonsmooth regularization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1646–1655. ACM.

Lee, J. D., Lin, Q., Ma, T., and Yang, T. (2017). Distributed stochastic variance reduced gradient methods by sampling extra data with replacement. *Journal of Machine Learning Research*, 18(122):1–43.

Lessard, L., Recht, B., and Packard, A. (2016). Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Ma, C., Smith, V., Jaggi, M., Jordan, M., Richtárik, P., and Takác, M. (2015). Adding vs. averaging in distributed primal-dual optimization. In *International Conference on Machine Learning*, pages 1973–1982.

Mairal, J. (2015). Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855.

Mansoori, F. and Wei, E. (2017). Superlinearly convergent asynchronous distributed network newton method. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2874–2879. IEEE.

Mishchenko, K., Iutzeler, F., Malick, J., and Amini, M.-R. (2018). A delay-tolerant proximal-gradient algorithm for distributed learning. In *International Conference on Machine Learning*, pages 3584–3592.

Mokhtari, A., Eisen, M., and Ribeiro, A. (2018a). IQN: An incremental quasi-Newton method with local superlinear convergence rate. *SIAM Journal on Optimization*, 28(2):1670–1698.

Mokhtari, A., Gürbüzbalaban, M., and Ribeiro, A. (2018b). Surpassing gradient descent provably: A cyclic incremental method with linear convergence rate. *SIAM Journal on Optimization*, 28(2):1420–1447.

Nedic, A. and Ozdaglar, A. (2009). Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48.

Nemirovskii, A., Yudin, D. B., and Dawson, E. R. (1983). *Problem complexity and method efficiency in optimization.* Wiley.

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

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

Pedregosa, F., Leblond, R., and Lacoste-Julien, S. (2017). Breaking the nonsmooth barrier: A scalable parallel method for composite optimization. In *Advances in Neural Information Processing Systems*, pages 56–65.

Peng, Z., Xu, Y., Yan, M., and Yin, W. (2016). Arock: An algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879.

Powell, M. J. (1976). Some global convergence properties of a variable metric algorithm for minimization without exact line searches. *Nonlinear programming*, 9(1):53–72.

Powell, M. J. D. (1971). *Some global convergence properties of a variable metric algorithm for minimization without exact line search.* Academic Press, London, UK, 2 edition.

Recht, B., Re, C., Wright, S., and Niu, F. (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701.

Reddi, S. J., Konečnỳ, J., Richtárik, P., Póczós, B., and Smola, A. (2016). Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879.*

Roux, N. L., Schmidt, M., and Bach, F. R. (2012). A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671.

Schmidt, M., Babanezhad, R., Ahmed, M., Defazio, A., Clifton, A., and Sarkar, A. (2015). Non-uniform stochastic average gradient method for training conditional random fields. In *Artificial Intelligence and Statistics*, pages 819–828.

Shamir, O., Srebro, N., and Zhang, T. (2014). Communication-efficient distributed optimization using an approximate Newton-type method. In *International conference on machine learning*, pages 1000–1008.

Şimşekli, U., Yıldız, Ç., Nguyen, T. H., Richard, G., and Cemgil, A. T. (2018). Asynchronous stochastic quasi-newton mcmc for non-convex optimization. *arXiv preprint arXiv:1806.02617.*

Smith, V., Forte, S., Ma, C., Takac, M., Jordan, M. I., and Jaggi, M. (2016). CoCoA: A General Framework for Communication-Efficient Distributed Optimization. *ArXiv e-prints*.

Takáč, M., Richtárik, P., and Srebro, N. (2015). Distributed Mini-Batch SDCA. *arXiv e-prints*, page arXiv:1507.08322.

Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G. D., Roskies, R., Scott, J. R., and Wilkins-Diehr, N. (2014). Xsede: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74.

Vanli, N. D., Gürbüzbalaban, M., and Ozdaglar, A. (2016). Global convergence rate of proximal incremental aggregated gradient methods. *arXiv preprint arXiv:1608.01713*.

Vanli, N. D., Gurbuzbalaban, M., and Ozdaglar, A. (2018). Global convergence rate of proximal incremental aggregated gradient methods. *SIAM Journal on Optimization*, 28(2):1282–1300.

Wai, H.-T., Freris, N. M., Nedic, A., and Scaglione, A. (2018a). Sucag: Stochastic unbiased curvature-aided gradient method for distributed optimization. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1751–1756. IEEE.

Wai, H.-T., Freris, N. M., Nedic, A., and Scaglione, A. (2018b). Sucag: Stochastic unbiased curvature-aided gradient method for distributed optimization. *arXiv preprint arXiv:1803.08198*.

Wang, S., Roosta-Khorasani, F., Xu, P., and Mahoney, M. W. (2017). GIANT: Globally Improved Approximate Newton Method for Distributed Optimization. *ArXiv e-prints*.

Wang, S., Roosta-Khorasani, F., Xu, P., and Mahoney, M. W. (2018). Giant: Globally improved approximate newton method for distributed optimization. In *Advances in Neural Information Processing Systems*, pages 2332–2342.

Xiao, L., Yu, A. W., Lin, Q., and Chen, W. (2019). Dscovr: Randomized primal-dual block coordinate algorithms for asynchronous distributed optimization. *Journal of Machine Learning Research*, 20(43):1–58.

Yang, T. (2013). Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 629–637.

Zhang, R. and Kwok, J. (2014). Asynchronous distributed ADMM for consensus optimization. In *International Conference on Machine Learning*, pages 1701–1709.

Zhang, Y. and Lin, X. (2015). Disco: Distributed optimization for self-concordant empirical loss. In *International Conference on Machine Learning*, pages 362–370.