Communication-Efficient Distributed Optimization with Quantized Preconditioners

Foivos Alimisis [1] **Peter Davies** [2] **Dan Alistarh** [2 3]

Abstract

We investigate fast and communication-efficient algorithms for the classic problem of minimizing a sum of strongly convex and smooth functions that are distributed among n different nodes, which can communicate using a limited number of bits. Most previous communication-efficient approaches for this problem are limited to first-order optimization, and therefore have *linear* dependence on the condition number in their communication complexity. We show that this dependence is not inherent: communication-efficient methods can in fact have sublinear dependence on the condition number. For this, we design and analyze the first communication-efficient distributed variants of preconditioned gradient descent for Generalized Linear Models, and for Newton's method. Our results rely on a new technique for quantizing both the preconditioner and the descent direction at each step of the algorithms, while controlling their convergence rate. We also validate our findings experimentally, showing fast convergence and reduced communication.

1. Introduction

Due to the sheer size of modern datasets, many practical instances of large-scale optimization are now *distributed*, in the sense that data and computation are split among several computing nodes, which collaborate to jointly optimize the global objective function. This shift towards distribution induces new challenges, and many classic algorithms have been revisited to reduce distribution costs. These costs are usually measured in terms of the number of bits sent and received by the nodes (*communication complexity*) or by the number of parallel iterations required for convergence (*round complexity*).

In this paper, we focus on the communication (bit) complexity of the classic empirical risk minimization problem

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

where the global d-dimensional cost function f is formed as the average of smooth and strongly-convex local costs f_i, each owned by a different machine, indexed by $i = 1, ..., n$.

This problem has a rich history. The seminal paper of Tsitsiklis & Luo (1986) considered the case $n = 2$, and provided a lower bound of $\Omega(d \log(d/\epsilon))$ for quadratic functions, as well as an almost-matching upper bound for this case, within logarithmic factors. (Here, d is the problem dimension and ϵ is the error-tolerance.)

The problem has concentrated significant attention, given the surge of interest in distributed optimization and machine learning, e.g. (Niu et al., 2011; Jaggi et al., 2014; Alistarh et al., 2016; Nguyen et al., 2018; Ben-Nun & Hoefler, 2019). In particular, a series of papers (Khirirat et al., 2018; Ye & Abbe, 2018; Magnússon et al., 2020; Alistarh & Korhonen, 2020) continued to provide improved upper and lower bounds for the communication complexity of this problem, both for deterministic and randomized algorithms, as well as examining related distributed settings and problems (Scaman et al., 2017; Jordan et al., 2018; Vempala et al., 2020; Mendler-Dünner & Lucchi, 2020; Hendrikx et al., 2020).

The best known lower bound for solving the above problem for deterministic algorithms and general d and n is of

$$\Omega(nd \log(d/\epsilon))$$

total communication bits, given recently by (Alistarh & Korhonen, 2020). This lower bound can be asymptotically matched for quadratic functions by a quantized variant of gradient descent (Magnússon et al., 2020; Alistarh & Korhonen, 2020) using

$$\mathcal{O}(nd\kappa \log \kappa \log(\gamma d/\epsilon))$$

total bits, where κ is the condition number of the problem and γ is the smoothness bound of f.

An intriguing open question concerns the optimal dependency on the condition number for general objectives. While existing lower bounds show no such explicit dependency, all known algorithms have linear (or worse) dependency

[1] Department of Mathematics, University of Geneva, Switzerland (work done while at IST Austria) [2] IST Austria [3] Neural Magic, US Correspondence to: Foivos Alimisis <Foivos.Alimisis@unige.ch>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

on κ. Resolving this problem is non-trivial, since one usually removes this dependency in the non-distributed case by leveraging curvature information in the form of preconditioning or full Newton steps. However, existing distribution techniques are designed for *gradient* quantization, and it is not at all clear for instance how using a preconditioning matrix would interact with the convergence properties of the algorithm, and in particular whether favourable convergence behaviour can be preserved at all following quantization.

Contribution. In this paper, we resolve this question in the positive, and present communication-efficient variants of preconditioned gradient descent for generalized linear models (GLMs) and distributed Newton's method.

Specifically, given a *small enough* error-tolerance ϵ, a communication-efficient variant of preconditioned gradient descent for GLMs (QPGD-GLM) can find an ϵ-minimizer of a γ-smooth function using a total number of bits

$$B^{QPGD\text{-}GLM} = \mathcal{O}\left(nd\kappa_\ell \log(n\kappa_\ell\kappa(M))\log(\gamma D/\epsilon)\right),$$

where d is the dimension, n is the number of nodes, κ_ℓ is the condition number of the loss function ℓ used to measure the distance of training data from the prediction, $\kappa(M)$ is the condition number of the averaged covariance matrix of the training data, and D is a bound on the initial distance from the optimum. In practice, κ_ℓ is often much smaller than the condition number κ of the problem, and is equal to 1 in the case that ℓ is a quadratic.

This first result suggests that distributed methods need not have linear dependence on the condition number of the problem. Our main technical result extends the approach to a distributed variant of Newton's method, showing that the same problem can be solved using

$$B^{Newton} = \mathcal{O}\left(nd^2 \log\left(d\kappa\right)\log(\gamma\mu/\sigma\epsilon)\right) \text{ total bits,}$$

under the assumption that the Hessian is σ-Lipschitz.

Viewed in conjunction with the above $\Omega(nd\log(d/\varepsilon))$ lower bound, these algorithms outline a new communication complexity trade-off between the dependency on the dimension of the problem d, and its condition number κ. Specifically, for ill-conditioned but low-dimensional problems, it may be advantageous to employ quantized Newton's method, whereas QPGD-GLM can be used in cases where the structure of the training data favors preconditioning. Further, our results suggest that there can be no general communication lower bound with linear dependence on the condition number of the problem.

Our results assume the classic coordinator / parameter server (Li et al., 2014) model of distributed computing, in which a distinguished node acts as a coordinator by gathering model updates from the nodes. In this context, we introduce a few tools which should have broader applicability. One is a lattice-based matrix quantization technique, which extends the state-of-the-art vector (gradient) quantization techniques to preconditioners. This enables us to carefully trade off the communication compression achieved by the algorithm with the non-trivial error in the descent directions due to quantization. Our main technical advance is in the context of quantized Newton's method, where we need to keep track of the concentration of quantized Hessians relative to the full-precision version. Further, our algorithms quantize directly the local descent directions obtained by multiplying the inverse of the quantized estimation of the preconditioner with the exact local gradient. This is a non-obvious choice, which turns out to be the correct way to deal with quantized preconditioned methods.

We validate our theoretical results on standard regression datasets, where we show that our techniques can provide an improvement of over 3× in terms of total communication complexity used by the algorithm, while maintaining convergence and solution quality.

Related Work. There has been a surge of interest in distributed optimization and machine learning. While a complete survey is beyond our scope, we mention the significant work on designing and analyzing communication-efficient versions of classic optimization algorithms, e.g. (Jaggi et al., 2014; Scaman et al., 2017; Jordan et al., 2018; Khirirat et al., 2018; Nguyen et al., 2018; Alistarh et al., 2016; 2018; Ye & Abbe, 2018; Ramezani-Kebrya et al., 2019; Magnússon et al., 2020; Ghadikolaei & Magnússon, 2020), and the growing interest in communication and round complexity lower bounds, e.g. (Zhang et al., 2013; Shamir, 2014; Arjevani & Shamir, 2015; Vempala et al., 2020; Alistarh & Korhonen, 2020). In this context, our work is among the first to address the bit complexity of optimization methods which explicitly employ curvature information, and shows that such methods can indeed be made communication-efficient.

Tsitsiklis & Luo (1986) gave the first upper and lower bounds for the communication (bit) complexity of distributed convex optimization, considering the case of two nodes. Their algorithm is a variant of gradient descent which performs *adaptive* quantization, in the sense that nodes adapt the number of bits they send and the quantization grid depending on the iteration. Follow-up work, e.g. (Khirirat et al., 2018; Alistarh et al., 2016) generalized their algorithm to an arbitrary number of nodes, and continued to improve complexity.

In this line, the work closest to ours is that of Magnússon et al. (2020), who introduce a family of adaptive gradient quantization schemes which can enable linear convergence in any norm for gradient-descent-type algorithms, in the same system setting considered in our work. However, we emphasize that this work did *not* consider preconditioning. (Alistarh & Korhonen (2020) also focus on GD, but use different quantizers and a more refined analysis to obtain truly tight communication bounds for quadratics.)

Conceptually, the quantization techniques we introduce serve a similar purpose—to allow the convergence properties of the algorithm to be preserved, despite noisy directional information. At the technical level, however, the schemes we describe and analyze are different, and arguably more complex. For instance, since only the gradient information is quantized, (Magnússon et al., 2020) can use grid quantization adapted to gradient norms, whereas employ more complex quantization, as well as fine-grained bookkeeping with respect to the concentration of quantized matrices and descent directions.

There has been significant work on distributed approximate second-order methods with the different goal of minimizing the *number of communication rounds* required for convergence. One of the first such works is (Shamir et al., 2014), who considered the strongly convex case, and proposed a method called DANE, where each worker solves a subproblem using full gradients at each iteration, and the global iterate is the average of these sub-solutions. Follow-up work (Zhang & Lin, 2015; Reddi et al., 2016; Wang et al., 2018; Zhang et al., 2020) proposed improvements both in terms of generalizing the structure of the loss functions, but also in terms of convergence rates. Recently, Hendrikx et al. (2020) also proposed a round-efficient distributed preconditioned accelerated gradient method for our setting, where preconditioning is done by solving a local optimization problem over a subsampled dataset at the server. Their convergence rate depends on the square root of the relative condition number between the global and local loss functions.

Concurrent work by Islamov et al. (2021) considers the same problem of reducing the bit cost of distributed second-order optimization, and proposes a series of algorithms based on the novel idea of *learning parameters of the Hessian at the optimum* in a communication-efficient manner. The resulting algorithms allow for ℓ_2-regularization, and can achieve local linear and superlinear rates, independent of the condition number, with *linear* communication cost per round in the dimension d.

Relative to our setting, their results require two additional assumptions: The first is that, for linear communication cost, either the coordinator must have access to *all the training data* at the beginning of the optimization process, or the data should be highly *sparse*. The second assumption is on the structure of the individual loss functions, which are weaker than the assumptions we make for our "warm-up" algorithm for GLMs, but stronger than the ones required for our generalized quantized Newton's method. Their results are therefore not directly comparable to ours, however, we note that our communication cost should be lower in e.g. the case where the data is dense and the number of points m is larger than the dimension d. The algorithmic techniques are rather different. Follow-up work extended their approach to the federated learning setting (Safaryan et al., 2021).

2. Preliminaries

Distributed Setting. As discussed, we are in a standard distributed optimization setting, where we have n nodes, and each node i has its own local cost function $f_i : \mathbb{R}^d \to \mathbb{R}$ (where d is the dimension of the problem). We wish to minimize the average cost $f = \frac{1}{n}\sum_{i=1}^n f_i$ and, for that, some communication between nodes is required. We denote the (unique) minimizer of f by x^* and the (unique) minimizer of each f_i by x_i^* (minimizers are unique since these functions are assumed to be strongly convex). Communication may be performed over various network topologies, but in this work we assume a simple structure where an arbitrary node plays the role of the central server, i.e. receives messages from the others, processes them, and finally sends the result back to all. (Such topologies are also common in practice (Li et al., 2014).) Then, the nodes compute an update based on their local cost, and subsequently transmit this information again to the master, repeating the pattern until convergence.

The two main usually considered complexity metrics are the total number of rounds, or iterations, which the algorithm requires, and the total number of bits transmitted. In this paper, we focus on the latter metric, and assume that nodes cannot communicate their information with infinite precision, but instead aim to limit the number of bits that each node can use to encode messages. Thus, we measure complexity in terms of the total number of bits that the optimization algorithm needs to use, in order to minimize f within some accuracy.

Matrix Vectorization. One of the main technical tools of our work is quantization of matrices. All the matrices that we care to quantize turn out to be symmetric. The first step for quantizing is to vectorize them. We do so by using the mapping

$$\phi : \mathbb{S}(d) \to \mathbb{R}^{\frac{d(d+1)}{2}}$$

defined by

$$\phi(P) = (p_{11}, ..., p_{1d}, p_{22}, ..., p_{2d}, ..., p_{dd}),$$

where $P = (p_{ij})_{i,j=1}^d$ and $\mathbb{S}(d)$ is the space of $d \times d$ symmetric matrices. Thus, the mapping ϕ just isolates the upper triangle of a symmetric matrix and writes it as a vector. It is direct to check that ϕ is a linear isomorphism $(\dim(\mathbb{S}(d)) = d(d+1)/2)$.

We can now bound the deformation of distances produced by this mapping for the ℓ_2 norm in $\mathbb{S}(d)$ and the ℓ_2 one in $\mathbb{R}^{\frac{d(d+1)}{2}}$:

Lemma 1. *For any matrices $P, P' \in \mathbb{S}(d)$, we have*

$$\frac{1}{\sqrt{d}}\|\phi(P)-\phi(P')\|_2 \leq \|P-P'\|_2 \leq \sqrt{2}\|\phi(P)-\phi(P')\|_2.$$

The proof can be found in Appendix A.

We will use the isomorphism ϕ later in our applications to

Generalized Linear Models and Newton's method. This is the reason of appearance of the extra d *inside a logarithm* in our upper bounds. From now on we use $\|\cdot\|$ to denote the ℓ_2 norm of either vectors or matrices.

Lattice Quantization. For estimating the gradient and Hessian in a distributed manner with limited communication, we use a quantization procedure developed in (Davies et al., 2021). The original quantization scheme involves randomness, but we use a *deterministic* version of it, by picking up the closest point to the vector that we want to encode. This is similar to the quantization scheme used by (Alistarh & Korhonen, 2020) for standard gradient descent, and has the following properties:

Proposition 2. *(Davies et al., 2021; Alistarh & Korhonen, 2020) Denoting by b the number of bits that each machine uses to communicate, there exists a quantization function*

$$Q : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}^d,$$

which, for each $\epsilon, y > 0$, consists of an encoding function $\mathrm{enc}_{\epsilon,y} : \mathbb{R}^d \to \{0,1\}^b$ *and a decoding one* $\mathrm{dec}_{\epsilon,y} : \{0,1\}^b \times \mathbb{R}^d \to \mathbb{R}^d$, *such that, for all $x, x' \in \mathbb{R}^d$,*

- $\mathrm{dec}_{\epsilon,y}(\mathrm{enc}_{\epsilon,y}(x), x') = Q(x, x', y, \epsilon)$, *if* $\|x - x'\| \leq y$.
- $\|Q(x, x', y, \epsilon) - x\| \leq \epsilon$, *if* $\|x - x'\| \leq y$.
- *If $y/\epsilon > 1$, the cost of the quantization procedure in number of bits satisfies $b = \mathcal{O}(d \log_2 \left(\frac{y}{\epsilon}\right))$.*

3. Quantized Preconditioned Gradient Descent for GLMs

As a warm-up, we consider the case of a Generalized Linear Model (GLM) with data matrix $A \in \mathbb{R}^{m \times d}$. GLMs are particularly attractive models to distribute, because the distribution across nodes can be performed naturally by partitioning the available data. For more background on distributing GLMs see (Mendler-Dünner & Lucchi, 2020).

The matrix A consists of the data used for training in its rows, i.e. we have m-many d-dimensional data points. As is custom in regression analysis, we assume that $m \gg d$, i.e. we are in the case of big but low-dimensional data. If m is very large, it can be very difficult to store the whole matrix A in one node, so we distribute it in n-many nodes, each one owning m_i-many data points ($m = \sum_{i=1}^n m_i$). We pack the data owned by node i in a matrix $A_i \in \mathbb{R}^{m_i \times d}$ and denote the function used to measure the error on machine i by $\ell_i : \mathbb{R}^{m_i} \to \mathbb{R}$. Then the local cost function $f_i : \mathbb{R}^d \to \mathbb{R}$ at machine i reads

$$f_i(x) = \ell_i(A_i x).$$

We can express the global cost function f in the form

$$f(x) = \ell(Ax)$$

where $\ell : \mathbb{R}^m \to \mathbb{R}$ is a global loss function defined by

$$\ell(y) = \frac{1}{n} \sum_{i=1}^n \ell_i(y_i),$$

where y_i are sets of m_i-many coordinates of y obtained by the same data partitioning.

Assumption 3. *The local loss functions ℓ_i are μ_ℓ-strongly convex and γ_ℓ-smooth.*

This assumption implies that the global loss function ℓ is $\frac{\mu_\ell}{n}$-strongly convex and $\frac{\gamma_\ell}{n}$-smooth. This is because the Hessian of ℓ has the block-diagonal structure

$$\nabla_y^2 \ell(y) = \frac{1}{n} \mathrm{diag}\left(\nabla_{y_1}^2 \ell_1(y_1), ..., \nabla_{y_n}^2 \ell_n(y_n)\right)$$

and the eigenvalues of all matrices $\nabla_{y_i}^2 \ell_i(y_i)$ are between μ_ℓ and γ_ℓ. The Hessian of f can be written as

$$\nabla^2 f(x) = A^T \nabla^2 \ell(Ax) A \in \mathbb{S}(d) \subseteq \mathbb{R}^{d \times d}.$$

We detail the computation of $\nabla^2 f$ in Appendix B.

Assumption 4. *The matrix $A \in \mathbb{R}^{m \times d}$ is of full rank (i.e. $rank(A) = d$, since $d < m$).*

This assumption is natural: if two columns of the matrix A were linearly dependent, we would not need both the related features in our statistical model. Practically, we can prune one of them and get a new data matrix of full-rank.

Proposition 5. *The maximum eigenvalue λ_{max} of $\nabla^2 f$ satisfies*

$$\gamma := \lambda_{max}(\nabla^2 f) \leq \gamma_\ell \lambda_{max}\left(\frac{A^T A}{n}\right)$$

and the minimum eigenvalue λ_{min} of $\nabla^2 f$ satisfies

$$\mu := \lambda_{min}(\nabla^2 f) \geq \mu_\ell \lambda_{min}\left(\frac{A^T A}{n}\right).$$

The proof is presented in Appendix B. Thus, we have that the condition number κ of our minimization problem satisfies

$$\kappa \leq \kappa_\ell \kappa\left(\frac{A^T A}{n}\right),$$

where $\kappa\left(\frac{A^T A}{n}\right)$ is the condition number of the covariance matrix $A^T A$ averaged in the number of machines. The convergence rate of gradient descent generally depends on κ, which can be much larger than κ_ℓ in case that the condition number of $A^T A$ is large. The usual way to get rid of $\kappa\left(\frac{A^T A}{n}\right)$ is to precondition gradient descent using $\frac{A^T A}{n}$, which we denote by M from now on (we recall the convergence analysis of this method in Appendix C). In our setting M is not known to all machines simultaneously, since each machine owns only a part of the overall data. However, we observe that

$$M = \frac{1}{n} \sum_{i=1}^n A_i^T A_i,$$

where $A_i^T A_i =: M_i$ is the local covariance matrix of the data owned by the node i.

3.1. The Algorithm

In this section we present our QPGD-GLM algorithm and study its communication complexity. We structure the algorithm in four steps: first, we describe how to recover a quantized version of the averaged covariance matrices. Then, we describe how nodes perform initialization. Next, we describe how nodes can quantize the initial descent direction. Finally, we describe how to quantize the descent directions for subsequent steps. Our notation for quantization operations follows Section 2.

---

1. Choose an arbitrary master node, say i_0.

(A) Averaged Covariance Matrix Quantization:

2. Compute $M_i := A_i^T A_i$ in each node.
3. Encode M_i in each node i and decode it in the master node using its information:
$\bar{M}_i = \phi^{-1}\left(Q\left(\phi(M_i), \phi(M_{i_0}), 2\sqrt{d}n\lambda_{max}(M), \frac{\lambda_{min}(M)}{16\sqrt{2}\kappa_\ell}\right)\right).$

   In detail, we first transform the local matrix M_i via the isomorphism ϕ, and then quantize it via Q, with carefully-set parameters. The matrix will be then de-quantized relative to the master's reference point $\phi(M_{i_0})$, and then re-constituted (in approximate form) via the inverse isomorphism.

4. Average the decoded matrices in the master node:
$S = \frac{1}{n}\sum_{i=1}^n \bar{M}_i$.
5. Encode the average in the master node and decode in each node i using its local information
$\bar{M} = \phi^{-1}\left(Q(\phi(S), \phi(M_i), \sqrt{d}\left(\frac{\lambda_{min}(M)}{16\kappa_\ell} + 2n\lambda_{max}(M)\right), \frac{\lambda_{min}(M)}{16\sqrt{2}\kappa_\ell})\right).$

(B) Starting Point and Parameters for Descent Direction Quantization:

6. Choose $D > 0$ and $x^{(0)} \in \mathbb{R}^d$, such that
$$\max_i\{\|x^{(0)} - x^*\|, \|x^{(0)} - x_i^*\|\} \leq D.$$
7. Define the parameters
$$\xi := 1 - \frac{1}{2\kappa_\ell}, K := \frac{2}{\xi}, \delta := \frac{\xi(1-\xi)}{4},$$
$$R^{(t)} := \frac{\gamma_\ell}{2}K\left(1 - \frac{1}{4\kappa_\ell}\right)^t D.$$

(C) Quantizing the Initial Descent Direction:

8. Compute $\bar{M}^{-1}\nabla f_i(x^{(0)})$ in each node.
9. Encode $\bar{M}^{-1}\nabla f_i(x^{(0)})$ in each node and decode it in the master node using its local information:
$v_i^{(0)} = Q\left(\bar{M}^{-1}\nabla f_i(x^{(0)}), \bar{M}^{-1}\nabla f_{i_0}(x^0), 4n\kappa(M)R^{(0)}, \frac{\delta R^{(0)}}{2}\right).$
10. Average the quantized local information in the master node:
$r^{(0)} = \frac{1}{n}\sum_{i=1}^n v_i^{(0)}$.
11. Encode $r^{(0)}$ in the master node and decode it in each machine i using its local information:
$v^{(0)} = Q\left(r^{(0)}, \bar{M}^{-1}\nabla f_i(x^{(0)}), \left(\frac{\delta}{2} + 4n\kappa(M)\right)R^{(0)}, \frac{\delta R^{(0)}}{2}\right).$

For $t \geq 0$:

12. Compute
$$x^{(t+1)} = x^{(t)} - \eta v^{(t)}$$
for $\eta > 0$.

(D) Descent Direction Quantization for Next Steps:

13. Encode $\bar{M}^{-1}\nabla f_i(x^{(t)})$ in each node i and decode in the master node using the previous local estimate:
$v_i^{(t+1)} = Q\left(\bar{M}^{-1}\nabla f_i(x^{(t+1)})\right), v_i^{(t)}, 4n\kappa(M)R^{(t+1)}, \frac{\delta R^{(t+1)}}{2}).$
14. Average the quantized local information:
$r^{(t+1)} = \frac{1}{n}\sum_{i=1}^n v_i^{(t+1)}$.
15. Encode $r^{(t+1)}$ in the master node and decode it in each node using the previous global estimate:
$v^{(t+1)} = Q\left(r^{(t+1)}, v^{(t)}, \left(\frac{\delta}{2} + 4n\kappa(M)\right)R^{(t+1)}, \frac{\delta R^{(t+1)}}{2}\right).$

We now discuss the algorithm's assumptions. First, we assume that an over-approximation D for the distance of the initialization from the minimizer is known. This is practical, especially in the case of GLMs: since the loss functions ℓ_i are often quadratics, we can use strong convexity and write

$$\|x^{(0)} - x^*\|^2 \leq \frac{2}{\mu}(f(x^{(0)}) - f^*) \leq \frac{2}{\mu}f(x^{(0)}) =: D^2.$$

and similarly for $\|x^{(0)} - x_i^*\|^2$. Further, following Magnússon et al. (2020) (Assumption 2, page 5), the value $f(x^{(0)})$ is often available, for example in the case of logistic regression. Of course, if we are restricted in a compact domain as is the case of (Tsitsiklis & Luo, 1986) and (Alistarh & Korhonen, 2020), then the domain itself provides an over approximation for all the distances inside it.

The parameters $\lambda_{max}(M), \lambda_{min}(M)$ used for quantization of the matrix M are usually assumed to be known. Specifically, it is common in distributed optimization to assume that all nodes know estimates of the smoothness and strong convexity constants of each of the local cost functions (Tsitsiklis & Luo, 1986). In our case this would imply knowing all $\lambda_{max}(M_i), \lambda_{min}(M_i)$. However, we assume knowledge of just $\lambda_{max}(M)$ and $\lambda_{min}(M)$. This also explains the appearance of the extra $\log n$ factor in our GLM bounds, relative to those for Newton's method.

The convergence and communication complexity of our algorithm are described in the following theorem:

Theorem 6. *The iterates $x^{(t)}$ produced by the previous algorithm with $\eta = \frac{2}{\mu_\ell + \gamma_\ell}$ satisfy*

$$\|x^{(t)} - x^*\| \leq \left(1 - \frac{1}{4\kappa_\ell}\right)^t D$$

and the total number of bits used for communication until $f(x^{(t)}) - f^* \leq \epsilon$ is

$$\mathcal{O}\left(nd^2 \log\left(\sqrt{d} n \kappa_\ell \kappa(M)\right)\right) + \mathcal{O}\left(nd\kappa_\ell \log(n\kappa_\ell\kappa(M)) \log \frac{\gamma D^2}{\epsilon}\right). \tag{1}$$

When the accuracy ϵ is sufficiently small (which is often the case in practice), the first summand is negligible and the total number of bits until reaching it is just

$$b = \mathcal{O}\left(nd\kappa_\ell \log(n\kappa_\ell\kappa(M)) \log \frac{\gamma D^2}{\epsilon}\right)$$

which gains over quantized gradient descent in (Alistarh & Korhonen, 2020) the linear dependence on the condition number of M. We prove Theorem 6 in Appendix D.

4. Quantized Newton's method

After warming-up with quantizing fixed preconditioners in the case of Generalized Linear Models, we move forward to quantize non-fixed ones. The extreme case of a preconditioner is the whole Hessian matrix; preconditioning with it yields Newton's method, which is computationally expensive, but removes completely the dependency on the condition number from the iteration complexity. We develop a quantized version of Newton's method in order to address a question raised by (Alistarh & Korhonen, 2020) regarding whether the communication complexity of minimizing a sum of smooth and strongly convex functions depends linearly on the condition number of the problem. The main technical challenge towards that, is keeping track of the concentration of the Hessians around the Hessian evaluated at the optimum, while the algorithm converges. We show that the linear dependence of the communication cost on the condition number of the problem is not necessary, in exchange with extra dependence on the dimension of the problem, i.e. d^2 instead of d. This can give significant advantage for low-dimensional and ill-conditioned problems (training generalized linear models is among them).

As it is natural for Newton's method, we make the following assumptions for the objective function f:

Assumption 7. *The functions f_i are all γ-smooth and μ-strongly convex with a σ-Lipschitz Hessian, $\gamma, \mu, \sigma > 0$.*

We note that the lower bound derived by Alistarh & Korhonen (2020) is obtained for the case that f_i are quadratic functions; quadratic functions indeed satisfy Assumption 7.

As in the case of GLMs, we define the condition number of the problem to be

$$\kappa := \frac{\gamma}{\mu}.$$

We also introduce a constant $\alpha \in [0, 1)$, to be specified later, which will control the convergence of the algorithm.

4.1. Algorithm Description

We now describe our quantized Newton's algorithm. Again, we split the presentation into several parts: local initialization (A), estimating the initial Hessian modulo quantization (B), as well as the quantized initial descent direction (C), and finally, quantization and update for each iteration (D,E).

---

1. Choose the master node at random, e.g. i_0.

(A) Starting Point and Parameters for Hessian Quantization:

2. Choose $x^{(0)} \in \mathbb{R}^d$, such that
$$\max_i\{\|x^{(0)} - x^*\|, \|x^{(0)} - x_i^*\|\} \leq \frac{\alpha\mu}{2\sigma}.$$
3. We define the parameter
$$G^{(t)} = \frac{\mu}{4}\alpha\left(\frac{1+\alpha}{2}\right)^t.$$

(B) Initial Hessian Quantized Estimation:

4. Compute $\nabla^2 f_i(x^{(0)})$ in each node.
5. Encode $\nabla^2 f_i(x^{(0)})$ in each node i and decode it in the master node i_0 using its information:
$H_0^i = \phi^{-1}\left(Q\left(\phi(\nabla^2 f_i(x^{(0)})), \phi(\nabla^2 f_{i_0}(x^{(0)})), 2\sqrt{d}\gamma, \frac{G^{(0)}}{2\sqrt{2}\kappa}\right)\right).$
6. Average the decoded matrices in the master node: $S_0 = \frac{1}{n}\sum_{i=1}^n H_0^i$.
7. Encode the average in the master node and decode in each node i using its local information
$H_0 = \phi^{-1}\left(Q\left(\phi(S_0), \phi(\nabla^2 f_i(x^{(0)})), \sqrt{d}\left(\frac{G^{(0)}}{2\kappa} + 2\gamma\right), \frac{G^{(0)}}{2\sqrt{2}\kappa}\right)\right).$

Parameters for Descent Direction Quantzation:

8. Define the parameters
$$\theta := \frac{\alpha(1-\alpha)}{4}, K := \frac{2}{\alpha}, P^{(t)} := \frac{\mu}{2\sigma}K\alpha\left(\frac{1+\alpha}{2}\right)^t.$$

(C) Initial Descent Direction Quantized Estimation:

9. Compute $H_0^{-1}\nabla f_i(x^{(0)})$ in each node.
10. Encode $H_0^{-1}\nabla f_i(x^{(0)})$ in each node and decode it in the master node using its local information:
$v_i^{(0)} = Q\left(H_0^{-1}\nabla f_i(x^{(0)}), H_0^{-1}\nabla f_{i_0}(x^{(0)}), 4\kappa P^{(0)}, \frac{\theta P^{(0)}}{2}\right).$
11. Average the quantized local information: $p^{(0)} = \frac{1}{n}\sum_{i=1}^n v_i^{(0)}$.

12. Encode $p^{(0)}$ in the master node and decode it in each machine i using its local information:
$$v^{(0)} = Q\left(p^{(0)}, H_0^{-1}\nabla f_i(x^{(0)}), \left(\tfrac{\theta}{2} + 4\kappa\right) P^{(0)}, \tfrac{\theta P^{(0)}}{2}\right).$$

For $t \geq 0$:

13. Compute
$$x^{(t+1)} = x^{(t)} - v^{(t)}.$$

(D) Hessian Quantized Estimation for Next Steps:

14. Compute $\nabla^2 f_i(x^{(t+1)})$ in each node.
15. Encode $\nabla^2 f_i(x^{(t+1)})$ in each node i and decode in the master node using the previous local estimate:
$$H_{t+1}^i = \phi^{-1}\left(Q\left(\phi(\nabla^2 f_i(x^{(t+1)})), \phi(H_t^i), \tfrac{10\sqrt{d}}{1+\alpha}G^{(t+1)}, \tfrac{G^{(t+1)}}{2\sqrt{2}\kappa}\right)\right).$$
16. Average the quantized local Hessian information: $S_{t+1} = \frac{1}{n}\sum_{i=1}^n H_{t+1}^i$.
17. Encode S_{t+1} in the master node and decode it back in each node using the previous global estimate:
$$H_{t+1} = \phi^{-1}\left(Q\left(\phi(S_{t+1}), \phi(H_t), \sqrt{d}\left(\tfrac{1}{2\kappa} + \tfrac{10}{1+\alpha}\right) G^{(t+1)}, \tfrac{G^{(t+1)}}{2\sqrt{2}\kappa}\right)\right).$$

(E) Descent Direction Quantized Estimation:

18. Compute $H_{t+1}^{-1}\nabla f_i(x^{(t+1)})$ in each node.
19. Encode $H_{t+1}^{-1}\nabla f_i(x^{(t+1)})$ in each node i and decode in the master node using the previous local estimate:
$$v_i^{(t+1)} = Q\left(H_{t+1}^{-1}\nabla f_i(x^{(t+1)}), v_i^{(t)}, 11\kappa P^{(t+1)}, \tfrac{\theta P^{(t+1)}}{2}\right).$$
20. Average the quantized local Hessian information: $p^{(t+1)} = \frac{1}{n}\sum_{i=1}^n v_i^{(t+1)}$.
21. Encode S_{t+1} in the master node and decode it back in each node using the previous global estimate:
$$v^{(t+1)} = Q\left(p^{(t+1)}, v^{(t)}, \left(\tfrac{\theta}{2} + 11\kappa\right) P^{(t+1)}, \tfrac{\theta P^{(t+1)}}{2}\right).$$

The restriction of the initialization $x^{(0)}$ is standard for Newton's method, which is known to converge only *locally*. Usually $x^{(0)}$ is chosen such that $\alpha \geq \frac{\sigma}{\mu}\|x^{(0)} - x^*\|$, while we choose it such that $\alpha \geq 2\frac{\sigma}{\mu}\|x^{(0)} - x^*\|$ (and the same for x_i^* in the place of x^*). This difference occurs from the extra errors due to quantization. This assumption implies also that the minima of the local costs cannot be too far away from each other.

We now state our theorem on communication complexity of quantized Newton's algorithm, which is the main result of the paper. The proof is in Appendix E, and relies on analyzing the behaviour of both the quantized Hessian estimates and the quantized descent direction estimates simultaneously, as can be seen in Lemma 16.

Theorem 8. *The iterates of the quantized Newton's method starting from a point $x^{(0)}$, such that*
$$\|x^{(0)} - x^*\| \leq \frac{\mu}{4\sigma}\left(\alpha = \frac{1}{2}\right)$$
satisfy
$$\|x^{(t)} - x^*\| \leq \frac{\mu}{4\sigma}\left(\frac{3}{4}\right)^t$$
and the communication cost until reaching accuracy ϵ in terms of function values is
$$\mathcal{O}\left(nd^2\log\left(\sqrt{d}\kappa\right)\log\frac{\gamma\mu^2}{\sigma^2\epsilon}\right) \qquad (2)$$
many bits in total.

We note that the lower bound derived in (Alistarh & Korhonen, 2020) is for the case that all functions f_i are quadratics. For quadratics, the Hessian is constant, thus $\sigma = 0$ and α can be chosen equal to 0 as well. Then, (non-distributed) Newton's method converges in only one step. However, in the distributed case, $\sigma = 0$ implies $G^{(t)} = 0$, thus the estimation of $\nabla^2 f(x^{(t)})$ must be exact. This would mean that we need to use an infinite number of bits, and this can be seen also in our communication complexity results. In order to apply our result in a practical manner, we need to allow the possibility for strictly positive quantization error of the Hessian, thus we must choose $\sigma > 0$.

5. Estimation of the Minimum in the Master

In the previous sections we computed an approximated minimizer of our objective function up to some accuracy and counted the communication cost of the whole process. We now extend our interest to the slightly harder problem of estimating the minimum f^* of the function f (which is again assumed to be γ-smooth and μ-strongly convex) in the master node with accuracy ϵ. This extension is not considered in (Magnússon et al., 2020), but is discussed in (Alistarh & Korhonen, 2020). To that end, we estimate the minimizer x^* of f by a vector $x^{(t)}$, such that $f(x^{(t)}) - f^* \leq \frac{\epsilon}{2}$, and the communication cost of doing that is again given by expression (1) for GLM training and expression (2) for Newton's method.

We denote x_i^* the minimizer of the local cost function f_i and $f_i^* := f_i(x_i^*)$ its minimum. We also assume that we are aware of an over approximation $C > 0$ of the maximum distance of x^* from the minimizers of the local costs x_i^*, i.e. $\max_{i=1,\ldots,n}\|x^* - x_i^*\| \leq C$ and a radius $c > 0$ for the minima of the local costs: $\max_{i=1,\ldots,n} \mid f_i^* \mid \leq c$. Estimating these constants can be feasible in many practical situations:

- We can always bound the quantity $\max_{i=1,\ldots,n}\|x^* - x_i^*\|$ by a known constant if we set our problem in a compact domain as it is the case in (Tsitsiklis & Luo,

1986) and (Alistarh & Korhonen, 2020). Also, if our local data are obtained from the same distribution, then we do not expect the minimizers of the local costs to be too far away from the global minimizer.

- The minima f_i^* of the local costs are often exactly 0 (as assumed in (Alistarh & Korhonen, 2020)). This is because the local cost functions f_i are often quadratics, as it happens in the case of GLMs. In the worst case, knowing just that $f_i \geq 0$, we can write
$$\mid f_i^* \mid = f_i^* \leq f_i(x^{(0)}) \leq nf(x^{(0)})$$
and the value $f(x^{(0)})$ is often available as discussed in Section 3 and in (Magnússon et al., 2020).

For estimating the minimum f^*, we start by computing $f_i(x^{(t)})$ in each node i and communicate them to the master node i_0 as follows:

$$q_i^{(t)} := Q(f_i(x^{(t)}), f_{i_0}(x^{(t)}), 2(\gamma C^2 + c), \epsilon/2).$$

Then the master node computes and outputs the average

$$\bar{f} = \frac{1}{n} \sum_{i=1}^{n} q_i^{(t)}.$$

Proposition 9. *The value $\bar{f}$ which occurs from the previous quantization procedure is an estimate of the true minimum f^* of f with accuracy ϵ and the cost of quantization is*

$$\mathcal{O}\left(n \log \frac{\gamma C^2 + c}{\epsilon}\right).$$

if ϵ is sufficiently small.

The proof is presented in Appendix F.
Thus, for the problem that the master node needs to output estimates for both the minimizer and the minimum with accuracy ϵ in terms of function values, the total communication cost is at most

$$\mathcal{O}\left(nd\kappa_\ell \log(n\kappa_\ell \kappa(M)) \log \frac{\gamma(C^2 + D^2) + c)}{\epsilon}\right)$$

many bits in total for QPGD-GLM

$$\mathcal{O}\left(nd^2 \log\left(\sqrt{d}\kappa\right) \log\left(\left(\gamma\left(\frac{\mu^2}{\sigma^2} + C^2\right) + c\right)\frac{1}{\epsilon}\right)\right).$$

many bits in total for quantized Newton's method when ϵ is sufficiently small.

6. Experiments

6.1. Experiment 1: Least-Squares Regression

We first test our method experimentally to compress a parallel solver for least-squares regression. The setting is as follows: we are given as input a data matrix A, with rows randomly partitioned evenly among the nodes, and a target vector b, with the goal of finding $x^* = \mathrm{argmin}_x \|Ax - b\|_2^2$. Since this loss function $f(x) := \|Ax - b\|_2^2$ is quadratic, its Hessian is constant, and so Newton's method and QPGD-GLM are equivalent: in both cases, we need only to provide the preconditioner matrix $A^T A$ in the first iteration, and machines can henceforth use it for preconditioning in every iteration.

To quantize the preconditioner matrix, we apply the 'practical version' (that is, using the cubic lattice with mod-based coloring) of the quantization method of (Davies et al., 2021), employing the 'error detection' method in order to adaptively choose the number of bits required for the decoding to succeed. Each node i quantizes the matrix $A_i^T A_i$, which is decoded by the master node i_0 using $A_{i_0}^T A_{i_0}$. Node i_0 computes the average, quantizes, and returns the result to the other nodes, who decode using $A_i^T A_i$.

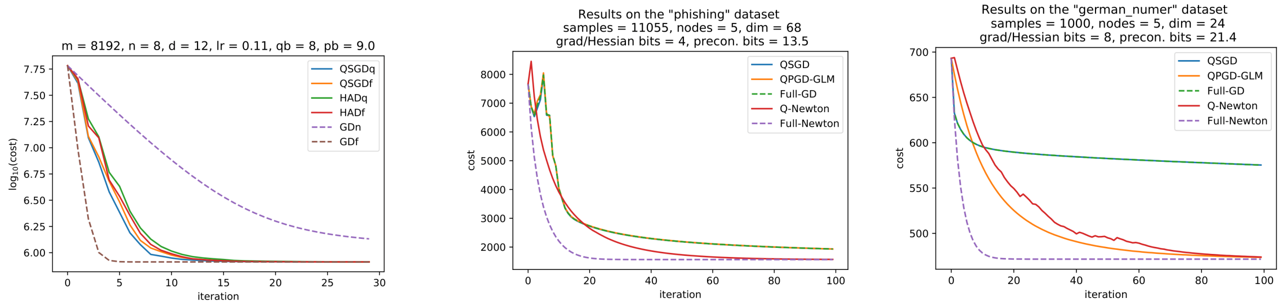
To quantize gradients, we use two leading gradient quantization techniques: QSGD (Alistarh et al., 2016), and the Hadamard-rotation based method of (Suresh et al., 2017), since these are optimized for such an application.[1] In each iteration (other than the first), we quantize the *difference* between the current local gradient and that of last iteration, average these at the master node i_0, and quantize and broadcast the result.

Compared Methods. In Figure 1a we compare the following methods: GDn and GDf are full-precision (i.e., using 32-bit floats) gradient descent using *no preconditioning* and *full-precision preconditioning* respectively, as baselines. QSGDq and QSGDf use QSGD for gradient quantization, and the *quantized* and *full-precision* preconditioner respectively. HADq and HADf are the equivalents using instead the Hadamard-rotation method for gradient quantization. When using a preconditioner, we rescale preconditioned gradients to preserve ℓ_2-norm, so that our comparison is based only on update direction and not step size.

Parameters. In addition to m, n, and d, we also have the following parameters: the learning rate (lr in the figure titles) is set close to the maximum for which gradient descent will converge, since this is the regime in which preconditioning can help. The number of bits per coordinate used to quantize gradients (qb) and preconditioners (pb) are also shown; the latter is an average since the quantization method uses a variable number of bits[2]. The results presented are an average of the cost function per descent iteration, over 10

[1] There is a wide array of other gradient quantization methods; we use these two as a representative examples, since we are mostly concerned with the effects of preconditioner quantization.

[2] These quantization methods (and most others) also require exchange of two full-precision scalars, which are not included in the per-coordinate costs since they are independent of dimension.



(a) Least-squares regression performance on `cpusmall_scale`

(b) Logistic regression performance on `phishing`

(c) Logistic regression performance on `german_numer`

repetitions with different random seeds.

Dataset We use the dataset `cpusmall_scale` from LIBSVM (Chang & Lin, 2011). Here we outperform non-preconditioned gradient descent and approach the performance of full-precision preconditioned gradient descent using significantly reduced communication (Figure 1a).

6.2. Experiment 2: Logistic Regression

In order to compare the performance of Q-Newton and QPGD-GLM, we implement a common application in which the Hessian is *not* constant: logistic regression, for binary classification problems.

QPGD-GLM, QSGD, and full-precision gradient descent are implemented as before; we now add full-precision Newton's method for comparison, and our Q-Newton algorithm. The latter uses the quantization method of (Davies et al., 2021) for the initial Hessian (as for QPGD-GLM), and QSGD for subsequent Hessian updates.

Rather than re-scaling gradients, we take a different approach to choosing a learning rates in order to compare the methods fairly: we test each with learning rates in $\{2^{-0}, 2^{-1}, 2^{-2}, \dots\}$, and plot the highest rate for which the method stably converges. Our results are averaged over five random seeds.

We demonstrate the methods on the `phishing` and `german_numer` datasets from the LIBSVM collection (Chang & Lin, 2011), in Figures 1b and 1c respectively. The former demonstrates that Q-Newton improves over (even full precision) first-order methods, while quantizing Hessians at only 4 bits per coordinate. The latter demonstrates an instance in which QPGD-GLM is even faster, since it remains stable under a higher learning rate.

7. Discussion

We proposed communication-efficient versions for two fundamental optimization algorithms, and analyzed their convergence and communication complexity. Our work shows that quantizing second-order information can i) theoretically yield to communication complexity upper bounds with sub-linear dependence on the condition number of the problem, and ii) empirically achieve superior performance over vanilla methods.

There are intriguing questions for future work:
The $\log \kappa$-dependency for Newton's method occurs because of our bounds for the input and output variance of quantization. It would be interesting to see whether this dependency can be avoided, making the bounds completely independent of the condition number.

Another interesting question is whether the $\log d$-dependency can be circumvented. $\log d$ is obtained directly from the use of the vectorization ϕ and could be avoided by quantization using lattices with good spectral norm properties. We are however unaware of such lattice constructions.

One key issue left is the d^2-dependence for the generalized Newton's method, which is due to quantization of d^2-dimensional preconditioners. It would be interesting to determine if linear communication per round can be achieved in the general setting we consider here.

Finally, we would like to point out that there exist more second order methods with superior guarantees compared to vanilla Newton, such as cubic regularization (Nesterov & Polyak, 2006). A very interesting direction for future work would be to investigate whether it is possible to run these algorithms in a distributed setting with limited communication by adding quantization.

Acknowledgements The authors would like to thank Janne Korhonen, Aurelien Lucchi, Celestine Mendler-Dünner and Antonio Orvieto for helpful discussions. FA and DA were supported during this work by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 805223 ScaleML). PD was supported by the European Union's Horizon 2020 programme under the Marie Skłodowska-Curie grant agreement No. 754411.

References

Alistarh, D. and Korhonen, J. H. Improved communication lower bounds for distributed optimisation. *arXiv preprint arXiv:2010.08222*, 2020.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *arXiv preprint arXiv:1610.02132*, 2016.

Alistarh, D., Hoefler, T., Johansson, M., Khirirat, S., Konstantinov, N., and Renggli, C. The convergence of sparsified gradient methods. *arXiv preprint arXiv:1809.10505*, 2018.

Arjevani, Y. and Shamir, O. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pp. 1756–1764, 2015.

Ben-Nun, T. and Hoefler, T. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chen, Y. Gradient methods for unconstrained problems. http://www.princeton.edu/~yc5/ele522_optimization/lectures/grad_descent_unconstrained.pdf, 2019. Princeton University, Fall 2019.

Davies, P., Gurunanthan, V., Moshrefi, N., Ashkboos, S., and Alistarh, D. New bounds for distributed mean estimation and variance reduction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=t86MwoUCCNe.

Ghadikolaei, H. S. and Magnússon, S. Communication-efficient variance-reduced stochastic gradient descent. *arXiv preprint arXiv:2003.04686*, 2020.

Hendrikx, H., Xiao, L., Bubeck, S., Bach, F., and Massoulie, L. Statistically preconditioned accelerated gradient method for distributed optimization. In *International Conference on Machine Learning*, pp. 4203–4227. PMLR, 2020.

Islamov, R., Qian, X., and Richtárik, P. Distributed second order methods with fast rates and compressed communication. *arXiv preprint arXiv:2102.07158. Accepted to ICML 2021.*, 2021.

Jaggi, M., Smith, V., Takáč, M., Terhorst, J., Krishnan, S., Hofmann, T., and Jordan, M. I. Communication-efficient distributed dual coordinate ascent. *arXiv preprint arXiv:1409.1458*, 2014.

Jordan, M. I., Lee, J. D., and Yang, Y. Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, 2018.

Khirirat, S., Feyzmahdavian, H. R., and Johansson, M. Distributed learning with compressed gradients. *arXiv preprint arXiv:1806.06573*, 2018.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B.-Y. Scaling distributed machine learning with the parameter server. In *Proc. 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2014)*, pp. 583–598, 2014.

Magnússon, S., Shokri-Ghadikolaei, H., and Li, N. On maintaining linear convergence of distributed learning and optimization under limited communication. *IEEE Transactions on Signal Processing*, 68:6101–6116, 2020.

Mendler-Dünner, C. and Lucchi, A. Randomized block-diagonal preconditioning for parallel learning. In *International Conference on Machine Learning*, pp. 6841–6851. PMLR, 2020.

Nesterov, Y. and Polyak, B. Cubic regularization of newton method and its global performance. LIDAM Reprints CORE 1927, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2006. URL https://EconPapers.repec.org/RePEc:cor:louvrp:1927.

Nguyen, L., Nguyen, P. H., Dijk, M., Richtárik, P., Scheinberg, K., and Takác, M. Sgd and hogwild! convergence without the bounded gradients assumption. In *International Conference on Machine Learning*, pp. 3750–3758. PMLR, 2018.

Niu, F., Recht, B., Ré, C., and Wright, S. J. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *arXiv preprint arXiv:1106.5730*, 2011.

Ramezani-Kebrya, A., Faghri, F., and Roy, D. M. NUQSGD: improved communication efficiency for data-parallel SGD via nonuniform quantization. *CoRR*, abs/1908.06077, 2019. URL http://arxiv.org/abs/1908.06077.

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

Safaryan, M., Islamov, R., Qian, X., and Richtárik, P. Fednl: Making newton-type methods applicable to federated learning, 2021.

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *international conference on machine learning*, pp. 3027–3036. PMLR, 2017.

Shamir, O. Fundamental limits of online and distributed algorithms for statistical learning and estimation. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pp. 163–171, 2014.

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

Suresh, A. T., Yu, F. X., Kumar, S., and McMahan, H. B. Distributed mean estimation with limited communication. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3329–3337, 2017.

Tsitsiklis, J. N. and Luo, Z. Communication complexity of convex optimization. In *1986 25th IEEE Conference on Decision and Control*, pp. 608–611, 1986. doi: 10.1109/CDC.1986.267379.

Vempala, S. S., Wang, R., and Woodruff, D. P. The communication complexity of optimization. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1733–1752. SIAM, 2020.

Wang, S., Roosta, F., Xu, P., and Mahoney, M. W. Giant: Globally improved approximate newton method for distributed optimization. *Advances in Neural Information Processing Systems*, 31:2332–2342, 2018.

Ye, M. and Abbe, E. Communication-computation efficient gradient coding. In *International Conference on Machine Learning*, pp. 5610–5619. PMLR, 2018.

Zhang, J., You, K., and Başar, T. Distributed adaptive newton methods with globally superlinear convergence. *arXiv preprint arXiv:2002.07378*, 2020.

Zhang, Y. and Lin, X. Disco: Distributed optimization for self-concordant empirical loss. In *International conference on machine learning*, pp. 362–370. PMLR, 2015.

Zhang, Y., Duchi, J., Jordan, M. I., and Wainwright, M. J. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pp. 2328–2336, 2013.