Communication Efficient Distributed Approximate Newton Method

Avishek Ghosh*, Raj Kumar Maity†, Arya Mazumdar†, Kannan Ramchandran*
*Department of Electrical Engineering and Computer Sciences, UC Berkeley,
avishek_ghosh@berkeley.edu, kannanr@eecs.berkeley.edu
†College of Information and Computer Sciences, UMass Amherst,
{rajkmaity, arya}@cs.umass.edu.

***Abstract*—In this paper, we develop a communication efficient second order distributed Newton-type algorithm. For communication efficiency, we consider a generic class of δ-approximate compressors (Karimireddy et al., 2019), which includes *sign*-based compression and top-k sparsification. We provide three potential settings where compression can be employed; and provide rate of convergence for smooth objectives. We show that, in the regime where δ is constant, our theoretical convergence rate matches that of a state-of-the-art distributed second order algorithm called *DINGO* (Crane and Roosta, 2019). This implies that we get the compression for free in this regime. *The full paper can be found at* https://tinyurl.com/ujnpt4c.**

I. Introduction

The presence of parallel and distributed computation in machine learning is universal now with the ever-growing size of training data in modern day applications. Availability of training data in a distributed fashion is a commonly used framework for data processing. In applications like *Federated Learning* [1], data is inherently distributed among users' personal devices. In a distributed setup, usually the computation (training, processing etc.) happens locally in the worker machines, and the local results are communicated to a central machine (parameter server). The central machine then aggregates them and updates the model parameters. With more worker machines, the burden of computation is alleviated, but the gain in speed-up often gets bottle-necked by the high communication overhead between the workers and the central machine. In recent years, a significant amount of effort and progress has been made in reducing the communication overhead in first order optimization by sparsification and quantization of gradients [2]–[9].

An alternative way to reduce the number of iterations (and hence the communication cost) between the workers and the central machine is to use second order optimization algorithms; which are known to converge much faster than their first order counterparts. Indeed, a handful of algorithms has been developed using this philosophy, such as DANE [10], DISCO [11], GIANT [12], DINGO [13], Newton-MR [14], INEXACTDANE and AIDE [15]. However, the question of whether it is possible to employ quantization/sparsification techniques in second order algorithms to further cut down communication cost remains unanswered.

In this paper, we answer this aforementioned question affirmatively. To the best of our knowledge, this is the first work to address the problem of high communication overhead in distributed second order optimization. Here, we use a δ-approximate compressor (see Definition 1) to provide communication efficiency in a recently proposed second order optimization algorithm DINGO [13]. We show three different settings where we apply compression to reduce communication overhead in each iteration.

In each iteration of DINGO, the worker machines first communicate the local gradients to the central machine. The central machine aggregates the local gradients and broadcasts the global gradient to the worker machines. The worker machines then compute the local Hessians, obtain the (pseudo) inverse, compute the product of the inverse Hessian and the global gradient and send this vector to the central machine. Observe that, there are multiple places where compression can be employed. Such as:

1) *One round compression:* Every worker machine sends the uncompressed gradients to the central machine. However, while sending the product of the inverse (local) Hessian and gradient, it uses a δ-approximate compressor, and sends the compressed vector. Hence, only one round of compression is employed in each iteration.
2) *Two round compression:* Here every worker compresses their local gradients and sends it to the central machine. Furthermore, while sending the product of local inverse Hessian and gradient, it also compresses that vector. So, each worker machine uses two rounds of compression (both with δ-approximate compressor) in each iteration.
3) *One round communication:* In this setting, the worker machines communicate to the central machine only once in each iteration. Each worker machine computes the local gradient and the (pseudo) inverse of the local Hessian, and thereby obtains their product. Now, using a δ-approximate compressor, it compresses this vector and communicate to the central machine.

We provide a careful analysis of the above settings such that we can track the effect of compression on the convergence rate. For setting 1, the conditions on the algorithm remain almost the same as that of DINGO with availability of the gradient but differs in a 'dot-product' condition [13] as the second round of the communication (Hessian-gradient product) is compressed. For setting 2, we provide analysis with both rounds of communication being compressed while adhering to the same protocol of the underlying second order algorithm.

Setting 3 achieves a very low communication cost which is the primary goal of this paper. We omit the discussion of

setting 3 in this submission, but remark that its theoretical analysis currently works under a strong assumption. Analysis with a weaker condition for one round communication is a challenging problem in setting 3, and can be thought of as an exciting future work.

Summary of Contributions: We propose and analyze a communication-efficient Newton-type algorithm. We use DINGO [13] as the baseline second order algorithm and make it communication-efficient by employing δ-approximate compressors. As mentioned, we handle one-round and two-round compression (settings 1 and 2) both theoretically and experimentally. We show that with proper choice of the step-size and hyper-parameters of the algorithm, we can achieve the same rate of convergence as DINGO. We prove that the gradient norm decreases exponentially over iterations of the algorithm. We also validate our results for regularized logistic regression for binary classification on real datasets [16]. Also, we emphasize here that when $\delta = 1$ (no compression), we recover the same convergence rate of DINGO [13]. Furthermore, in the regime where the compression factor δ is constant ($\Theta(1)$), with a careful choice of learning rate, our rate of convergence matches (order-wise) to that of DINGO. So we get compression for *free* in this parameter regime. Note that, as illustrated in [17], $\delta = \Theta(1)$ is usually observed in most practical applications. We have omitted proofs of theorems in this submission, which can be found in the full version.

A. Related Work

Distributed Second Order Optimization: In the past few years, several distributed second order algorithms such as DANE [10], INEXACTDANE and AIDE [15], DISCO [11] and GIANT [12] have been proposed and analyzed. These algorithms requires convexity of the objective function (in addition to 2nd order oracle access. Very recently, [14] and [13] alleviate these disadvantages. In [18], a numerical linear algebra based sketching method has been developed to compute the approximate Hessian. In this work, we deal with non-convex objective and emphasize communication efficiency.

Gradient Compression: In the works [2]–[4], [7], [8], [19], communication efficiency is achieved by coordinate-wise quantization of gradients and in [9], vector quantization of gradients has been studied. Recently, gradient sparsification where only *top-k* component of the d-dimensional gradient vector is communicated, have been proposed in [5], [6], [20], [21]. In [17], the authors exploit the 'error in compression' as feedback to improve convergence of first order optimization. Very recently, Byzantine resilient communication efficient method have been analyzed in [22], [23]. Note that all the compression techniques discussed here are only applicable for first order optimization. To the best of our knowledge, this is the first work to propose and analyze compression schemes for second order optimization.

II. Background and Problem Statement

Our objective is to solve the following problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{w}), \tag{1}$$

in a distributed environment with m worker machines, where each machine has local access to the ith loss function f_i. We assume that the worker machines can communicate to the central machine, but can not interact among themselves. This is a commonly used distributed setup particularly in applications like Federated Learning, large scale neural net training etc. We assume that ith worker machine has n i.i.d data points $\{x_{i,j}\}_{j=1}^{n}$, and hence $f_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} l(\mathbf{w}; x_{i,j})$, where $l(\mathbf{w}; x_{i,j})$ is the loss associated with jth data point $x_{i,j}$. Classically, we use the second order Newton method for convex optimization, and the update is given by,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{p}_t \text{ where } \mathbf{p}_t = -[\nabla^2 f(\mathbf{w}_t)]^{-1}\nabla f(\mathbf{w}_t) \tag{2}$$

where α_t is the step size. Here we provide a communication efficient Newton type algorithm in distributed setup. We use the recently proposed and popular second order distributed optimization algorithm called DINGO [13]. Here we address the setting 1 and 2 as presented in Section I. Next, we define δ-approximate compressor (as in [17]).

Definition 1 (δ-approximate compressor)**.** *An operator $\mathcal{Q} : \mathbb{R}^d \to \mathbb{R}^d$ is called an δ-approximate compressor on a set $S \subset \mathbb{R}^d$ if $\|\mathcal{Q}(x) - x\|^2 \leq (1-\delta)\|x\|^2$ for all $x \in S$ and $\delta \in [0,1]$ is the compression factor. Furthermore, a randomized operator $\mathcal{Q}$ is δ-approximate compressor if $\mathbb{E}\|\mathcal{Q}(x) - x\|^2 \leq (1-\delta)\|x\|^2$ for all $x \in S$.*

We list a few example of δ-approximate compressor:

1) quantized SGD with ℓ_1 norm [17], $\mathcal{Q}(x) = \frac{\|x\|_1}{d}\text{sign}(x)$, which is a $\frac{\|x\|_1^2}{d\|x\|^2}$-approximate compressor. Here, $\text{sign}(x) \in \{0, \pm 1\}^d$ is the coordinate-wise signs of the vector x. We choose this compressor for the experiments.
2) top$_k$ operator, which selects k coordinates with largest absolute value; for $1 \leq k \leq d$, $(\mathcal{Q}(x))_i = (x)_{\pi(i)}$ if $i \leq k$, and 0 otherwise, where π is a permutation of $[d]$ with $(|x|)_{\pi(i)} \geq (|x|)_{\pi(i+1)}$ for $i \in [d-1]$. This is a k/d-approximate compressor.
3) k-PCA that uses top k eigenvectors to approximate a matrix X [3].

Notation: Throughout the paper, we use bold upper case $\mathbf{H}$ for matrices and lowercase $\mathbf{w}$ for vector. By $\langle . \rangle$, we denote Euclidean dot-product. For a vector and a matrix $\|.\|$ will denote the l_2 and spectral norm respectively. The *Moore-Penrose Inverse* of any matrix $\mathbf{H}$ is denoted by $\mathbf{H}^\dagger$. For vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ by $[\mathbf{x}, \mathbf{y}]$ we denote the line $\{(1-t)\mathbf{x} + t\mathbf{y} | 0 \leq t \leq 1\}$. Also for the purpose of our algorithm we use the following definitions:

$$\begin{aligned} \mathbf{g}_{i,t} &\equiv \nabla f_i(\mathbf{w}_t) \quad \mathbf{H}_{i,t} \equiv \nabla^2 f_i(\mathbf{w}_t) \\ \mathbf{g}_t &\equiv \nabla f(\mathbf{w}_t) \quad \mathbf{H}_t \equiv \nabla^2 f(\mathbf{w}_t), \\ \tilde{\mathbf{H}}_{i,t} \equiv \begin{bmatrix} \mathbf{H}_{i,t} \\ \phi \mathbf{I} \end{bmatrix} &\in \mathbb{R}^{2d\times d} \quad \tilde{\mathbf{g}}_{i,t} \equiv \begin{bmatrix} \mathbf{g}_{i,t} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2d}, \end{aligned} \tag{3}$$

where $\phi > 0$ and $\mathbf{0} \in \mathbb{R}^d$ is the all zero vector.

We conclude this section with the set of assumptions required for the analysis presented in the subsequent sections.

Assumption 1 (Twice Differentiablity)**.** *For all $i \in [m]$, the local loss functions f_i are twice differentiable.*

Assumption 2 (Moral Smoothness)**.** *For all iteration t, there exists a constant $L > 0$ such that, for all $\mathbf{w} \in [\mathbf{w}_t, \mathbf{w}_t + \mathbf{p}_t]$, where $\mathbf{p}_t$ is the update direction we have*

$$\|\nabla^2 f(\mathbf{w})\nabla f(\mathbf{w}) - \nabla^2 f(\mathbf{w}_t)\nabla f(\mathbf{w}_t)\| \leq L\|\mathbf{w} - \mathbf{w}_t\|.$$

Note that this is a weaker assumption than the Lipschitzness of both gradient and Hessian, typically used in related literature [13], [14]. The assumption requires the gradient and Hessian to be Lipschitz continuous on the piece-wise linear path of the update direction. In [13], [14], more detailed discussion on this can be found. As shown in [13], we have the following useful lemma.

Lemma 1. *Let $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d, \beta \in (0, \infty), L \in [0, \infty)$ and $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and suppose $\mathbf{y} \in [\mathbf{x}, \mathbf{z}]$. If $\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\| \leq L\|\mathbf{y} - \mathbf{x}\|^\beta$, then,*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \mathbf{y} - \mathbf{x}, \nabla f(\mathbf{x})\rangle + \frac{L}{1+\beta}\|\mathbf{y} - \mathbf{x}\|^{1+\beta}.$$

We use the fact $\nabla(\frac{1}{2}\|\nabla f(\mathbf{w})\|^2) = \nabla^2 f(\mathbf{w})\nabla f(\mathbf{w})$, and Assumptions 1, 2 in Lemma 1 with $\beta = 1$

$$\|\nabla f(\mathbf{w})\|^2 \leq \|\nabla f(\mathbf{w}_t)\|^2 + \langle \mathbf{w} - \mathbf{w}_t, \nabla^2 f(\mathbf{w}_t)\nabla f(\mathbf{w}_t)\rangle + L\|\mathbf{w} - \mathbf{w}_t\|^2, \tag{4}$$

for all $\mathbf{w} \in [\mathbf{w}_t, \mathbf{w}_t + \mathbf{p}_t]$ and all iteration t.

Assumption 3. *For all $i \in [m]$ there exists constants $\gamma_i \in (0, \infty)$ such that $\|\mathbf{H}_{i,t}^\dagger\| \leq \gamma_i$.*

Assumption 4. *For all $i \in [m]$ there exists constants $\tau_i \in (0, \infty)$ such that $\|\mathbf{H}_{i,t}\| \leq \tau_i$.*

Assumptions 3,4 characterize the spectrum of the Hessian and its pseudo-inverse. Assumption 4 implies that each local Hessian has largest singular value uniformly bounded for all iterates. Also, Assumption 3 deals with the smallest singular value of the Hessian. If the function is strongly convex, then the smallest singular value of the Hessian is always positive. But in more general sense, the Hessian is always positive definite in its own range space [14]. In the following assumption, we state the more general form of the Assumption 3.

Assumption 5. *There exists a constant $\beta \in (0, \infty)$ such that for all iteration t we have $\|\mathbf{H}_t\mathbf{p}\| \geq \beta\|\mathbf{p}\|$ where $\mathbf{p}$ is in the range space of $\mathbf{H}_t$ i,e $\mathbf{p} \in \mathcal{R}(\mathbf{H}_t)$.*

Also an assumption similar to below appear in [13].

Assumption 6. *There exists a constant η_i such that $\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t\mathbf{g}_t)\| \geq \eta_i\|\mathbf{g}_t\|$.*

Observe that the assumption is dependent on the compression $\mathcal{Q}$. In Lemma 2, we justify this assumption by providing a proper value to η_i.

The next assumption is basically a restatement of Pythagoras theorem, the proof of which can be found in [14].

Assumption 7. *There exists a constant $\nu \in (0, 1)$ such that*

$$\|(\mathbf{U}_\mathbf{w}^\perp)^T\nabla f(\mathbf{w})\|^2 \leq \frac{1-\nu}{\nu}\|(\mathbf{U}_\mathbf{w}^T\nabla f(\mathbf{w})\|^2,$$

for all $\mathbf{w} \in \mathbb{R}^d$, where $\mathbf{U}$ and $\mathbf{U}^\perp$ are the orthonormal basis for the range space of Hessian and its orthogonal complement.

Using Assumption 7, we infer that $\|\nabla f(\mathbf{w})\|^2 \leq \frac{1}{\nu}\|(\mathbf{U}_\mathbf{w}^T\nabla f(\mathbf{w})\|^2$ for all $\mathbf{w} \in \mathbb{R}^d$.

Lemma 2. *Under the Assumptions 4, 5 and 7, Assumption 6 holds with*

$$\eta_i = \beta(1 - \sqrt{1-\delta})(\frac{\nu}{\tau_i^2 + \phi^2})^{1/2},$$

where ϕ is described in 3 and δ is the compression factor as defined in the definition 1

III. One Round Compression

In this section, we propose and analyze an algorithm for the communication efficient second order optimization. In particular, we obtain a contraction on the gradient norm. It is formally written in Algorithm 1. The algorithm works mainly on the estimation of the gradient in the first round and the estimation of the update direction on the next. In this section, we assume that the worker machines do not compress the local gradients in the first round (setting 1 of Section I), i.e., $\mathcal{Q}_1(x) = x$ for all x, in Algorithm 1. So, the central machine computes the full gradient $\mathbf{g}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{g}_{i,t}$, after receiving the local gradients. Next the central machine broadcasts the gradient $\mathbf{g}_t$ and each worker machine computes the following (compressed) vectors $\mathcal{Q}(\mathbf{H}_{i,t}\mathbf{g}_t), \mathcal{Q}(\mathbf{H}_{i,t}^\dagger\mathbf{g}_t), \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\mathbf{g}_t)$. The central machine computes $\mathcal{Q}(\mathbf{H}_t\mathbf{g}_t) = \frac{1}{m}\sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}\mathbf{g}_t)$. Update direction $\mathbf{p}_t$ and step-size α are computed based on these vectors. The iterated update $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\mathbf{p}_t$ is then performed based on the three cases (Algorithm 1). The constant $\tilde{G}$ in the algorithm is $\theta\|\mathbf{g}_t\|^2$. We now give results on the three cases of the algorithm. For shorthand, we define $\gamma = \frac{1}{m}\sum_{i=1}^m \gamma_i$ and $\tau = \frac{1}{m}\sum_{i=1}^m \tau_i$, which are used in the results of this and subsequent sections.

Case 1: If

$$\langle \frac{1}{m}\sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger\mathbf{g}_t), \mathcal{Q}(\mathbf{H}_t\mathbf{g}_t)\rangle \geq \theta\|\mathbf{g}_t\|^2 \tag{5}$$

then the update is $\mathbf{p}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{p}_{i,t}$ where $\mathbf{p}_{i,t} = -\mathcal{Q}(\mathbf{H}_{i,t}^\dagger\mathbf{g}_t)$ and set $\alpha \leq \frac{1}{L\gamma(1+\sqrt{1-\delta})^2}[\frac{\theta(1-\rho)}{\gamma} - (1 - \delta + \sqrt{1-\delta})\tau]$. We provide the following exponential rate of convergence to a critical point of the loss function $f(.)$.

Theorem 1. *Under the Assumptions 1,2,3 and 4, if we run Algorithm 1 we have*

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta)\|\mathbf{g}_t\|^2. \tag{6}$$

Remark 1. *Note that we achieve an exponential convergence even for non-convex functions. When δ is a constant, our algorithm enjoys the same order of convergence with compression as in [17]. So, we get the compression for* free*.*

Remark 2. *Case 1 is often satisfied if we choose the value of the hyper-parameter $\theta \sim \gamma\tau$ and a constant δ. Experimentally we find that it to be the most frequent case.*

Case 2: If

$$\langle \frac{1}{m}\sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\tilde{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t\mathbf{g}_t)\rangle \geq \theta\|\mathbf{g}_t\|^2 \tag{7}$$

then the update is $\mathbf{p}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{p}_{i,t}$ where $\mathbf{p}_{i,t} = -\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t)$ and set $\alpha \leq \frac{2\phi^2}{L((1+\sqrt{1-\delta})^2}[\theta(1-\rho) - (1-\delta+\sqrt{1-\delta})\frac{\tau}{\phi}]$. Here, (from the construction) we assume

$$\|\tilde{\mathbf{H}}_{i,t}^\dagger\| \leq 1/\phi. \tag{8}$$

Theorem 2. *Under the Assumptions 1,2, 4 and condition* (8)*, if we run Algorithm 1 we have*

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta)\|\mathbf{g}_t\|^2. \tag{9}$$

Remark 3. *We resort to case 2 if the condition for case 1 is not satisfied. We observe the similar exponential convergence. The convergence rate here depends on the choice of ϕ which is the spectral upper bound of $\tilde{\mathbf{H}}^\dagger$. In our experiments, we did not encounter this case.*

Case 3: When the conditions for case 1 and case 2 are not satisfied, the central machine broadcasts $\mathcal{Q}(\mathbf{H}_t\mathbf{g}_t)$ and solve the following optimization problem locally:

$$\begin{aligned} &\text{argmin}_{\mathbf{p}} \|\tilde{\mathbf{H}}_{i,t}\mathbf{p} + \tilde{\mathbf{g}}_t\|^2 \\ &\text{such that } \langle \mathbf{p}, \mathcal{Q}(\mathbf{H}_t\mathbf{g}_t)\rangle \leq -\theta\|\mathbf{g}_t\|^2. \end{aligned} \tag{10}$$

Proposition 1. *The solution to the optimization problem* (10) *is*

$$\begin{aligned} &\hat{\mathbf{p}}_{i,t} = -\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t - \lambda_{i,t}(\tilde{\mathbf{H}}_{i,t}^T\tilde{\mathbf{H}}_{i,t})^{-1}\mathcal{Q}(\mathbf{H}_t\mathbf{g}_t) \\ &\textit{where } \lambda_{i,t} = \frac{\theta\|\mathbf{g}_t\|^2 - (\mathcal{Q}(\mathbf{H}_t\mathbf{g}_t))^T\tilde{\mathbf{H}}_{i,t}^\dagger\tilde{\mathbf{g}}_t}{(\mathcal{Q}(\mathbf{H}_t\mathbf{g}_t))^T(\tilde{\mathbf{H}}_{i,t}^T\tilde{\mathbf{H}}_{i,t})^{-1}\mathcal{Q}(\mathbf{H}_t\mathbf{g}_t)}. \end{aligned}$$

In Algorithm 1, we use $\mathbf{p}_{i,t}$ (to get the direction $\mathbf{p}_t$), which is defined as: $\mathbf{p}_{i,t} = \mathcal{Q}(\hat{\mathbf{p}}_{i,t})$ and set $\alpha \leq \frac{2}{L(1+\sqrt{1-\delta})^2c^2}(\theta(1-\rho) - (2\sqrt{1-\delta}c\tau))$, where $c \equiv \frac{1}{m}\sum_{i=1}^m(\frac{1}{\phi}(2+\frac{\theta}{\eta_i}))$.

Theorem 3. *Suppose Assumptions 1,6 and Lemma 2 hold. Then, Algorithm 1 yields*

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta)\|\mathbf{g}_t\|^2. \tag{11}$$

Remark 4. *Note that, although we retain the same exponential rate of convergence, case 3 is not ideal both in terms of convergence rate and communication as it requires one more round of communication between the central and the worker machines. Fortunately, this case occurs rarely in practical situation (as we observe experimentally in Section V).*

So far, we let the worker machines send uncompressed local gradients and only compress the second round. Hence, with one round compression we do not gain savings in communication order-wise. In the subsequent section, we remove this issue by employing compression in both gradient and Hessian based computation.

IV. Two Round Compression

Here, we use compression in both rounds of communication with the central machine. In particular, the worker machines use a δ-approximate compressor to compress the gradient. Hence, each worker machine sends $\mathcal{Q}_1(\mathbf{g}_{i,t})$ to the central machine (Algorithm 1). The central machine then computes $\hat{\mathbf{g}}_t = \frac{1}{m}\sum_{i=1}^m \mathcal{Q}_1(\mathbf{g}_{i,t})$ and broadcasts it. Next, the updates are done similar to the one round compression. Here, for simplicity

Algorithm 1

Input: Initial iterate $\mathbf{w}_0 \in \mathbb{R}^d$, gradient tolerance $\xi > 0$, Maximum iteration T, parameter $\rho \in [0,1]$, parameter $\theta > 0$ and regularization parameter $\phi > 0$ and Compressors $\mathcal{Q}, \mathcal{Q}_1$

for $t = 0, 1, \ldots, T-1$ **do**
  All worker $i \in [m]$ locally compute and compress $\mathcal{Q}_1(\mathbf{g}_{i,t})$ and communicate it to the central server
  Central machine compute full gradient $\hat{\mathbf{g}}_t = \frac{1}{m}\sum_{i=1}^m \mathcal{Q}_1(\mathbf{g}_{i,t})$
  if $\|\hat{\mathbf{g}}_t\| < \xi$ **then**
    return $\mathbf{w}_t$
  else
    The central machine broadcasts $\hat{\mathbf{g}}_t$ and in parallel each worker computes using compression scheme $\mathcal{Q}(\mathbf{H}_{i,t}\hat{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t), \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\hat{\mathbf{g}}_t)$
    Central machine computes $\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t) = \frac{1}{m}\sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}\hat{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t)$, $\frac{1}{m}\sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t)$ and $\frac{1}{m}\sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\tilde{\mathbf{g}}_t)$
    if (Case 1) $\langle \frac{1}{m}\sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t)\rangle \geq \tilde{G}$ **then**
      $\mathbf{p}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{p}_{i,t}$ with $\mathbf{p}_{i,t} = -\mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t)$
    else if (Case 2) $\langle \frac{1}{m}\sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\tilde{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t)\rangle \geq \tilde{G}$ **then**
      $\mathbf{p}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{p}_{i,t}$ with $\mathbf{p}_{i,t} = -\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\tilde{\mathbf{g}}_t)$
    else
      (Case 3) The central machine broadcasts $\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t)$ and all the worker solve in parallel
      $\mathbf{p}_{i,t} = \mathcal{Q}(-\tilde{\mathbf{H}}_{i,t}^\dagger\tilde{\mathbf{g}}_t - \lambda_{i,t}(\tilde{\mathbf{H}}_{i,t}^T\tilde{\mathbf{H}}_{i,t})^{-1}\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t))$
      where $\lambda_{i,t} = \frac{\theta\|\hat{\mathbf{g}}_t\|^2 - (\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t))^T\tilde{\mathbf{H}}_{i,t}^\dagger\tilde{\mathbf{g}}_t}{(\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t))^T(\tilde{\mathbf{H}}_{i,t}^T\tilde{\mathbf{H}}_{i,t})^{-1}\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t)}$
      Compress and send $\mathbf{p}_{i,t}$. The central machine computes $\mathbf{p}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{p}_{i,t}$.
    end if
    The central machine sets the value of α according to text and updates $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\mathbf{p}_t$
  end if
end for

we choose the same compression factor δ for both rounds, i.e., $\mathcal{Q}_1 = \mathcal{Q}$. Also, we choose $\tilde{G} = \theta\frac{1}{m}\sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2$. Similar to Section III, we analyze case by case basis.

Case 1: If

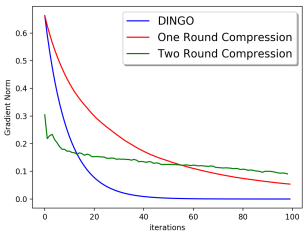
$$\langle \frac{1}{m}\sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t)\rangle \geq \theta\frac{1}{m}\sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2 \tag{12}$$

then the update is $\mathbf{p}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{p}_{i,t}$ where $\mathbf{p}_{i,t} = -\mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t)$ and set $\alpha \leq \frac{2}{\alpha L\gamma^2(1+\sqrt{1-\delta})^2} \times (\theta(1-\rho) - \sqrt{1-\delta}(1+\sqrt{1-\delta})\gamma\tau(\frac{1-\sqrt{2-\delta}}{1-\sqrt{1-\delta}}))$.

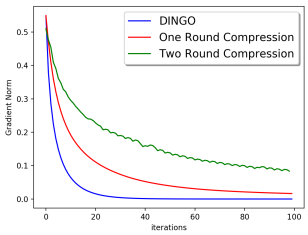
Theorem 4. *Under Assumptions 1,2,3 and 4, if we run Algorithm 1, we obtain*

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta(1-\sqrt{1-\delta})^2)\|\mathbf{g}_t\|^2. \tag{13}$$

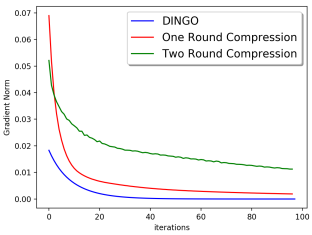
Remark 5. *Compared to the case 1 of one round compression (Theorem 1), the convergence rate here suffers due to the*



((a)) a5a



((b)) w1a



((c)) mushroom

Fig. 1: Comparison of the convergence of DINGO and one and two round compression methods in terms of $\|\mathbf{g}_t\|$ (gradient norm) of regularized logistic regression for binary classification data.

compressed gradient information. But the exponential decay still retains. For compression factor $\delta = \Theta(1)$ (constant), we do not lose order-wise performance compared to DINGO.

Remark 6. *Remarkably, the communication cost here is extremely low as compared the one round compression (as we see in experiments as well). Here both rounds of communication from workers to the central machine are compressed.*

Case 2: If

$$\langle \frac{1}{m}\sum_{i=1}^{m} \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^{\dagger}\tilde{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t)\rangle \geq \theta \frac{1}{m}\sum_{i=1}^{m} \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2 \quad (14)$$

then the update is $\mathbf{p}_t = \frac{1}{m}\sum_{i=1}^{m}\mathbf{p}_{i,t}$ where $\mathbf{p}_{i,t} = -\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^{\dagger}\tilde{\mathbf{g}}_t)$ and set $\alpha \leq \frac{2\phi^2}{\alpha L(1+\sqrt{1-\delta})^2} \times (\theta(1-\rho) - \sqrt{1-\delta}(1+\sqrt{1-\delta})\frac{\tau}{\phi}(\frac{1-\sqrt{2-\delta}}{1-\sqrt{1-\delta}}))$.

Theorem 5. *Under the assumption 1,2, 4 and equation* (8)*, if we run Algorithm, 1 we obtain*

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta(1-\sqrt{1-\delta})^2)\|\mathbf{g}_t\|^2. \quad (15)$$

Remark 7. *Similar to the situation of one round compression, we do not encounter this case in simulation (Section V). The study and analysis of this case is mainly of theoretical importance.*

Case 3: Note that, since we consider compressed gradient along with compression local Hessian gradient product, acquiring the required theoretical guarantee seems quite challenging. Hence we fall back to the one round compression scenario. The local gradients are communicated without any compression and the follow the update rule of Case 3 of one round compression. The convergence result of Theorem 3 holds here too.

Remark 8. *In Section V, we implement this setting in experiments and observe that this case never happens. Hence, case 3 in not a practical deterrent to the convergence of Algorithm 1 with two round compression.*

V. Experimental Result

In this section we provide experimental validation of Algorithm 1. For compression we choose the following scheme: for any given vector $x \in \mathbb{R}^d$ the compressor outputs $\mathcal{Q}(x) = \frac{\|x\|_1}{d}\text{sign}(x)$ where $\text{sign}(x)$ is the quantized vector and $\frac{\|x\|_1}{d}$ is the scaling factor.

We consider regularized logistic regression for binary classification defined as

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\log(1+\exp(-y_i\mathbf{x}_i^T\mathbf{w})) + \frac{1}{2n}\|\mathbf{w}\|^2 \quad (16)$$

where $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$ are data and $\{y_i\}_{i=1}^n \in \{-1,+1\}$ are the corresponding labels. We choose *a5a* (number of training data, $n = 6414$, dimension $d = 123$), *mushroom* (training data, $n = 8124$, dimension $d = 112$) and *w1a* (training data $n = 2477$, dimension $d = 300$), binary classification datasets from UCI repository [16] . We simulate the distributed set up by partitioning the data into 10 different worker machines. In Figure 1, we plot the norm of gradient ($\|\mathbf{g}_t\|$) to validate the optimization method described in the theoretical analysis. We choose $\theta = 0.01$ as defined in Algorithm 1. With this choice of the hyper-parameter, we find that for all the algorithms only case 1 occurs in all the iterations.

Figure 1 shows that even with compression, Algorithm 1 converges. We observe that with a decrease in communication cost, the convergence gets slower. DINGO, which has no compression ($\delta = 1$) shows the fastest convergence and two round compression (with least communication cost) shows the slowest rate. One round compression is in between them in terms of communication and convergence rate.

We also compare the performance of Algorithm 1 (two round compression) with the compressed first order algorithm of [23] with identical learning rates. A threshold of 0.02 on the gradient norm is set as stopping criterion. For the mushroom data [16], we observe that Algorithm 1 communicates a total of 576 bits per machine, whereas [23, Algorithm 1] requires 1008 bits. Hence, a total savings of $432 \times m$ (m: number of machines) or a savings of 43% in bits is achieved. However, for Algorithm 1, the computation complexity at local machines are more than that of first order algorithm. This can be seen as a complexity communication trade-off.

VI. Conclusion and Future Work

In this work, we address the problem of communication efficiency in distributed second order optimization methods for the first time. We provide algorithms for three different settings that we also analyze and validate with experimental results. The most challenging future work would be to perform a theoretical analysis of the one round communication setting (setting 3 in Section I) under reasonable assumptions.

REFERENCES

[1] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[2] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3329–3337.

[3] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*, 2018, pp. 9850–9861.

[4] D. Alistarh, D. Grubic, J. Liu, R. Tomioka, and M. Vojnovic, "Communication-efficient stochastic gradient descent, with applications to neural networks," 2017.

[5] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems*, 2018, pp. 5973–5983.

[6] J. Acharya, C. De Sa, D. J. Foster, and K. Sridharan, "Distributed learning with sublinear communication," *arXiv preprint arXiv:1902.11259*, 2019.

[7] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signsgd: Compressed optimisation for non-convex problems," *arXiv preprint arXiv:1802.04434*, 2018.

[8] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

[9] V. Gandikota, R. K. Maity, and A. Mazumdar, "vqsgd: Vector quantized stochastic gradient descent," *arXiv preprint arXiv:1911.07971*, 2019.

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

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

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

[13] R. Crane and F. Roosta, "Dingo: Distributed Newton-type method for gradient-norm optimization," in *Advances in Neural Information Processing Systems*, 2019.

[14] F. Roosta, Y. Liu, P. Xu, and M. W. Mahoney, "Newton-mr: Newton's method without smoothness or convexity," *arXiv preprint arXiv:1810.00303*, 2018.

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

[16] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[17] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes," in *International Conference on Machine Learning*, 2019, pp. 3252–3261.

[18] V. Gupta, S. Kadhe, T. Courtade, M. W. Mahoney, and K. Ramchandran, "Oversketched newton: Fast convex optimization for serverless systems," *arXiv preprint arXiv:1903.08857*, 2019.

[19] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in neural information processing systems*, 2017, pp. 1509–1519.

[20] N. Ivkin, D. Rothchild, E. Ullah, V. Braverman, I. Stoica, and R. Arora, "Communication-efficient distributed sgd with sketching," *arXiv preprint arXiv:1903.04488*, 2019.

[21] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in *Advances in Neural Information Processing Systems*, 2018, pp. 4447–4458.

[22] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "signsgd with majority vote is communication efficient and byzantine fault tolerant," *arXiv preprint arXiv:1810.05291*, 2018.

[23] A. Ghosh, R. K. Maity, S. Kadhe, A. Mazumdar, and K. Ramchandran, "Communication-efficient and byzantine-robust distributed learning," *arXiv preprint arXiv:1911.09721*, 2019.