DINO: Distributed Newton-Type Optimization Method

Rixon Crane [1] **Fred Roosta** [1 2]

Abstract

We present a novel communication-efficient Newton-type algorithm for finite-sum optimization over a distributed computing environment. Our method, named DINO, overcomes both theoretical and practical shortcomings of similar existing methods. Under minimal assumptions, we guarantee global sub-linear convergence of DINO to a first-order stationary point for general nonconvex functions and arbitrary data distribution over the network. Furthermore, for functions satisfying Polyak-Lojasiewicz (PL) inequality, we show that DINO enjoys a linear convergence rate. Our proposed algorithm is practically parameter free, in that it will converge regardless of the selected hyper-parameters, which are easy to tune. Additionally, its sub-problems are simple linear least-squares, for which efficient solvers exist, and numerical simulations demonstrate the efficiency of DINO as compared with similar alternatives.

1. Introduction

Consider the generic finite-sum optimization problem

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

in a centralized distributed computing environment comprised of one central driver machine communicating to m worker machines, where each worker i only has access to f_i. A common application of this problem is where each worker i has access to a portion of n data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, indexed by a set $S_i \subseteq \{1, \ldots, n\}$, with

$$f_i(\mathbf{w}) = \frac{|S_i|}{n} \sum_{j\in S_i} \ell_j(\mathbf{w}; \mathbf{x}_j), \tag{2}$$

where ℓ_i is a loss function corresponding to $\mathbf{x}_i$ and parameterized by $\mathbf{w}$. For example, such settings are popular in industry in the form of federated learning when data is collected and processed locally, which increases computing resources and facilitates data privacy (Agarwal et al., 2018). Another example is in big data regimes, where it is more practical, or even necessary, to partition and store large datasets on separate machines (Zhang & Lin, 2015). Distributing machine learning model parameters is also becoming a necessity as some highly successful models now contain billions of parameters, such as GPT-2 (Radford et al., 2019; Lee et al., 2014).

The need for distributed computing has motivated the development of many frameworks. For example, the popular machine learning packages PyTorch (Paszke et al., 2017) and Tensorflow (Abadi et al., 2016) contain comprehensive functionality for distributed training of machine learning models. Despite many benefits to distributed computing, there are significant computational bottlenecks, such as those introduced through bandwidth and latency (Shamir & Srebro, 2014; Li et al., 2014; Wangni et al., 2018). Frequent transmission of data is expensive in terms of time and physical resources (Zhang & Lin, 2015). For example, even transferring data on a local machine can be the main contributor to energy consumption (Shalf et al., 2011).

With the bottleneck of communication, there has recently been significant focus on developing communication-efficient distributed optimization algorithms. This is particularly apparent in regards to popular first-order methods, such as stochastic gradient descent (SGD) (Haddadpour et al., 2019; Ivkin et al., 2019; Vogels et al., 2019; Basu et al., 2019; Teng et al., 2019; Zheng et al., 2019). As first-order methods solely rely on gradient information, which can often be computed easily in parallel, they are usually straightforward to implement in a distributed setting. However, their typical inherent nature of performing many computationally inexpensive iterations, which is suitable and desirable on a single machine, leads to significant data transmission costs and ineffective utilization of increased computing resources in a distributed computing environment (Wang et al., 2018).

[*]Equal contribution [1]School of Mathematics and Physics, University of Queensland, Australia [2]International Computer Science Institute, Berkeley, CA, USA. Correspondence to: Rixon Crane <r.crane@uq.edu.au>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

Related Work

In contrast to first-order methods, second-order methods perform more computation per iteration and, as a result, often require far fewer iterations to achieve similar results. In distributed settings, these properties directly translate to more efficient utilization of the increased computing resources and far fewer communications over the network. Motivated by this potential, several distributed Newton-type algorithms have recently been developed, most notably DANE (Shamir et al., 2014), DiSCO (Zhang & Lin, 2015), InexactDANE and AIDE (Reddi et al., 2016), GIANT (Wang et al., 2018), and DINGO (Crane & Roosta, 2019).

While each of these second-order distributed methods have notable benefits, they all come with disadvantages that limit their applicability. DiSCO and GIANT are simple to implement, as they involve sub-problems in the form of linear systems. Whereas, the sub-problems of InexactDANE and AIDE involve non-linear optimization problems and their hyper-parameters are difficult and time consuming to tune. DiSCO and GIANT rely on strong-convexity assumptions, and GIANT theoretically requires particular function form and data distribution over the network in (2). In contrast, InexactDANE and AIDE are applicable to non-convex objectives. DINGO's motivation is to not require strong-convexity assumptions, i.e., it converges for invex problems (Mishra & Giorgi, 2008), and still be easy to use in practice, i.e., simple to tune hyper-parameters and linear-least squares sub-problems. DINGO achieves this by optimizing the norm of the gradient as a surrogate function. Thus, it may converge to a local maximum or saddle point in non-invex problems. Moreover, the theoretical analysis of DINGO is limited to exact update.

Contributions

We present a novel communication-efficient distributed second-order optimization algorithm that combines many of the above-mentioned desirable properties. Our algorithm is named DINO, for "**DI**stributed **N**ewton-type **O**ptimization method". Our method is inspired by the novel approach of DINGO, which allowed it to circumvent various theoretical and practical issues of other methods. However, unlike DINGO, we minimize (1) directly and our analysis involves less assumptions and is under inexact update; see Tables 1 and 2 for high-level algorithm properties.

A summary of our contributions is as follows.

1. The analysis of DINO is simple, intuitive, requires very minimal assumptions and can be applied to arbitrary non-convex functions. Namely, by requiring only Lipschitz continuous gradients ∇f_i, we show global sub-linear convergence for general non-convex (1). Recall that additional assumptions are typically required for the analysis of second-order methods. Such common assumptions include strong convexity, e.g., in (Roosta & Mahoney, 2019), and Lipschitz continuous Hessian, e.g., in (Xu et al., 2019), which are both required for GIANT. Although the theory of DINGO does not require these, it still assumes additional unconventional properties of the Hessian and, in addition, is restricted to invex problems, as a strict sub-class of general non-convex models. Furthermore, in our analysis, we don't assume specific function form or data distribution for applications of the form (2). For example, this is in contrast to GIANT, which is restricted to loss functions involving linear predictor models and specific data distributions.

2. DINO is practically parameter free, in that it will converge regardless of the selected hyper-parameters. This is in sharp contrast to many first-order methods. The hyper-parameters of InexactDANE and AIDE require meticulous fine-tuning and these are sensitive to the given application. DINO is simple to tune and performs well across a variety of problems without modification of the hyper-parameters.

3. The sub-problems of DINO are simple. Like DINGO, the sub-problems of our method are simple linear least-squares problems for which efficient and robust direct and iterative solvers exists. In contrast, non-linear optimization sub-problems, such as those arising in InexactDANE and AIDE, can be difficult to solve and often involve additional hard to tune hyper-parameters.

Notation and Definitions

Throughout the paper, vectors and matrices are denoted by bold lower-case and bold upper-case letters, respectively, e.g., $\mathbf{v}$ and $\mathbf{V}$. We use regular lower-case and upper-case letters to denote scalar constants, e.g., d or L. The common *Euclidean inner product* is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T\mathbf{y}$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Given a vector $\mathbf{v}$ and matrix $\mathbf{A}$, we denote their vector ℓ_2 norm and matrix *spectral* norm as $\|\mathbf{v}\|$ and $\|\mathbf{A}\|$, respectively. The *Moore–Penrose inverse* of $\mathbf{A}$ is denoted by $\mathbf{A}^\dagger$. We let $\mathbf{w}_t \in \mathbb{R}^d$ denote the point at iteration t. For notational convenience, we denote $\mathbf{g}_{t,i} \triangleq \nabla f_i(\mathbf{w}_t)$, $\mathbf{g}_t \triangleq \nabla f(\mathbf{w}_t)$, $\mathbf{H}_{t,i} \triangleq \nabla^2 f_i(\mathbf{w}_t)$ and $\mathbf{H}_t \triangleq \nabla^2 f(\mathbf{w}_t)$. We also let

$$\tilde{\mathbf{H}}_{t,i} \triangleq \begin{bmatrix} \mathbf{H}_{t,i} \\ \phi \mathbf{I} \end{bmatrix} \in \mathbb{R}^{2d \times d} \quad \text{and} \quad \tilde{\mathbf{g}}_t \triangleq \begin{pmatrix} \mathbf{g}_t \\ \mathbf{0} \end{pmatrix} \in \mathbb{R}^{2d}, \quad (3)$$

where $\phi > 0$, $\mathbf{I}$ is the identity matrix, and $\mathbf{0}$ is the zero vector. We say that a *communication round* is performed when the driver uses a *broadcast* or *reduce* operation to send or receive information to or from the workers in parallel, respectively. For example, computing the gradient $\mathbf{g}_t$ requires two communication rounds, i.e., the driver broadcasts $\mathbf{w}_t$ and then, using a reduce operation, receives $\mathbf{g}_{t,i}$ from all workers to then form $\mathbf{g}_t = \sum_{i=1}^m \mathbf{g}_{t,i}/m$.

Table 1. Comparison of problem class, function form and data distribution. DINGO is suited to invex problems in practice, as it may converge to a local maximum or saddle point in non-invex problems (Crane & Roosta, 2019). This is a modified table from (Crane & Roosta, 2019).

	Problem Class	Function Form	Data Distribution
DINO	Non-Convex	Any	Any
DINGO (Crane & Roosta, 2019)	Invex	Any	Any
GIANT (Wang et al., 2018)	Strongly Convex	$\ell_j(\mathbf{w}; \mathbf{x}_j) = \psi_j(\langle \mathbf{w}, \mathbf{x}_j \rangle) + \gamma \|\mathbf{w}\|^2$ in (2)	$\lvert S_i \rvert > d$ in (2)
DiSCO (Zhang & Lin, 2015)	Strongly Convex	Any	Any
InexactDANE (Reddi et al., 2016)	Non-Convex	Any	Any
AIDE (Reddi et al., 2016)	Non-Convex	Any	Any

Table 2. Comparison of number of hyper-parameters (under exact update), communication rounds per iteration (under inexact update) and the type of optimization problem to solve in the sub-problem. Additional hyper-parameters may be introduced under inexact update, such as in the solver used for the non-linear optimization sub-problems of InexactDANE and AIDE. We assume DINO, DINGO and GIANT use two communication rounds per iteration for line-search. This is a modified table from (Crane & Roosta, 2019).

	Number of Hyper-parameters (Under Exact Update)	Communication Rounds Per Iteration (Under Inexact Update)	Sub-Problem Optimization Type
DINO	2	6	Linear
DINGO (Crane & Roosta, 2019)	2	4 to 8	Linear
GIANT (Wang et al., 2018)	0	6	Linear
DiSCO (Zhang & Lin, 2015)	0	$2 + 2 \cdot$ (sub-problem iterations)	Linear
InexactDANE (Reddi et al., 2016)	2	4	Non-Linear
AIDE (Reddi et al., 2016)	3	$4 \cdot$ (inner InexactDANE iterations)	Non-Linear

2. Derivation

In this section, we describe the derivation of DINO, as depicted in Algorithm 1. Each iteration t involves computing an update direction $\mathbf{p}_t$ and a step-size α_t and then forming the next iterate $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{p}_t$.

Update Direction

When forming the update direction $\mathbf{p}_t$, computing $\tilde{\mathbf{H}}_{t,i}^{\dagger} \tilde{\mathbf{g}}_t$ and $(\tilde{\mathbf{H}}_{t,i}^T \tilde{\mathbf{H}}_{t,i})^{-1} \mathbf{g}_t$ constitute the sub-problems of DINO, where $\tilde{\mathbf{H}}_{t,i}$ and $\tilde{\mathbf{g}}_t$ are as in (3). Despite these being the solutions of simple linear least-squares problems, it is still unreasonable to assume these will be computed exactly. In this light, we only require that the approximate solutions satisfy the following conditions.

Condition 1 (Inexactness Condition)**.** *For all iterations t, all worker machines $i = 1, \ldots, m$ are able to compute approximations $\mathbf{v}_{t,i}^{(1)}$ and $\mathbf{v}_{t,i}^{(2)}$ of $\tilde{\mathbf{H}}_{t,i}^{\dagger} \tilde{\mathbf{g}}_t$ and $(\tilde{\mathbf{H}}_{t,i}^T \tilde{\mathbf{H}}_{t,i})^{-1} \mathbf{g}_t$, respectively, that satisfy:*

$$\|\tilde{\mathbf{H}}_{t,i}^T \tilde{\mathbf{H}}_{t,i} \mathbf{v}_{t,i}^{(1)} - \mathbf{H}_{t,i} \mathbf{g}_t\| \leq \varepsilon_i^{(1)} \|\mathbf{H}_{t,i} \mathbf{g}_t\|, \tag{4a}$$

$$\|\tilde{\mathbf{H}}_{t,i}^T \tilde{\mathbf{H}}_{t,i} \mathbf{v}_{t,i}^{(2)} - \mathbf{g}_t\| \leq \varepsilon_i^{(2)} \|\mathbf{g}_t\|, \tag{4b}$$

$$\langle \mathbf{v}_{t,i}^{(2)}, \mathbf{g}_t \rangle > 0, \tag{4c}$$

where $0 \leq \varepsilon_i^{(1)}, \varepsilon_i^{(2)} < 1$ are constants.

For practical implementations of DINO, as with DINGO, DiSCO and GIANT, we don't need to compute or store explicitly formed Hessian matrices, i.e., our implementations are Hessian-free. Namely, approximations of $\tilde{\mathbf{H}}_{t,i}^{\dagger} \tilde{\mathbf{g}}_t$ and $(\tilde{\mathbf{H}}_{t,i}^T \tilde{\mathbf{H}}_{t,i})^{-1} \mathbf{g}_t$ can be efficiently computed using iterative least-squares solvers that only require access to Hessian-vector products, such as in our implementation described in Section 4. These products can be computed at a similar cost to computing the gradient (Schraudolph, 2002). Hence, DINO is applicable to (1) with a large dimension d.

Condition 1 has a significant practical benefit. Namely, the criteria (4) are practically verifiable as they don't involve any unknowable terms. Requiring $\varepsilon_i^{(1)}, \varepsilon_i^{(2)} < 1$ ensures that the approximations are simply better than the zero vector. The condition in (4c) is always guaranteed if one uses the conjugate gradient method (CG) (Nocedal & Wright, 2006), regardless of the number of CG iterations.

We now derive the update direction $\mathbf{p}_t$ for iteration t. Our approach is to construct $\mathbf{p}_t$ so that it is a suitable descent direction of (1). Namely, it satisfies $\langle \mathbf{p}_t, \mathbf{g}_t \rangle \leq -\theta \|\mathbf{g}_t\|^2$, where θ is a selected hyper-parameter of DINO. We begin by distributively computing the gradient, $\mathbf{g}_t$, of (1) and then broadcast it to all workers. Each worker i computes the vector $\mathbf{v}_{t,i}^{(1)}$, as in (4a), lets $\mathbf{p}_{t,i} = -\mathbf{v}_{t,i}^{(1)}$ and checks the condition $\langle \mathbf{v}_{t,i}^{(1)}, \mathbf{g}_t \rangle \geq \theta \|\mathbf{g}_t\|^2$.

All workers i in

$$\mathcal{I}_t \triangleq \big\{ i = 1, \ldots, m \mid \langle \mathbf{v}_{t,i}^{(1)}, \mathbf{g}_t \rangle < \theta \|\mathbf{g}_t\|^2 \big\}, \tag{5}$$

has a local update direction $\mathbf{p}_{t,i}$ that is not a suitable descent direction of (1), as $\langle \mathbf{p}_{t,i}, \mathbf{g}_t \rangle > -\theta \|\mathbf{g}_t\|^2$. We now enforce descent in their local update direction. For this, we consider the problem

$$\begin{aligned} \min_{\mathbf{p}_{t,i}} \quad & \|\mathbf{H}_{t,i}\mathbf{p}_{t,i} + \mathbf{g}_t\|^2 + \phi^2 \|\mathbf{p}_{t,i}\|^2 \\ \text{s.t.} \quad & \langle \mathbf{p}_{t,i}, \mathbf{g}_t \rangle \leq -\theta \|\mathbf{g}_t\|^2, \end{aligned} \tag{6}$$

where ϕ is a selected hyper-parameter of DINO as in (3). It is easy to see that, when $\langle \tilde{\mathbf{H}}_{t,i}^{\dagger} \tilde{\mathbf{g}}_t, \mathbf{g}_t \rangle < \theta \|\mathbf{g}_t\|^2$, the problem (6) has the exact solution

$$\begin{aligned} \mathbf{p}_{t,i} &= -\tilde{\mathbf{H}}_{t,i}^{\dagger} \tilde{\mathbf{g}}_t - \lambda_{t,i} (\tilde{\mathbf{H}}_{t,i}^T \tilde{\mathbf{H}}_{t,i})^{-1} \mathbf{g}_t, \\ \lambda_{t,i} &= \frac{-\langle \tilde{\mathbf{H}}_{t,i}^{\dagger} \tilde{\mathbf{g}}_t, \mathbf{g}_t \rangle + \theta \|\mathbf{g}_t\|^2}{\big\langle (\tilde{\mathbf{H}}_{t,i}^T \tilde{\mathbf{H}}_{t,i})^{-1} \mathbf{g}_t, \mathbf{g}_t \big\rangle} > 0. \end{aligned}$$

Therefore, to enforce descent, each worker $i \in \mathcal{I}_t$ computes

$$\begin{aligned} \mathbf{p}_{t,i} &= -\mathbf{v}_{t,i}^{(1)} - \lambda_{t,i} \mathbf{v}_{t,i}^{(2)}, \\ \lambda_{t,i} &= \frac{-\langle \mathbf{v}_{t,i}^{(1)}, \mathbf{g}_t \rangle + \theta \|\mathbf{g}_t\|^2}{\langle \mathbf{v}_{t,i}^{(2)}, \mathbf{g}_t \rangle} > 0, \end{aligned}$$

where $\mathbf{v}_{t,i}^{(2)}$ is as in (4b) and (4c). The term $\lambda_{t,i}$ is positive by the definition of $\mathcal{I}_t$ and the condition in (4c). This local update direction $\mathbf{p}_{t,i}$ has the property that $\langle \mathbf{p}_{t,i}, \mathbf{g}_t \rangle = -\theta \|\mathbf{g}_t\|^2$. Using a reduce operation, the driver obtains the update direction $\mathbf{p}_t = \sum_{i=1}^{m} \mathbf{p}_{t,i}/m$. By construction, $\mathbf{p}_t$ is now guaranteed to be a descent direction for (1) satisfying $\langle \mathbf{p}_t, \mathbf{g}_t \rangle \leq -\theta \|\mathbf{g}_t\|^2$.

Step-Size

We use Armijo line search to compute a step-size α_t. Namely, we choose the largest $\alpha_t > 0$ such that

$$f(\mathbf{w} + \alpha_t \mathbf{p}_t) \leq f(\mathbf{w}_t) + \alpha_t \rho \langle \mathbf{p}_t, \mathbf{g}_t \rangle, \tag{7}$$

with some constant $\rho \in (0, 1)$. As $\mathbf{p}_t$ is always a descent direction, we obtain a strict decrease in the function value. This happens regardless of the selected hyper-parameters θ and ϕ. Line search can be conducted distributively in parallel with two communication rounds, such as in our implementation in Section 4. DINO only transmits vectors of size linear in dimension d, i.e., $\mathcal{O}(d)$. This is an important property of distributed optimization methods and is consistent with DINGO, DiSCO, DANE, InexactDANE and AIDE.

Algorithm 1 DINO

```
input initial point $\mathbf{w}_0 \in \mathbb{R}^d$, gradient tolerance $\delta \geq 0$,
    maximum iterations $T$, line search parameter $\rho \in (0,1)$,
    parameter $\theta > 0$ and regularization parameter $\phi > 0$ as
    in (3).
for $t = 0, 1, 2, \ldots, T-1$ do
    Distributively compute the full gradient $\mathbf{g}_t$.
    if $\|\mathbf{g}_t\| \leq \delta$ then
        return $\mathbf{w}_t$
    else
        The driver broadcasts $\mathbf{g}_t$ and, in parallel, each
        worker $i$ computes $\mathbf{v}_{t,i}^{(1)}$ in (4a).
        In parallel, each worker $i$, such that $\langle \mathbf{v}_{t,i}^{(1)}, \mathbf{g}_t \rangle \geq$
        $\theta \|\mathbf{g}_t\|^2$, lets $\mathbf{p}_{t,i} = -\mathbf{v}_{t,i}^{(1)}$.
        In parallel, each worker $i$, such that $\langle \mathbf{v}_{t,i}^{(1)}, \mathbf{g}_t \rangle <$
        $\theta \|\mathbf{g}_t\|^2$, computes

            $\mathbf{p}_{t,i} = -\mathbf{v}_{t,i}^{(1)} - \lambda_{t,i} \mathbf{v}_{t,i}^{(2)}$,
            $\lambda_{t,i} = \frac{-\langle \mathbf{v}_{t,i}^{(1)}, \mathbf{g}_t \rangle + \theta \|\mathbf{g}_t\|^2}{\langle \mathbf{v}_{t,i}^{(2)}, \mathbf{g}_t \rangle} > 0$,

        where $\mathbf{v}_{t,i}^{(2)}$ is as in (4b) and (4c).
        Using a reduce operation, the driver computes
        $\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^{m} \mathbf{p}_{t,i}$.
        Choose the largest $\alpha_t > 0$ such that

            $f(\mathbf{w} + \alpha_t \mathbf{p}_t) \leq f(\mathbf{w}_t) + \alpha_t \rho \langle \mathbf{p}_t, \mathbf{g}_t \rangle$.

        The driver computes $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{p}_t$.
    end if
end for
return $\mathbf{w}_T$.
```

3. Analysis

In this section, we present convergence results for DINO. We assume that f, in (1), attains its minimum on some non-empty subset of $\mathbb{R}^d$ and we denote the corresponding optimal function value by f^*. As was previously mentioned, we are able to show global sub-linear convergence under minimal assumptions. Specifically, we only assume that the local gradient ∇f_i, on each worker i, is Lipschitz continuous.

Assumption 1 (Local Lipschitz Continuity of Gradient)**.** *The function f_i in (1) is twice differentiable for all $i = 1, \ldots, m$. Moreover, for all $i = 1, \ldots, m$, there exists constants $L_i \in (0, \infty)$ such that*

$$\big\| \nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y}) \big\| \leq L_i \|\mathbf{x} - \mathbf{y}\|,$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.

As already mentioned, assuming Lipschitz continuous gradient is common place. Recall that Assumption 1 implies

$$\big\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\big\| \leq L\|\mathbf{x} - \mathbf{y}\|, \tag{8}$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, where $L \triangleq \sum_{i=1}^m L_i/m$. This in turn gives for all $\alpha \geq 0$

$$f(\mathbf{w}_t + \alpha\mathbf{p}_t) \leq f(\mathbf{w}_t) + \alpha\big\langle\mathbf{p}_t, \mathbf{g}_t\big\rangle + \frac{\alpha^2 L}{2}\|\mathbf{p}_t\|^2. \tag{9}$$

Recall, for a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, with non-singular square matrix $\mathbf{A}$, the condition number of $\mathbf{A}$ is $\kappa(\mathbf{A}) \triangleq \|\mathbf{A}^{-1}\|\|\mathbf{A}\|$. As $\tilde{\mathbf{H}}_{t,i}$ has full column rank, the matrix $\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i}$ is invertible and, under Assumption 1, has condition number at most $(L_i^2 + \phi^2)/\phi^2$. Therefore, under Condition 1 and Assumption 1 we have

$$\begin{aligned}
&\|\mathbf{v}_{t,i}^{(1)} - \tilde{\mathbf{H}}_{t,i}^{\dagger}\tilde{\mathbf{g}}_t\| \\
&\qquad \leq \varepsilon_i^{(1)}\left(\frac{L_i^2 + \phi^2}{\phi^2}\right)\|\tilde{\mathbf{H}}_{t,i}^{\dagger}\tilde{\mathbf{g}}_t\|, \qquad (10a)\\
&\big\|\mathbf{v}_{t,i}^{(2)} - (\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i})^{-1}\mathbf{g}_t\big\| \\
&\qquad \leq \varepsilon_i^{(2)}\left(\frac{L_i^2 + \phi^2}{\phi^2}\right)\big\|(\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i})^{-1}\mathbf{g}_t\big\|, \qquad (10b)
\end{aligned}$$

for all iterations t and all workers $i = 1, \ldots, m$. We also have the upper bound

$$\|\tilde{\mathbf{H}}_{t,i}^{\dagger}\| \leq \frac{1}{\phi}, \tag{11}$$

for all iterations t and all $i = 1, \ldots, m$; see (Crane & Roosta, 2019) for a proof.

Theorem 1 (Convergence of DINO)**.** *Suppose Assumption 1 holds and that we run Algorithm 1 with inexact update such that Condition 1 holds with $\varepsilon_i^{(2)} < 2\sqrt{K_i}/(1 + K_i)$ for all $i = 1, \ldots, m$, with $K_i = (L_i^2 + \phi^2)/\phi^2$. Then for all iterations t we have $f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) - \tau\rho\theta\|\mathbf{g}_t\|^2$ with constants*

$$\begin{aligned}
\tau &= \frac{2(1-\rho)\theta}{La^2}, & (12a)\\
a &= \frac{1}{\phi}\left(1 + \frac{1}{m}\sum_{i=1}^m \varepsilon_i^{(1)}K_i\right) + \frac{1}{m}\sum_{i=1}^m b_i, & (12b)\\
b_i &= \left(\frac{1 + \varepsilon_i^{(2)}K_i}{1 - \varepsilon_i^{(2)}(1 + K_i)/(2\sqrt{K_i})}\right) & (12c)\\
&\quad \times \left(\frac{1}{\phi}\left(1 + \varepsilon_i^{(1)}K_i\right) + \theta\right)\sqrt{K_i}, & (12d)
\end{aligned}$$

where ρ, θ and ϕ are as in Algorithm 1, L_i are as in Assumption 1, L is as in (8)*, $\varepsilon_i^{(1)}$ are as in* (4a)*, and $\varepsilon_i^{(2)}$ are as in* (4b).

Proof. Recall that for iteration t, each worker $i \in \mathcal{I}_t$, as defined in (5), computes

$$\begin{aligned}
\mathbf{p}_{t,i} &= -\mathbf{v}_{t,i}^{(1)} - \lambda_{t,i}\mathbf{v}_{t,i}^{(2)},\\
\lambda_{t,i} &= \frac{-\langle\mathbf{v}_{t,i}^{(1)}, \mathbf{g}_t\rangle + \theta\|\mathbf{g}_t\|^2}{\langle\mathbf{v}_{t,i}^{(2)}, \mathbf{g}_t\rangle} > 0.
\end{aligned}$$

The term $\lambda_{t,i}$ is both well-defined and positive by the definition of $\mathcal{I}_t$ and the condition in (4c). The inexactness condition in (4b) implies

$$\begin{aligned}
&-\langle\mathbf{v}_{t,i}^{(2)}, \mathbf{g}_t\rangle + \big\langle(\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i})^{-1}\mathbf{g}_t, \mathbf{g}_t\big\rangle\\
&\quad = -\big\langle\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i}\mathbf{v}_{t,i}^{(2)} - \mathbf{g}_t, (\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i})^{-1}\mathbf{g}_t\big\rangle\\
&\quad \leq \varepsilon_i^{(2)}\|\mathbf{g}_t\|\big\|(\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i})^{-1}\mathbf{g}_t\big\|.
\end{aligned}$$

By Assumption 1 and the Kantorovich inequality (Gustafson, 1995), we have

$$\begin{aligned}
&\varepsilon_i^{(2)}\|\mathbf{g}_t\|\big\|(\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i})^{-1}\mathbf{g}_t\big\|\\
&\quad \leq \varepsilon_i^{(2)}\frac{1 + K_i}{2\sqrt{K_i}}\big\langle(\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i})^{-1}\mathbf{g}_t, \mathbf{g}_t\big\rangle.
\end{aligned}$$

Therefore,

$$\langle\mathbf{v}_{t,i}^{(2)}, \mathbf{g}_t\rangle \geq \left(1 - \varepsilon_i^{(2)}\frac{1 + K_i}{2\sqrt{K_i}}\right)\big\langle(\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i})^{-1}\mathbf{g}_t, \mathbf{g}_t\big\rangle,$$

where the right-hand side is positive by the assumption $\varepsilon_i^{(2)} < 2\sqrt{K_i}/(1 + K_i)$ and the condition $\|\mathbf{g}_t\| > \delta$ in Algorithm 1.

It follows from (10) and (11) that

$$\begin{aligned}
&\lambda_{t,i}\|\mathbf{v}_{t,i}^{(2)}\|\\
&\leq \left(\frac{1 + \varepsilon_i^{(2)}K_i}{1 - \varepsilon_i^{(2)}(1 + K_i)/(2\sqrt{K_i})}\right)\\
&\quad \times \left(-\langle\mathbf{v}_{t,i}^{(1)}, \mathbf{g}_t\rangle + \theta\|\mathbf{g}_t\|^2\right)\frac{\big\|(\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i})^{-1}\mathbf{g}_t\big\|}{\big\|(\tilde{\mathbf{H}}_{t,i}^T)^{\dagger}\mathbf{g}_t\big\|^2}\\
&\leq \frac{1}{\phi}\left(\frac{1 + \varepsilon_i^{(2)}K_i}{1 - \varepsilon_i^{(2)}(1 + K_i)/(2\sqrt{K_i})}\right)\\
&\quad \times \left(\frac{\|\mathbf{v}_{t,i}^{(1)}\|\|\mathbf{g}_t\| + \theta\|\mathbf{g}_t\|^2}{\big\|(\tilde{\mathbf{H}}_{t,i}^T)^{\dagger}\mathbf{g}_t\big\|}\right)\\
&\leq \left(\frac{1 + \varepsilon_i^{(2)}K_i}{1 - \varepsilon_i^{(2)}(1 + K_i)/(2\sqrt{K_i})}\right)\\
&\quad \times \left(\frac{1}{\phi}\left(1 + \varepsilon_i^{(1)}K_i\right) + \theta\right)\sqrt{K_i}\|\mathbf{g}_t\|.
\end{aligned}$$

This and (10a), and how $\mathbf{p}_{t,i} = -\mathbf{v}_{t,i}^{(1)}$ for $i \notin \mathcal{I}_t$, imply

$$\begin{aligned}\|\mathbf{p}_t\| &\leq \frac{1}{m}\left(\sum_{i\notin\mathcal{I}_t}\|\mathbf{p}_{t,i}\| + \sum_{i\in\mathcal{I}_t}\|\mathbf{p}_{t,i}\|\right)\\ &\leq \frac{1}{m}\left(\sum_{i=1}^{m}\|\mathbf{v}_{t,i}^{(1)}\| + \sum_{i\in\mathcal{I}_t}\lambda_{t,i}\|\mathbf{v}_{t,i}^{(2)}\|\right)\\ &\leq a\|\mathbf{g}_t\|,\end{aligned}$$

where a is as in (12b). This and (9) imply

$$f(\mathbf{w}_t + \alpha\mathbf{p}_t) \leq f(\mathbf{w}_t) + \alpha\langle\mathbf{p}_t, \mathbf{g}_t\rangle + \frac{\alpha^2 L a^2}{2}\|\mathbf{g}_t\|^2, \quad (13)$$

for all $\alpha \geq 0$.

For all $\alpha \in (0, \tau]$, where τ is as in (12a), we have

$$\frac{\alpha^2 L a^2}{2}\|\mathbf{g}_t\|^2 \leq \alpha(1-\rho)\theta\|\mathbf{g}_t\|^2,$$

and as $\langle\mathbf{p}_t, \mathbf{g}_t\rangle \leq -\theta\|\mathbf{g}_t\|^2$, by construction, we obtain

$$\frac{\alpha^2 L a^2}{2}\|\mathbf{g}_t\|^2 \leq \alpha(\rho-1)\langle\mathbf{p}_t, \mathbf{g}_t\rangle.$$

From this and (13) we have

$$f(\mathbf{w}_t + \alpha\mathbf{p}_t) \leq f(\mathbf{w}_t) + \alpha\rho\langle\mathbf{p}_t, \mathbf{g}_t\rangle,$$

for all $\alpha \in (0, \tau]$. Therefore, line-search (7) will pass for some step-size $\alpha_t \geq \tau$. Moreover, $f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) - \tau\rho\theta\|\mathbf{g}_t\|^2$. □

Theorem 1 implies a global sub-linear convergence rate for DINO. Namely, as $f^* \leq f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) - \tau\rho\theta\|\mathbf{g}_t\|^2$, we have

$$\sum_{k=1}^{t}\|\mathbf{g}_k\|^2 \leq \frac{f(\mathbf{w}_0) - f^*}{\tau\rho\theta},$$

for all iterations t. This implies $\lim_{t\to\infty}\|\mathbf{g}_t\| = 0$ and

$$\min_{0\leq k\leq t}\left\{\|\mathbf{g}_k\|^2\right\} \leq \frac{f(\mathbf{w}_0) - f^*}{t\tau\rho\theta},$$

for all iterations t. Reducing the inexactness error $\varepsilon_i^{(1)}$ or $\varepsilon_i^{(2)}$ in Condition 1 lead to improved constants in the rate obtained in Theorem 1.

The hyper-parameters θ and ϕ have intuitive effects on DINO. Increasing θ will also increase the chance of $\mathcal{I}_t$, in (5), being large. In fact, if $\mathcal{I}_t$ is empty for all iterations t, then, in Theorem 1, the condition on $\varepsilon_i^{(2)}$ can be removed and the term τ can be improved to

$$\tau = \frac{2(1-\rho)\theta}{La^2}, \quad a = \frac{1}{\phi}\left(1 + \frac{1}{m}\sum_{i=1}^{m}\varepsilon_i^{(1)}\frac{L_i^2 + \phi^2}{\phi^2}\right).$$

The hyper-parameter ϕ controls the condition number of $\tilde{\mathbf{H}}_{t,i}^T\tilde{\mathbf{H}}_{t,i}$, which is at most $(L_i^2 + \phi^2)/\phi^2$. Increasing ϕ will decrease the condition number and make the sub-problems of DINO easier to solve, as can be seen in (10), while also causing a loss of curvature information in the update direction. Also, the upper bound on $\varepsilon_i^{(2)}$ can be made arbitrarily close to 1 by increasing ϕ. In practice, simply setting θ and ϕ to be small often gives the best performance.

Theorem 1 applies to arbitrary non-convex (1) satisfying the minimal Assumption 1. Additional assumptions on the function class of (1) can lead to improved convergence rates. One such assumption is to relate the gradient $\nabla f(\mathbf{w}_t)$ to the distance of the current function value $f(\mathbf{w}_t)$ to optimality f^*. This can allow rates to be derived as iterates approach optimality. A simple assumption of this type is the long-standing Polyak-Lojasiewicz (PL) inequality (Karimi et al., 2016). A function satisfies the PL inequality if there exists a constant $\mu > 0$ such that

$$f(\mathbf{w}) - f^* \leq \frac{1}{\mu}\left\|\nabla f(\mathbf{w})\right\|^2, \quad (14)$$

for all $\mathbf{w} \in \mathbb{R}^d$. Under this inequality, DINO enjoys the following linear convergence rate.

Corollary 1 (Convergence of DINO Under PL Inequality)**.** *In addition to the assumptions of Theorem 1, suppose that the PL inequality* (14) *holds and we run Algorithm 1. Then for all iterations t we have $f(\mathbf{w}_{t+1}) - f^* \leq (1 - \tau\rho\mu\theta)\big(f(\mathbf{w}_t) - f^*\big)$, where ρ and θ are as in Algorithm 1, τ is as in Theorem 1, and μ is as in* (14)*. Moreover, for any choice of $\theta > 0$ we have $0 \leq 1 - \tau\rho\mu\theta < 1$.*

Proof. As $f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) + \alpha_t\rho\langle\mathbf{p}_t, \mathbf{g}_t\rangle$ and $\alpha_t \geq \tau$, the PL inequality, (14), implies

$$\begin{aligned}f(\mathbf{w}_{t+1}) - f(\mathbf{w}_t) &\leq \alpha_t\rho\langle\mathbf{p}_t, \mathbf{g}_t\rangle\\ &\leq -\tau\rho\theta\|\mathbf{g}_t\|^2\\ &\leq -\tau\rho\mu\theta\big(f(\mathbf{w}_t) - f^*\big),\end{aligned}$$

which gives $f(\mathbf{w}_{t+1}) - f^* \leq (1 - \tau\rho\mu\theta)\big(f(\mathbf{w}_t) - f^*\big)$.

From $\langle\mathbf{p}_t, \mathbf{g}_t\rangle \leq -\theta\|\mathbf{g}_t\|^2$ and (13) we have

$$f(\mathbf{w}_t + \alpha\mathbf{p}_t) \leq f(\mathbf{w}_t) - \alpha\theta\|\mathbf{g}_t\|^2 + \frac{\alpha^2 La^2}{2}\|\mathbf{g}_t\|^2, \quad (15)$$

for all $\alpha \geq 0$. The right-hand side of (15) is minimized when $\alpha = \theta/(La^2)$. It has a minimum value of $f(\mathbf{w}_t) - \big(\theta^2/(2La^2)\big)\|\mathbf{g}_t\|^2$, which, by (15), must be at least f^*. This and (14) imply

$$\frac{\theta^2}{2La^2}\|\mathbf{g}_t\|^2 \leq f(\mathbf{w}_t) - f^* \leq \frac{1}{\mu}\|\mathbf{g}_t\|^2,$$

which gives $\theta \leq \sqrt{2La^2/\mu}$. Therefore,

$$\tau\rho\mu\theta = \frac{2\rho\mu(1-\rho)\theta^2}{La^2} \leq 4\rho(1-\rho) \leq 1,$$

Table 3. Number of iterations completed in one hour. In one column, the driver and all five worker machines are running on one node. In another, they are running on their own instances in the distributed computing environment over AWS. The performance improvement, going from one node to AWS, is also presented. We use the code from (Crane & Roosta, 2019) to replicate their AWS results.

	Number of Iterations (Running on one Node)	Number of Iterations (Running over AWS)	Change When Going to AWS
DINO	3	12	$+300\%$
DINGO (Crane & Roosta, 2019)	3	12	$+300\%$
GIANT (Wang et al., 2018)	4	18	$+350\%$
DiSCO (Zhang & Lin, 2015)	7	19	$+171\%$
InexactDANE (Reddi et al., 2016)	213	486	$+128\%$
AIDE (Reddi et al., 2016)	214	486	$+127\%$
SGD (Chen et al., 2016)	1743	1187	-32%

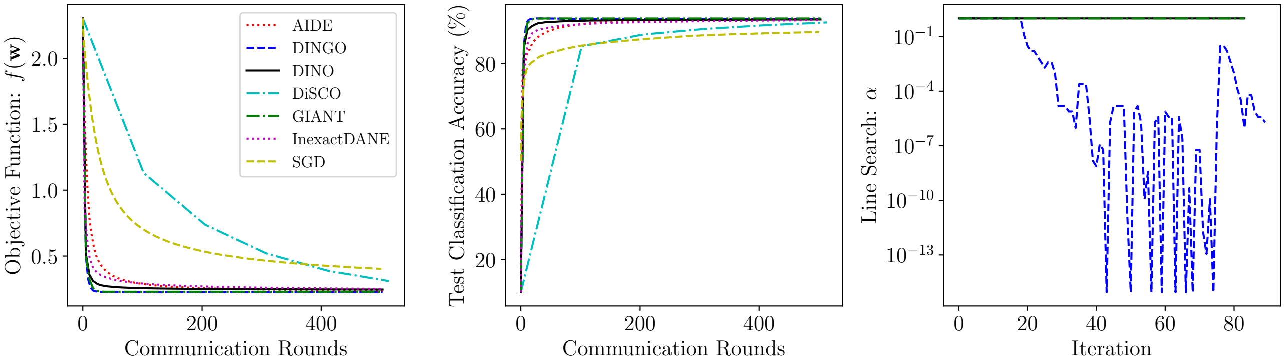

Figure 1. Softmax regression problem on the EMNIST Digits dataset. SVRG, in InexactDANE, and SGD both have a learning rate of 10^{-1} and AIDE has $\tau = 1$. Here, we have five worker nodes, i.e., $m = 5$ in (1).

which implies $0 \leq 1 - \tau\rho\mu\theta < 1$. □

The PL inequality has become widely recognized in both optimization and machine learning literature (Lei et al., 2019). The class of functions satisfying the condition contains strongly-convex functions as a sub-class and contains functions that are non-convex (Karimi et al., 2016). This inequality has shown significant potential in the analysis of over-parameterized problems and is closely related to the property of interpolation (Bassily et al., 2018; Vaswani et al., 2019). Functions satisfying (14) are a subclass of invex functions (Roosta et al., 2018). Invexity is a generalization of convexity and was considered by DINGO. Linear MLP and some linear ResNet are known to satisfy the PL inequality (Furusho et al., 2019).

4. Experiments

In this section, we examine the empirical performance of DINO in comparison to the, previously discussed, distributed second-order methods DINGO, DiSCO, GIANT, InexactDANE and AIDE. We also compare these to synchronous SGD (Chen et al., 2016). In all experiments, we consider (1) with (2), where $S_1, \ldots, S_m$ partition $\{1, \ldots, n\}$ with each having equal size n/m. In Table 3 and Figures 1 and 2, we compare performance on the strongly convex problem of softmax cross-entropy minimization with regularization on the EMNIST Digits dataset. In Figure 3, we consider the non-convex problem of non-linear least-squares without regularization on the CIFAR10 dataset with $\ell_j(\mathbf{w}; \mathbf{x}_j) = \big(y_j - \log(1 + \exp\langle \mathbf{w}, \mathbf{x}_j \rangle)\big)^2$ in (2), where y_j is the label of $\mathbf{x}_j$. Although GIANT and DiSCO require strong convexity, we run them on this problem and indicate, with an "×" on the plot, if they fail. Code is available at `https://github.com/RixonC/DINO`.

We first describe some of the implementation details. The sub-problems of DINO, DINGO, DiSCO, GIANT and InexactDANE are limited to 50 iterations, without preconditioning. For DINO, we use the well known iterative least squares solvers LSMR (Fong & Saunders, 2011) and CG to approximate $\mathbf{v}_{t,i}^{(1)}$ and $\mathbf{v}_{t,i}^{(2)}$ in Algorithm 1, respectively. For DINO, and DINGO as in (Crane & Roosta, 2019), we use

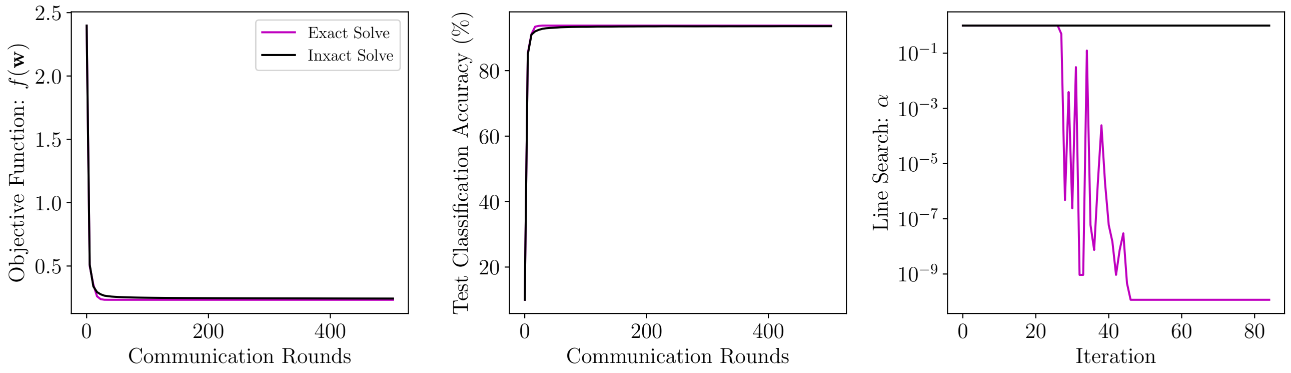

Figure 2. Softmax regression problem on the EMNIST Digits dataset. We compare DINO with exact sub-problem solve and inexact sub-problem solve. Here, we have five worker nodes, i.e., $m = 5$ in (1).

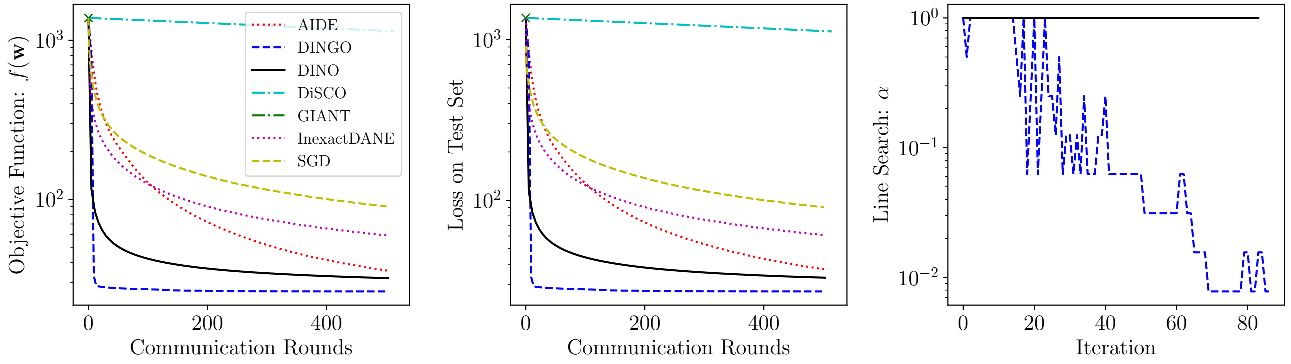

Figure 3. Non-linear least-squares problem on the CIFAR10 dataset. SVRG, in InexactDANE, and SGD have a learning rate of 10^{-4} and 10^{-3}, respectively, and AIDE has $\tau = 100$. GIANT failed immediately. Here, we have 50 worker nodes, i.e., $m = 50$ in (1).

the hyper-parameters $\theta = 10^{-4}$ and $\phi = 10^{-6}$. For DINO, DINGO and GIANT we use distributed backtracking line-search to select the largest step-size in $\{1, 2^{-1}, \ldots, 2^{-50}\}$ that passes, with an Armijo line-search parameter of 10^{-4}. For InexactDANE, we set the hyper-parameters $\eta = 1$ and $\mu = 0$, as in (Reddi et al., 2016), which gave high performance in (Shamir et al., 2014). We also use the sub-problem solver SVRG (Johnson & Zhang, 2013) and report the best learning rate from $\{10^{-5}, \ldots, 10^{5}\}$. We let AIDE call only one iteration of InexactDANE, which has the same parameters as the stand-alone InexactDANE algorithm. We also report the best acceleration parameter, τ in (Reddi et al., 2016), from $\{10^{-5}, \ldots, 10^{5}\}$. For SGD, we report the best learning rate from $\{10^{-5}, \ldots, 10^{5}\}$ and at each iteration all workers compute their gradient on a mini-batch of $n/(5m)$ data points.

The run-time is highly dependent on the distributed computing environment, which is evident in Table 3. Here, we run the methods on a single node on our local compute cluster. We also run them over a distributed environment comprised of six Amazon Elastic Compute Cloud instances via Amazon Web Services (AWS). These instances are located in Ireland, Ohio, Oregon, Singapore, Sydney and Tokyo. This setup is to highlight the effect of communication costs on run-time. As can be seen in Table 3, the second-order methods experience a notable speedup when going to the more powerful AWS setup, whereas SGD experiences a slowdown. DINO, DINGO and GIANT performed the most local computation in our experiments and they also had the largest increase in iterations. Similar behaviour to that in Table 3 can also be observed for the non-linear least-squares problem.

In Figures 1, 2 and 3, we compare the number of communication rounds required to achieve descent. We choose communication rounds as the metric, as time is highly dependent on the network. DINO is competitive with the other second-order methods, which all outperform SGD. Recall that InexactDANE and AIDE are difficult to tune. Between

Figures 1 and 3, notice the significant difference in the selected learning rate for SVRG of InexactDANE and the acceleration parameter of AIDE. Meanwhile, DINO and DINGO have consistent performance, despite not changing hyper-parameters.

In Figure 2, we compare the convergence of DINO with exact sub-problem solve and with inexact sub-problem solve. As suggested by our theory, exact update gives better convergence, while this improvement is only minor. As was previously discussed, approximations to the sub-problem solutions can be efficiently computed using iterative least-squares solvers, which only need access to Hessian-vector products, that require $\mathcal{O}(d^2)$ time. Whereas, exact solution to the sub-problems, which often necessitates access to the explicitly formed Hessian matrix, requires $\mathcal{O}(d^3)$ time. This is infeasible with moderate to large problem dimension.

In the non-convex problem in Figure 3, GIANT fails immediately as CG fails on all 50 worker nodes. DiSCO does not fail and has poor performance. This suggests that locally around the initial point $\mathbf{w}_0$, the full function f is exhibiting convexity, while the local functions f_i are not. Moreover, in Figure 3 the dimension d, which is 3072, is larger than the number of training samples, 1000, on each worker node.

5. Conclusion and Future Work

In the context of centralized distributed computing environment, we present a novel distributed Newton-type method, named DINO, which enjoys several advantageous properties. DINO is guaranteed to converge under minimal assumption, its analysis is simple and intuitive, it is practically parameter free, and it can be applied to arbitrary non-convex functions and data distributions. Numerical simulations highlight some of these properties.

The following is left for future work. First, characterizing the relationship between the hyper-parameters θ and ϕ of DINO. As was discussed, they have intuitive effects on the algorithm and they are easy to tune. However, there is a non-trivial trade off between them that will be explored in future work. Second, analysing the connection between DINO and over-parameterized problems. Finally, extending the theory of DINO to alternative forms of line search, such as having each worker perform local line search and then aggregating this information in a way that preserves particular properties.

Acknowledgements

Both authors gratefully acknowledge the generous support by the Australian Research Council Centre of Excellence for Mathematical & Statistical Frontiers (ACEMS). Fred Roosta was partially supported by the ARC DECRA Award (DE180100923). This material is based on research partially sponsored by DARPA and the Air Force Research Laboratory under agreement number FA8750-17-2-0122. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA and the Air Force Research Laboratory or the U.S. Government.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016. URL https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

Agarwal, N., Suresh, A. T., Yu, F. X., Kumar, S., and McMahan, B. cpSGD: communication-efficient and differentially-private distributed SGD. In *Advances in Neural Information Processing Systems*, pp. 7564–7575, 2018.

Bassily, R., Belkin, M., and Ma, S. On exponential convergence of sgd in non-convex over-parametrized learning. *arXiv preprint arXiv:1811.02564*, 2018.

Basu, D., Data, D., Karakus, C., and Diggavi, S. Qsparse-local-SGD: distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, pp. 14668–14679, 2019.

Chen, J., Monga, R., Bengio, S., and Jozefowicz, R. Revisiting distributed synchronous SGD. In *International Conference on Learning Representations Workshop Track*, 2016. URL https://arxiv.org/abs/1604.00981.

Crane, R. and Roosta, F. DINGO: distributed Newton-type method for gradient-norm optimization. In *Advances in Neural Information Processing Systems*, pp. 9494–9504, 2019.

Fong, D. C.-L. and Saunders, M. LSMR: an iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011.

Furusho, Y., Liu, T., and Ikeda, K. Skipping two layers in resnet makes the generalization gap smaller than skipping one or no layer. In *INNS Big Data and Deep Learning conference*, pp. 349–358. Springer, 2019.

Gustafson, K. Matrix trigonometry. *Linear algebra and its applications*, 217:117–140, 1995.

Haddadpour, F., Kamani, M. M., Mahdavi, M., and Cadambe, V. Trading redundancy for communication: speeding up distributed SGD for non-convex optimization. In *International Conference on Machine Learning*, pp. 2545–2554, 2019.

Ivkin, N., Rothchild, D., Ullah, E., Stoica, I., Arora, R., et al. Communication-efficient distributed SGD with sketching. In *Advances in Neural Information Processing Systems*, pp. 13144–13154, 2019.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pp. 315–323, 2013.

Karimi, H., Nutini, J., and Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases*, pp. 795–811, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46128-1.

Lee, S., Kim, J. K., Zheng, X., Ho, Q., Gibson, G. A., and Xing, E. P. On model parallelization and scheduling strategies for distributed machine learning. In *Advances in neural information processing systems*, pp. 2834–2842, 2014.

Lei, Y., Hu, T., Li, G., and Tang, K. Stochastic gradient descent for nonconvex learning without bounded gradient assumptions. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–7, 2019.

Li, M., Andersen, D. G., Smola, A. J., and Yu, K. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2014.

Mishra, S. K. and Giorgi, G. *Invexity and Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

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

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. 2017.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

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.

Roosta, F. and Mahoney, M. W. Sub-sampled Newton methods. *Mathematical Programming*, 174(1-2):293–326, 2019.

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

Schraudolph, N. N. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Shalf, J., Dosanjh, S., and Morrison, J. Exascale computing technology challenges. In *High Performance Computing for Computational Science – VECPAR 2010*, pp. 1–25, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-19328-6.

Shamir, O. and Srebro, N. Distributed stochastic optimization and learning. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 850–857. IEEE, 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, 2014.

Teng, Y., Gao, W., Chalus, F., Choromanska, A. E., Goldfarb, D., and Weller, A. Leader stochastic gradient descent for distributed training of deep learning models. In *Advances in Neural Information Processing Systems*, pp. 9821–9831, 2019.

Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G., and Lacoste-Julien, S. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *Advances in Neural Information Processing Systems*, pp. 3727–3740, 2019.

Vogels, T., Karimireddy, S. P., and Jaggi, M. PowerSGD: practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 14236–14245, 2019.

Wang, S., Roosta, F., Xu, P., and Mahoney, M. W. GIANT: globally improved approximate Newton method for distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 2338–2348, 2018.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1299–1309, 2018.

Xu, P., Roosta, F., and Mahoney, M. W. Newton-type methods for non-convex optimization under inexact Hessian information. *Mathematical Programming*, 2019. doi:10.1007/s10107-019-01405-z.

Zhang, Y. and Lin, X. DiSCO: distributed optimization for self-concordant empirical loss. In *International Conference on Machine Learning*, pp. 362–370, 2015.

Zheng, S., Huang, Z., and Kwok, J. Communication-efficient distributed blockwise momentum SGD with error-feedback. In *Advances in Neural Information Processing Systems*, pp. 11446–11456, 2019.