Communication Efficient Distributed Newton Method with Fast Convergence Rates

Chengchang Liu
Department of Computer Science & Engineering
The Chinese University of Hong Kong
7liuchengchang@gmail.com

Lesi Chen
School of Data Science
Fudan University
lschen19@fudan.edu.cn

Luo Luo*
School of Data Science
Fudan University
luoluo@fudan.edu.cn

John C.S. Lui
Department of Computer Science & Engineering
The Chinese University of Hong Kong
cslui@cse.cuhk.edu.hk

ABSTRACT

We propose a communication and computation efficient second-order method for distributed optimization. For each iteration, our method only requires $O(d)$ communication complexity, where d is the problem dimension. We also provide theoretical analysis to show the proposed method has the similar convergence rate as the classical second-order optimization algorithms. Concretely, our method can find $(\epsilon, \sqrt{dL\epsilon})$-second-order stationary points for nonconvex problem by $O(\sqrt{dL}\,\epsilon^{-3/2})$ iterations, where L is the Lipschitz constant of Hessian. Moreover, it enjoys a local superlinear convergence under the strongly-convex assumption. Experiments on both convex and nonconvex problems show that our proposed method performs significantly better than baselines.

CCS CONCEPTS

• **Mathematics of computing → Mathematical optimization**.

KEYWORDS

Distributed Optimization, Second-Order Methods

ACM Reference Format:
Chengchang Liu, Lesi Chen, Luo Luo, and John C.S. Lui . 2023. Communication Efficient Distributed Newton Method with Fast Convergence Rates. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599280

1 INTRODUCTION

Distributed optimization has received lots of attention in machine learning and data mining communities [3, 4, 13, 17, 20, 28, 31, 33, 35]. Its major attractive feature is that it allows solving large-scale problems in a parallel fashion, which significantly improves the efficiency for training the models [14, 15, 34]. This paper considers the distributed optimization problem of the form

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

where d is the problem dimension, n is the number of clients and f_i is the smooth local function associated with the i-th client. We focus on the scenario that each client can access its local function and communicate with a central server.

The first-order methods are popular in such distributed setting [1, 11, 12, 16, 18, 19, 22]. For each iteration of these methods, the clients compute the gradients of their local functions in parallel and the server updates the variable after receiving the local information from clients, which results the computation on each machine be efficient. The main drawback of first-order methods is that they require a large amount of iterations to find an accurate solution. As a consequence, factors like bandwidth limitation and latency in the network leads to expensive communication cost in total.

The classical second-order methods have superior convergence rates [22, 24] than the first-order methods. However, their extension to a distributed setting is non-trivial. The key challenge is how to deliver the second-order information efficiently. One straightforward strategy is to perform classical Newton-type step on the server within the aggregate Hessian, but this requires $O(d^2)$ communication complexity at each iteration, which is unacceptable in distributed settings.

Several previous works address the communication issues in distributed second-order optimization. They avoid $O(d^2)$ communication cost per iteration but have several other shortcomings. We briefly review three main approaches as follows.

- For the first one, each client performs Newton step with its local Hessian and sends the descent direction to the server, then the server aggregates all of local descent directions to update the global variable [6, 8, 25, 27, 32, 37]. Their convergence rates depend on the assumption of specific structure of the problems or data similarity. In general case, their theoretical results are no stronger than first-order methods.
- For the second one, the algorithms formulate a Newton step by a (regularized) quadratic sub-problem and solve it with distributed first-order methods [25, 30, 36, 38]. The procedure of solving sub-problem introduces additional computation and

[*] The corresponding author

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599280

communication complexity, which increases the total cost and leads to complicated implementation.

- For the third one, each client keeps the local second-order information whose change at each iteration can be stored in $O(d)$ space complexity, which leads to the efficient communication [10, 26, 29]. However, the storage of local second-order information on each client requires $O(d^2)$ space complexity. Additionally, the convergence guarantees of these methods require strong assumptions such as Lipschitz continuity of each local Hessian.

Furthermore, most existing distributed second-order methods are designed for convex optimization problems [6, 10, 26, 27, 29, 30, 32, 36–38]. Few works consider the theory for nonconvex case [8, 25]. Reddi et al. [25] provided non-asymptotic convergence rate to find approximate first-order stationary point of nonconvex objective function by distributed second-order method, but the convergence result has no advantage over first-order methods. Ghosh et al. [8] proposed a distributed cubic-regularized Newton method to find the approximate second-order stationary point, while its convergence guarantee requires very strong assumption that all of the local Hessians during iterations are sufficiently close to the global Hessian.

In this paper, we provide a novel mechanism to reduce the communication cost for distributed second-order optimization. It only communicates by gradients and Hessian-vector products. Concretely, the server forms a cubic-regularized Newton step [7, 23] based on the current aggregated gradient and the delayed Hessian which is formed by Hessian-vector products received in recent iterations. As a result, we propose a Communication and Computation Efficient DistributEd Newton (C2EDEN) method, which has the following attractive properties.

- It only requires $O(d)$ communication complexity at each iteration, matching the cost of first-order methods.
- It has simple local update rule, which does not require any subroutine with additional communication cost. Furthermore, the client avoids storing the local Hessian with $O(d^2)$ storage complexity.
- It can find an $\left(\epsilon, \sqrt{dL\epsilon}\,\right)$-second-order stationary point within $O\left(\sqrt{dL}\,\epsilon^{-3/2}\right)$ iterations for nonconvex problem, and exhibit local superlinear convergence for strongly-convex problem.
- Its convergence guarantees do not require additional assumptions such as Lipschitz continuouity of the local Hessian or strong convexity of the local function.

We summarize the main theoretical results of C2EDEN and related work in Table 1. We also provide numerical experiments on both convex and nonconvex problems to show the empirical superiority for the proposed method.

Paper Organization. The remainder of the paper is organized as follows. In Section 2 presents preliminaries. Section 3 provide the details of C2EDEN method and provides the convergence analysis. Section 4 validates our algorithm empirically to show its superiority. Finally, we conclude our work in Section 5.

2 PRELIMINARIES

We use $\|\cdot\|$ to present spectral norm and Euclidean norm of matrix and vector respectively. We denote the standard basis for $\mathbb{R}^d$ by $\{\mathbf{e}_1, \dots, \mathbf{e}_d\}$ and let $\mathbf{I}$ be the identity matrix. Given matrix $\mathbf{H} \in \mathbb{R}^{d\times d}$, we write its j-th column as $\mathbf{H}(j)$, which is equal to $\mathbf{He}_j$. We also denote the smallest eigenvalue of symmetric matrix by $\lambda_{\min}(\cdot)$.

Throughout this paper, we make the following assumption.

Assumption 2.1. We assume each local function $f_i(\,\cdot\,)$ is twice differentiable and the objective $f(\,\cdot\,)$ has a Lipschitz continuous Hessian, i.e., there exists constant $L > 0$ such that

$$\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\| \le L\|\mathbf{x} - \mathbf{y}\| \tag{2}$$

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

Compared with assuming each local Hessian $\nabla^2 f_i(\cdot)$ is L-Lipschitz continuous [9, 10, 25–27, 29, 37, 38], the Lipschitz continuity on the global Hessian $\nabla f^2(\,\cdot\,)$ in Assumption 2.1 is much weaker.

For the nonconvex case, we assume the global objective is lower bounded.

Assumption 2.2. We assume the global objective $f(\,\cdot\,)$ is lower bounded, i.e., we have $f^* \triangleq \inf_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) > -\infty$.

We also consider the specific case in which the global objective is strongly-convex.

Assumption 2.3. We assume the global objective $f(\,\cdot\,)$ is strongly-convex, i.e., there exists a constant $\mu > 0$ such that $\nabla^2 f(\mathbf{x}) \succeq \mu\mathbf{I}$ for any $\mathbf{x} \in \mathbb{R}^d$.

Note that all existing superlinear convergent distributed second-order methods require the strongly-convex assumption on each local function $f_i(\,\cdot\,)$ [9, 10, 26, 29], while our Assumption 2.3 only requires the strong convexity of the global objective.

This paper studies second-order optimization on both convex and nonconvex problems. For the convex case, we target to find approximate first-order stationary point.

Definition 2.4. We call $\mathbf{x}$ is an ϵ-first-order stationary point of $f(\cdot)$ if it satisfies $\|\nabla f(\mathbf{x})\| \le \epsilon$.

For the nonconvex case, finding global solution is intractable and the first-order stationary point may lead to the undesired saddle point. Hence, we target to find the approximate second-order stationary point to characterize the local optimality.

Definition 2.5. We call $\mathbf{x}$ is an (ϵ, δ)-second-order stationary point of $f(\cdot)$ if it satisfies $\|\nabla f(\mathbf{x})\| \le \epsilon$ and $\nabla^2 f(\mathbf{x}) \succeq -\delta\mathbf{I}$.

3 ALGORITHM AND MAIN RESULTS

We first introduce our Communication and Computation Efficient DistributEd Newton (C2EDEN) method and the main ideas, then we provide the convergence guarantees.

3.1 The C2EDEN Method

We present the details of C2EDEN in Algorithm 1, where the notation $k\%d$ presents the remainder of k divided by d, $M > 0$ is the cubic-regularized parameter and we use

$$\begin{aligned} &\mathrm{T}_M(\mathbf{g}, \mathbf{A}; \mathbf{x}) \\ &= \mathop{\arg\min}_{\mathbf{y}\in\mathbb{R}^d} \left\{ \langle \mathbf{g}, \mathbf{y}-\mathbf{x}\rangle + \frac{1}{2}\langle \mathbf{A}(\mathbf{y}-\mathbf{x}), \mathbf{y}-\mathbf{x}\rangle + \frac{M}{6}\|\mathbf{y}-\mathbf{x}\|^3 \right\} \end{aligned} \tag{3}$$

to present the solution of the cubic-regularized sub-problem for a given vector $\mathbf{g} \in \mathbb{R}^d$ and a symmetric matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$.

Table 1: We summarize the distributed optimization algorithms by their communication complexity per iteration, space complexity on each client, global convergence in the general nonconvex (NC) case, local convergence in the strongly-convex (SC) case and additional assumption for convergence guarantee.

Method	Communication Complexity	Space Complexity	Global Convergence (NC)	Local Superlinear Convergence (SC)	No Additional Assumption for Convergence
First-Order Methods [18, 22]	$O(d)$	$O(d)$	$O(\epsilon^{-2})$	✗	✓
Distributed-(Cubic)-Newton [21, 23]	$O(d^2)$	$O(d^2)$	$O(\epsilon^{-3/2})$	✓	✓
AIDE [a] [25]	$O(d)$	$O(d)$	✗	✗	✗ [f], [g]
Disco/Accelerated ADAN [a] [36, 38]	$O(d)$	$O(d)$	✗	✗	✗ [b]
(Inexact)-DANE/DANE-HB [25, 27, 37]	$O(d)$	$O(d)$	$O(\epsilon^{-2})$ [d]	✗	✗ [b], [f], [g]
GIANT [32]	$O(d)$	$O(d)$	✗	✗	✗ [b]
Local CRN [8]	$O(d)$	$O(d)$	$O(\epsilon^{-3/2})$ [e]	✗	✗ [b]
Compressed Distributed Newton [9, 10, 26]	$O(d)$	$O(d^2)$	✗	✓ [c]	✗[h]
Distributed-Quasi-Newton [29]	$O(d)$	$O(d^2)$	✗	✓	✗[f], [g], [h]
C2EDEN Algorithm 1	$O(d)$	$O(d)$	$O(\epsilon^{-3/2})$	✓	✓

[a] The method(s) have to solve a sub-problem at each iteration, which requires additional communication complexity.

[b] The method(s) require data similarity to guarantee their convergence.

[c] The local superlinear convergence requires all of the local client Hessian estimator be close to $\nabla^2 f_i(\mathbf{x}^*)$ at each iteration.

[d] The global convergence rate is only established for finding approximate first-order stationary point.

[e] The global convergence rate requires very strong assumption such that $\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\| \leq \epsilon$ and $\|\nabla^2 f_i(\mathbf{x}) - \nabla^2 f(\mathbf{x})\| \leq \sqrt{\epsilon}$.

[f] The convergence guarantee requires each $f_i(\cdot)$ is strongly-convex.

[g] The convergence guarantee requires each $\nabla f_i(\cdot)$ is Lipschitz continuous.

[h] The convergence guarantee requires each $\nabla^2 f_i(\cdot)$ is Lipschitz continuous.

We partition the Hessian at at snapshot point $\tilde{\mathbf{x}}$ into d columns

$$\nabla^2 f_i(\tilde{\mathbf{x}}) = \begin{bmatrix} | & | & & | \\ \nabla^2 f_i(\tilde{\mathbf{x}})\mathbf{e}_1 & \nabla^2 f_i(\tilde{\mathbf{x}})\mathbf{e}_2 & \cdots & \nabla^2 f_i(\tilde{\mathbf{x}})\mathbf{e}_d \\ | & | & & | \end{bmatrix}.$$

C2EDEN atomize the communication of local $\nabla^2 f_i(\tilde{\mathbf{x}})$ into d consecutive iterations by sending $\mathbf{v}_{i,k} = \nabla^2 f_i(\tilde{\mathbf{x}})\mathbf{e}_{k\%d+1}$ for $i = 1, \dots, n$ at the k-th iteration. Note that the cubic-regularized Newton steps in our method only use the Hessian at snapshot point $\tilde{\mathbf{x}}$, which is updated per d iterations. Also note that any iteration of C2EDEN avoids the construction of a full local Hessian on the client, which significantly reduces computational cost of accessing the second-order information. Additionally, the iteration of C2EDEN only communicates the local gradient $\mathbf{g}_{i,k}$ and Hessian-vector product $\mathbf{v}_{i,k}$ together, which means each client only requires $O(d)$ communication complexity and $O(d)$ space complexity to deal with its local second-order information.

The computational cost of C2EDEN is dominated by computing Hessian-vector products on clients and solving the cubic-regularized sub-problem on server. We emphasis that these two steps can be executed in parallel. Since the computation of $\mathbf{x}_{k+1}$ (line 22) on the server only depends on previous Hessian at $\tilde{\mathbf{x}}$, all of the clients are allowed to compute $\mathbf{v}_{i,k}$ (line 16) at the same time. Additionally, the reuse of Hessian at snapshot $\tilde{\mathbf{x}}$ leads to a computational complexity for achieving $\mathbf{x}_{k+1}$ in $O(d^2)$ on average [7].

We present the pipeline for one epoch of C2EDEN in Figure 1 to illustrate our mechanism.

3.2 Convergence Analysis

This section provides theoretical guarantees of C2EDEN for both nonconvex case and strongly-convex case.

At the k-th iteration of Algorithm 1 with $k = td + q$, it performs the cubic-regularized Newton step on the server by using the matrix $\mathbf{H}$ that is exactly the Hessian at the previous snapshot point $\mathbf{x}_{t(d-1)}$ and vector gradient $\mathbf{g}_k$ that is the gradient at the current point $\mathbf{x}_k$. Hence, the update rule of C2EDEN on the server can be written as

$$\mathbf{x}_{k+1} = \mathrm{T}_M\left(\nabla f(\mathbf{x}_k), \nabla^2 f\left(\mathbf{x}_{\tau(k;d)}\right); \mathbf{x}_k\right), \tag{4}$$

where $\tau(k;d) = d(\lfloor k/d \rfloor - 1)$.

3.2.1 The Analysis in Nonconvex Case. For the general non-convex case, we introduce the auxiliary quantity [7, 23]

$$\gamma(\mathbf{x}) \triangleq \max\left\{-\frac{1}{648M^2}\lambda_{\min}\left(\nabla^2 f(\mathbf{x})\right)^3, \frac{1}{72\sqrt{2M}}\|\nabla f(\mathbf{x})\|^{3/2}\right\}. \tag{5}$$

for our analysis.

Iteration	$k = td$		$\cdots\cdots$	$k = td + (d-1)$	
***i*-th Client**	$\mathbf{g}_{i,td} = \nabla f_i(\mathbf{x}_{td})$	$\mathbf{v}_{i,td} = \nabla^2 f_i(\tilde{\mathbf{x}})\mathbf{e}_1$	$\cdots\cdots$	$\mathbf{g}_{i,td+(d-1)} = \nabla f_i\big(\mathbf{x}_{td+(d-1)}\big)$	$\mathbf{v}_{i,td+(d-1)} = \nabla^2 f_i(\tilde{\mathbf{x}})\mathbf{e}_d$
Server		$\mathbf{x}_{td+1} = \mathrm{T}_M\big(\mathbf{g}_{td}, \mathbf{H}; \mathbf{x}_{td}\big)$	$\cdots\cdots$		$\mathbf{x}_{(t+1)d} = \mathrm{T}_M\big(\mathbf{g}_{td+(d-1)}, \mathbf{H}; \mathbf{x}_{td+(d-1)}\big)$

Figure 1: We present the pipeline of C2EDEN from its td-th iteration to $(td + (d-1))$-th iteration, where $\tilde{\mathbf{x}} = \mathbf{x}_{td}$ and $\mathbf{H} = \nabla^2 f\big(\mathbf{x}_{(t-1)d}\big)$. The algorithm atomizes the cost of constructing $\nabla^2 f(\tilde{\mathbf{x}}) = \nabla^2 f(\mathbf{x}_{td})$ by computing Hessian-vector products $\nabla^2 f_i(\tilde{\mathbf{x}})\mathbf{e}_1, \dots, \nabla^2 f_i(\tilde{\mathbf{x}})\mathbf{e}_d$ on clients during these d iterations. The server has already obtained matrix $\mathbf{H}$ at the $(td-1)$-th iteration and the Hessian $\nabla^2 f(\tilde{\mathbf{x}})$ is constructed for the next epoch, that is from the $(t+1)d$-th iteration to the $((t+1)d + (d-1))$-th iteration.

Algorithm 1 C2EDEN $(\mathbf{x}_0, M, K)$

```
for $k = 0, 1 \cdots d-1$ do
    for $i = 1, \dots, n$ do in parallel
        i-th client:
            compute $\mathbf{v}_{i,k} = \nabla^2 f_i(\mathbf{x}_0)\mathbf{e}_{k+1}$
    end if
    server:
        aggregate $\mathbf{H}^+(k+1) = \frac{1}{n}\sum_{i=1}^n \mathbf{v}_{i,k}$
end for
$\mathbf{x}_d = \mathbf{x}_0$
for $k = d, d+1, \dots, K$ do
    if $k\%d = 0$ then
        server:
            update $\tilde{\mathbf{x}} = \mathbf{x}_k$ and $\mathbf{H} = \mathbf{H}^+$
    end if
    for $i = 1, \dots, n$ do in parallel
        i-th client:
            compute $\mathbf{v}_{i,k} = \nabla^2 f_i(\tilde{\mathbf{x}})\mathbf{e}_{k\%d+1}$
            compute $\mathbf{g}_{i,k} = \nabla f_i(\mathbf{x}_k)$
            send $\mathbf{g}_{i,k}$ and $\mathbf{v}_{i,k}$
    end for
    server:
        aggregate $\mathbf{g}_k = \frac{1}{n}\sum_{i=1}^n \mathbf{g}_{i,k}$
        compute and broadcast $\mathbf{x}_{k+1} = \mathrm{T}_M(\mathbf{g}_k, \mathbf{H}; \mathbf{x}_k)$
        aggregate $\mathbf{H}^+(k\%d+1) = \frac{1}{n}\sum_{i=1}^n \mathbf{v}_{i,k}$
end for
```

We provide the following lemma to show the decrease of function value by a step of solving cubic-regularized sub-problem.

Lemma 3.1 ([7, Theorem 1]). *Suppose Assumption 2.1 holds and we have $M \geq L$, then the update $\mathrm{T} = \mathrm{T}_M\big(\nabla f(\mathbf{x}), \nabla^2 f(\mathbf{z}); \mathbf{x}\big)$ holds that*

$$f(\mathbf{x}) - f(\mathrm{T}) \geq \gamma(\mathrm{T}) + \frac{M}{48}\|\mathrm{T} - \mathbf{x}\|^3 - \frac{11L^3}{M^2}\|\mathbf{z} - \mathbf{x}\|^3, \qquad (6)$$

for any $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$.

Then we present some technical lemmas for later analysis.

Lemma 3.2. *For any sequence of positive numbers $\{r_k\}_{k\geq 0}$, it holds for any $m \geq 1$ that*

$$\left(\sum_{i=0}^{m-1} r_i\right)^3 \leq m^2 \sum_{i=0}^{m-1} r_i^3. \qquad (7)$$

Proof. We directly prove this result by using Jensen's inequality that $\sum_{i=0}^{m-1}\left(\frac{1}{m}r_i^3\right)^3 \leq \frac{1}{m}\sum_{i=0}^{m-1} r_i^3$. □

Lemma 3.3. *For any sequence of positive numbers $\{r_k\}_{k\geq 0}$, it holds for any $m \geq 1$ that*

$$\sum_{k=m}^{2m-1}\left(\sum_{i=0}^{k} r_i\right)^3 \leq 3(m+1)^3 \sum_{i=0}^{2m-1} r_i^3. \qquad (8)$$

Proof. We prove this statement by induction.

For $m = 1$, it follows that $(r_0 + r_1)^3 \overset{(7)}{\leq} 4(r_1^3 + r_2^3)$. Suppose the inequality (8) holds for $m = 1, 2 \dots, n-1$, then we have

$$\begin{aligned}
\sum_{k=n}^{2n-1}\left(\sum_{i=0}^{k} r_i\right)^3 &= \sum_{k=n}^{2n-3}\left(\sum_{i=0}^{k} r_i\right)^3 + \left(\sum_{i=0}^{2n-1} r_i\right)^3 + \left(\sum_{i=0}^{2n-2} r_i\right)^3 \\
&\leq \sum_{k=n-1}^{2n-3}\left(\sum_{i=0}^{k} r_i\right)^3 + \left(\sum_{i=0}^{2n-1} r_i\right)^3 + \left(\sum_{i=0}^{2n-2} r_i\right)^3 \\
&\overset{(7)}{\leq} \sum_{k=n-1}^{2n-3}\left(\sum_{i=0}^{k} r_i\right)^3 + (2n)^2 \sum_{i=0}^{2n-1} r_i^3 + (2n-1)^2 \sum_{i=0}^{2n-2} r_i^3 \\
&\overset{(8)}{\leq} 3n^3 \sum_{i=0}^{2n-3} r_i^3 + 8n^2 \sum_{i=0}^{2n-1} r_i^3 \leq 3(n+1)^3 \sum_{i=0}^{2n-1} r_i^3,
\end{aligned}$$

which finishes the induction. □

Lemma 3.4. *For any sequence of positive numbers $\{r_k\}_{k\geq 0}$, it holds that*

$$\sum_{k=m}^{m+p}\left(\sum_{i=0}^{k-1} r_i\right)^3 \leq 4(m+1)^3 \sum_{i=0}^{m+p} r_i^3 \qquad (9)$$

for any m and p such that $1 \leq p \leq m$.

Proof. We have

$$\sum_{k=m}^{m+p}\left(\sum_{i=0}^{k-1} r_i\right)^3 \le \sum_{k=m}^{m+p}\left(\sum_{i=0}^{m+p-1} r_i\right)^3$$
$$\overset{(7)}{\le} (m+p)^2(p+1)\sum_{i=0}^{m+p-1} r_i^3 \le 4(m+1)^3\sum_{i=0}^{m+p} r_i^3.$$

□

Now, we formally present the global convergence of C2EDEN in the following theorem, which shows that our algorithm can find an $(\epsilon, \sqrt{dL\epsilon})$-second-order stationary point by $O(\sqrt{dL}\,\epsilon^{-3/2})$ iterations.

Theorem 3.5. *Suppose Assumption 2.1 and 2.2 hold. Running C2EDEN (Algorithm 1) with $M = 12dL$ holds that*

$$\min_{d<i\le K}\gamma(\mathbf{x}_i) \le \frac{f(\mathbf{x}_0) - f^*}{K-d}, \tag{10}$$

which means by setting the number iterations as $K = O(\sqrt{dL}\epsilon^{-3/2})$ there exists some $\mathbf{x}_i$ with $d < i \le K$ such that

$$\|\nabla f(\mathbf{x}_i)\| \le \epsilon \quad \text{and} \quad \lambda_{\min}(\nabla^2 f(\mathbf{x}_i)) \ge -\sqrt{dL\epsilon}.$$

Proof. We write the total number of iterations as $K = dt + p$, where $t = \lfloor K/d \rfloor$ and $p = K\%d$. We denote $\mathbf{x}_{d-1} = \mathbf{x}_{d-2}\cdots = \mathbf{x}_0$ in the following analysis. We have

$$f(\mathbf{x}_d) - f(\mathbf{x}_K) = \sum_{i=d}^{K-1} f(\mathbf{x}_i) - f(\mathbf{x}_{i+1})$$
$$\overset{(6)}{\ge}\sum_{i=d}^{K-1}\left(\gamma(\mathbf{x}_{i+1}) + \frac{M}{48}\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3 - \frac{11L^3}{M^3}\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3\right) \tag{11}$$
$$= \left(\sum_{i=d}^{K-1}\gamma(\mathbf{x}_{i+1})\right) + \sum_{i=d}^{K-1}\left(\frac{M}{48}\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3 - \frac{11L^3}{M^2}\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3\right).$$

We first focus on the term of

$$\sum_{i=d}^{K-1}\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3 = \underbrace{\sum_{i=d}^{dt-1}\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3}_{A} + \underbrace{\sum_{i=dt}^{dt+p-1}\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3}_{B}.$$

For any integer $N \ge 1$, we have

$$\sum_{i=Nd}^{(N+1)d-1}\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3 \le \sum_{i=Nd}^{(N+1)d-1}\left(\sum_{k=(N-1)d}^{i-1}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|\right)^3$$
$$\le 3(d+1)^3\sum_{i=(N-1)d}^{(N+1)d-1}\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3, \tag{12}$$

where the first step comes from the triangle inequality such that

$$\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\| = \left\|\sum_{k=\tau(i;d)}^{i-1}(\mathbf{x}_{k+1} - \mathbf{x}_k)\right\| \le \sum_{k=\tau(i;d)}^{i-1}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|,$$

and the second step is obtained by using Lemma 3.3 with

$$r_k = \|\mathbf{x}_{(N-1)d+k+1} - \mathbf{x}_{(N-1)d+k}\|,$$

which leads to

$$\sum_{i=Nd}^{(N+1)d-1}\left(\sum_{k=(N-1)d}^{i-1}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|\right)^3 = \sum_{i=d}^{2d-1}\left(\sum_{k=0}^{i-1} r_k\right)^3$$
$$\le \sum_{i=d}^{2d-1}\left(\sum_{k=0}^{i} r_k\right)^3 \overset{(8)}{\le} 3(d+1)^3\sum_{k=0}^{2d-1} r_k^3$$
$$= 3(d+1)^3\sum_{i=(N-1)d}^{(N+1)d-1}\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3.$$

Hence, we have

$$A \le \sum_{N=1}^{t}\left(\sum_{i=Nd}^{(N+1)d-1}\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3\right)$$
$$\overset{(12)}{\le} 6(d+1)^3\sum_{k=d}^{(t-1)d-1}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^3$$
$$+ 3(d+1)^3\left(\sum_{k=0}^{d-1}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^3 + \sum_{k=(t-1)d}^{td}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^3\right).$$

Using Lemma 3.4 with $r_k = \|\mathbf{x}_{d(t-1)+k+1} - \mathbf{x}_{d(t-1)+k}\|$, we obtain

$$B = \sum_{i=dt}^{dt+p}\|\mathbf{x}_i - \mathbf{x}_{d(t-1)}\|^3 \le \sum_{i=d}^{d+p}\left(\sum_{k=0}^{i-1} r_k^3\right)$$
$$\overset{(9)}{\le} 4(m+1)^3\sum_{i=d(t-1)}^{dt-1}\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3 + 4(d+1)^3\sum_{dt}^{dt+p}\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3.$$

Combining above results, we have

$$\sum_{i=d}^{K-1}\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3 = A + B$$
$$\le 7(d+1)^3\sum_{k=d}^{K-1}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^3 + 3(d+1)^3\sum_{k=0}^{d-1}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^3, \tag{13}$$

which implies

$$\frac{M}{48}\sum_{i=d}^{K-1}\left(\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3\right) - \frac{11L^3}{M^2}\sum_{i=d}^{K-1}\left(\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3\right)$$
$$\overset{(13)}{\ge}\left(\frac{M}{48} - \frac{77(d+1)^3L^3}{M^2}\right)\sum_{i=d}^{K-1}\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3$$
$$- \frac{33L^3(d+1)^3}{M^2}\sum_{i=0}^{d-1}\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3 \ge 0.$$

The last inequality comes from the facts that we set $M = 12dL$ and $\mathbf{x}_0 = \mathbf{x}_1\cdots\mathbf{x}_{d-1} = \mathbf{x}_d$. Connecting above results to inequality (11), we obtain

$$f(\mathbf{x}_0) - f^* \ge f(\mathbf{x}_d) - f(\mathbf{x}_K)$$
$$\overset{(11)}{\ge}\sum_{i=d}^{K-1}\gamma(\mathbf{x}_{i+1}) + \frac{M}{48}\sum_{i=d}^{K-1}\left(\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^3\right)$$
$$- \frac{11L^3}{M^2}\sum_{i=d}^{K-1}\|\mathbf{x}_i - \mathbf{x}_{\tau(i;d)}\|^3 \ge \sum_{i=d}^{K-1}\gamma(\mathbf{x}_{i+1}),$$

which proves (10). By setting $K = O\left(\sqrt{dL}\epsilon^{-3/2}\right)$, we can find some $\mathbf{x}_i$ with $d \le i \le K$ such that $\mathbf{x}_i$ is a $\left(\epsilon, \sqrt{dL\epsilon}\right)$-second-order stationary point of $f(\cdot)$. □

3.2.2 The Analysis in Strongly-Convex Case. The classical second-order methods enjoy local superlinear convergence for minimizing the strongly-convex objective on single machine [22–24]. However, most of distributed second-order methods only achieve linear convergence rate because of the trade-off between the communication complexity and the convergence rate [25, 27, 32, 36, 38].

In contrast, the efficient communication mechanism of C2EDEN still keeps the superlinear convergence like classical second-order methods. The following theorem formally presents this property.

Theorem 3.6. *Under Assumption 2.1 and 2.3, running Algorithm 1 with $M \ge 0$ and the initial point $\mathbf{x}_0$ such that*

$$\|\nabla f(\mathbf{x}_0)\| \le \frac{\mu^2}{2(M+3L)}, \tag{14}$$

then for any $k \ge d$, we have

$$\|\nabla f(\mathbf{x}_k)\| \le \frac{\mu^2}{M+3L}\left(\frac{1}{2}\right)^{h(k)} \tag{15}$$

where $h(k) = \left((1+d)^{\lfloor k/d \rfloor \%2} + k\%d\right)(1+d)^{\lfloor \lfloor k/d \rfloor /2 \rfloor -1}$.

Proof. According to the analysis in Section 5 of Doikov et al. [7], the cubic-regularized update (4) satisfies

$$\|\nabla f(\mathbf{x}_{k+1})\| \le \frac{M+3L}{2\mu^2}\|\nabla f(\mathbf{x}_k)\|^2 + \frac{L}{\mu^2}\|\nabla f(\mathbf{x}_k)\|\|\nabla f(\mathbf{x}_{\tau(k;d)})\|.$$

We let $c = \frac{M+3L}{\mu^2}$ and $s_j = c\|\nabla f(\mathbf{x}_j)\|$, then above inequality can be written as $s_{k+1} \le \frac{1}{2}s_k^2 + \frac{1}{2}s_k s_{\iota(k;d)}$. We first use induction to show

$$s_{k+1} \le s_k \quad \text{and} \quad s_k \le \frac{1}{2}. \tag{16}$$

hold for any $k \ge d$. For $k = d$, the initial condition (14) and the setting $\mathbf{x}_0 = \mathbf{x}_1 \cdots = \mathbf{x}_d$ in the algorithm means $s_{d+1} \le s_0^2 \le s_0 = s_d$ and $s_d = s_0 \le \frac{1}{2}$. Suppose the results of (16) hold for any $k \le k' - 1$. Then for $k = k'$, we have $s_{k'+1} \le \frac{1}{2}s_{k'}^2 + \frac{1}{2}s_{k'}s_{\tau(k';d)} \overset{(16)}{\le} s_{k'} \overset{(16)}{\le} \frac{1}{2}$, which finish the induction. Thus, we have

$$s_{k+1} \le \frac{1}{2}s_k^2 + \frac{1}{2}s_k s_{\tau(k;d)} \overset{(16)}{\le} s_k s_{\tau(k;d)}. \tag{17}$$

Then we use induction to show

$$s_{td} \le \left(\frac{1}{2}\right)^{(1+d)^{\lfloor (t+1)/2 \rfloor -1}}, \tag{18}$$

for all $t \ge 1$. For $t = 1$, we directly have $s_d = s_0 \le \frac{1}{2}$. Suppose the inequality (18) holds for $t = t'$. For $t = t' + 1$, we have

$$\begin{aligned} s_{(t'+1)d} &\overset{(16)}{\le} s_{(t'+1)d-1}s_{t'd-d} \overset{(17)}{\le} s_{(t'+1)d-2}s_{(t'-1)d}^2 \\ &\le \cdots \le s_{t'd}s_{(t'-1)d}^d \\ &\overset{(18)}{\le} \left(\frac{1}{2}\right)^{(1+d)^{\lfloor (t'+1)/2 \rfloor -1}} \left(\frac{1}{2}\right)^{(1+d)^{\lfloor t'/2 \rfloor -1}d}. \end{aligned} \tag{19}$$

If t' is an even number, we can write $t' = 2q$ and it holds that

$$\begin{aligned} s_{(t'+1)d} &\overset{(19)}{\le} \left(\frac{1}{2}\right)^{(1+d)^{q-1}+(1+d)^{q-1}d} \\ &= \left(\frac{1}{2}\right)^{(1+d)^q} = \left(\frac{1}{2}\right)^{(1+d)^{\lfloor (t'+2)/2 \rfloor -1}}. \end{aligned}$$

If t' is an odd number, we can write $t' = 2q + 1$ and it holds that

$$\begin{aligned} s_{(t'+1)d} &\overset{(19)}{\le} \left(\frac{1}{2}\right)^{(1+d)^q+(1+d)^{q-1}d} \\ &\le \left(\frac{1}{2}\right)^{(1+d)^q} = \left(\frac{1}{2}\right)^{(1+d)^{\lfloor (t'+2)/2 \rfloor -1}}. \end{aligned}$$

Above discussion finishes the induction.

For the k-th iteration, we write $k = dt + p$, where $t = \lfloor k/d \rfloor$ and $p = k\%d$. Then, we have

$$\begin{aligned} s_k &\overset{(17)}{\le} s_{k-1}s_{(t-1)d} \le s_{td}s_{(t-1)d}^p \\ &\overset{(18)}{\le} \left(\frac{1}{2}\right)^{(1+d)^{\lfloor (t+1)/2 \rfloor -1}} \left(\frac{1}{2}\right)^{(1+d)^{\lfloor t/2 \rfloor -1}p} \le \left(\frac{1}{2}\right)^{\left((1+d)^{t\%2}+p\right)(1+d)^{\lfloor t/2 \rfloor -1}}. \end{aligned}$$

Substituting the definition of s_k into above result, we obtain (15). □

Remark 3.7. We can verify the superlinear convergence of C2EDEN as follows

$$\lim_{k\to\infty}\frac{\|\nabla f(\mathbf{x}_{k+1})\|}{\|\nabla f(\mathbf{x}_k)\|} = \lim_{k\to\infty}\frac{s_{k+1}}{s_k} \overset{(17)}{\le} \lim_{k\to\infty}s_{\tau(k;d)} \overset{(15)}{=} 0.$$

Remark 3.8. Theorem 3.6 indicates the local superlinear convergence rate of C2CEDEN in strongly-convex case can be obtained by taking $M = 0$, which leads to the step of line 23 in Algorithm 1 has the closed form expression of $\mathbf{x}_{k+1} = \mathbf{H}^{-1}\mathbf{g}_k$.

4 EXPERIMENT

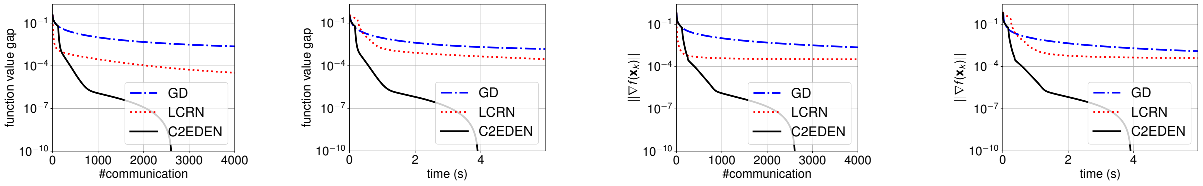
In this section, we provide numerical experiments for the proposed Communication and Computation Efficient DistributEd Newton (C2EDEN) method on regularized logistic regression. The model is formulated as the form of (1) with

$$f_i(\mathbf{x}) = \frac{1}{m_i}\sum_{j=1}^{m_i}\ln\left(1+\exp\left(-b_{ij}\mathbf{x}^\top\mathbf{a}_{ij}\right)\right) + \lambda R(\mathbf{x}), \tag{20}$$

where $\mathbf{a}_{ij} \in \mathbb{R}^d$ and $b_{ij} \in \{-1,+1\}$ are the feature and the corresponding label of the j-th sample on the i-th client, m_i denotes the number of samples on the i-th client, $R(\mathbf{x})$ is the regularization term. We set the hyperparameter $\lambda = 10^{-6}$ and the number of clients $n = 16$ and $n = 32$.

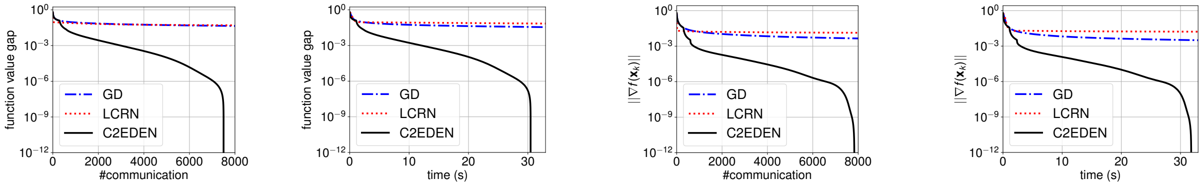
Our experiments are conducted on AMD EPYC 7H12 64-Core Processor with 512GB memory[1]. We use MPI 3.3.2 and Python 3.9 to implement the algorithms and the operating system is Ubuntu 20.04.2. We compare our the proposed C2EDEN with the baseline algorithms that require $O(d)$ communication complexity per iteration and avoid $O(d^2)$ space complexity on the client. All of the datasets we used are can be downloaded from LIBSVM repository [5].

[1] The code for our experiments can be downloaded from https://github.com/7CCLiu/C2EDEN for reproducibility.



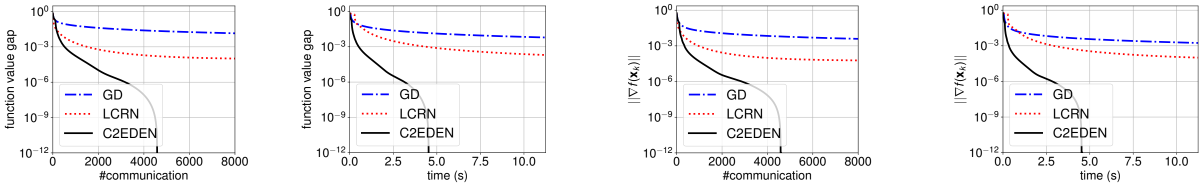
(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 2: The results of the model of nonconvex regularized logistic regression on a9a (*n*=32).



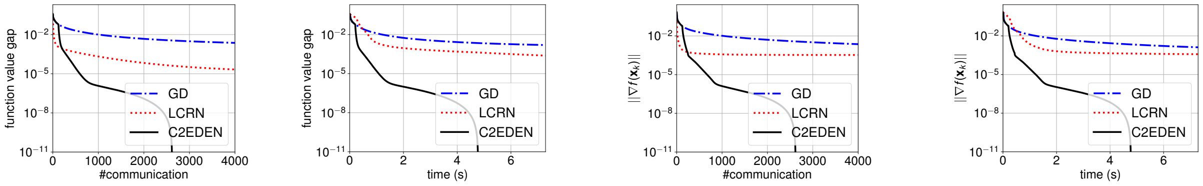
(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 3: The results of the model of nonconvex regularized logistic regression on w8a (*n*=32).



(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 4: The results of the model of nonconvex regularized logistic regression on mushrooms (*n*=32).



(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 5: The results of the model of nonconvex regularized logistic regression on a9a (*n*=16).

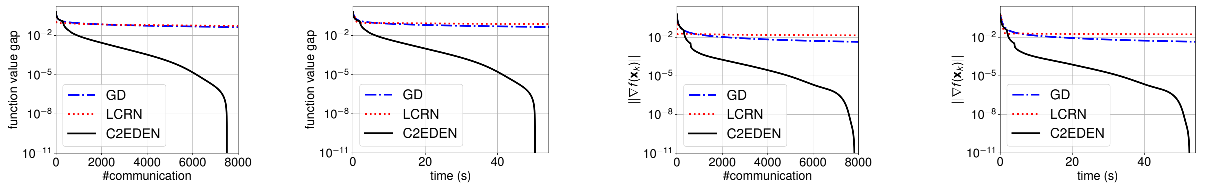
4.1 The Nonconvex Case

We consider logistic regression with the following regularization $R(\mathbf{x}) = \sum_{p=1}^{d} \frac{x_{(p)}^2}{1+x_{(p)}^2}$, where $x_{(p)}$ is the p-th coordinate of $\mathbf{x} \in \mathbb{R}^d$. Such setting leads to the objective $f(\cdot)$ be nonconvex [2].

We compare the proposed C2EDEN with the distributed gradient descent (GD) [22] and the local cubic-regularized Newton (LCRN) [8] method. We evaluate all of the algorithms on datasets “a9a” ,“w8a” and “mushrooms”. For GD, we tune the step-size from $\{10^{-1}, 10^{-2}, 10^{-3}\}$. For C2EDEN and local cubic-Newton, we tune the regularization parameter M from $\{1, 10, 100\}$.

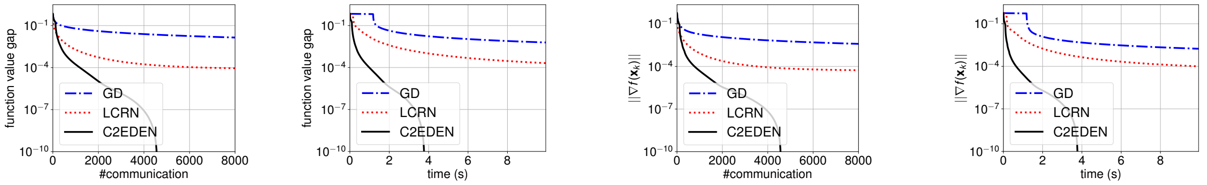
We present the results for $n = 32$ in Figure 2, 3 and 4 and the results for $n = 16$ in Figure 5, 6 and 7 by showing the comparison on the number of iterations and computational time (seconds) against the function value gap $f(\mathbf{x}_k) - \hat{f}$ and gradient norm, where $\hat{f}$ is the smallest function value appears in the iterations of all algorithms.

All the figures show that both of the function value gap and the gradient norm of C2EDEN decrease much faster than the baselines.



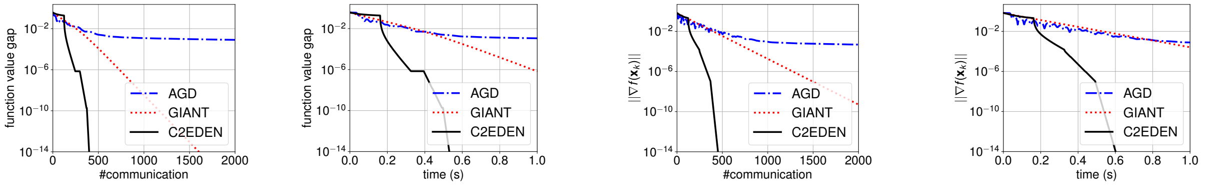
(a) #communication vs. gap (b) time (s) vs. gap (c)#communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 6: The results of the model of nonconvex regularized logistic regression on w8a (n=16).



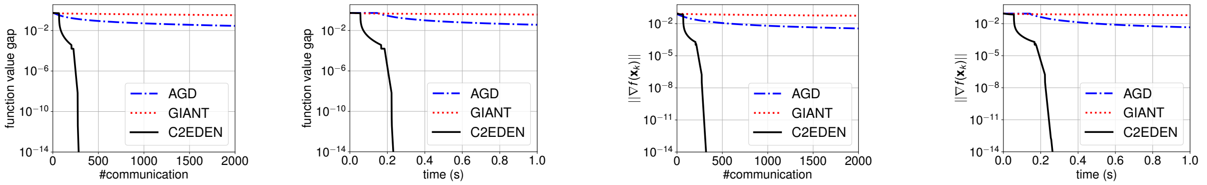
(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 7: The results of the model of nonconvex regularized logistic regression on mushrooms (n=16).



(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 8: The results of the model of ℓ_2-regularized logistic regression on a9a (n=32).



(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 9: The results of the model of ℓ_2-regularized logistic regression on a9a (n=32).

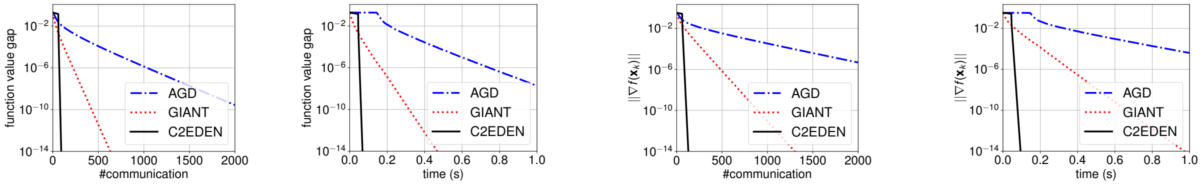
4.2 The Strongly-Convex Case

Then we consider logistic regression with the ℓ_2-regularization term $R(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|^2$, which leads to the objective be strongly-convex.

We compare the proposed C2EDEN with the distributed accelerated gradient descent (AGD) [22] and the GIANT [32]. We evaluate all of the algorithms on datasets "a9a", "phishing" and "splice". For AGD, we tune the step size form $\{10^{-1}, 10^{-2}, 10^{-3}\}$ and tune the momentum parameter from $\{0.9, 0.99, 0.999\}$. We also introduce a warm-up phrase for GIANT to achieve its region of local convergence.

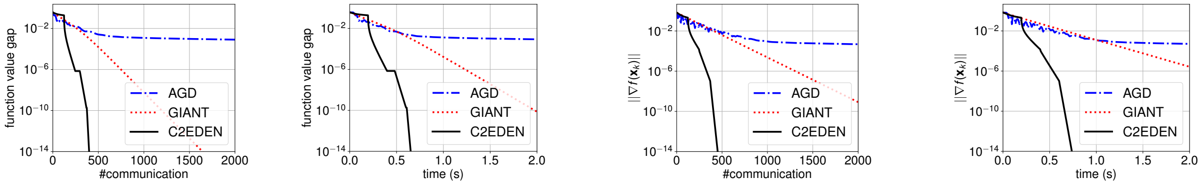
We present the results for $n = 32$ in Figure 8, 9 and 10 and the results for $n = 16$ in Figure 11, 12 and Figure 13 by showing the comparison on the number of communication rounds and computational time (seconds) against function value gap and gradient norm.

We can observe that C2EDEN performs the superlinear rate from early iterations (i.e. (c) of Figure 8 after $\nabla f(\mathbf{x}_k) \leq 1\mathrm{e}-2$) and it converges much faster than baselines.



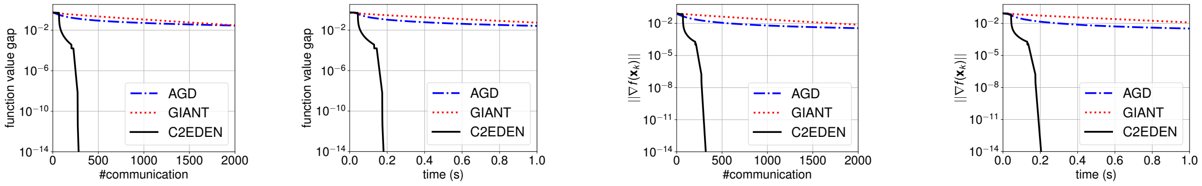
(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 10: The results of the model of ℓ_2-regularized logistic regression on splice (n=32).



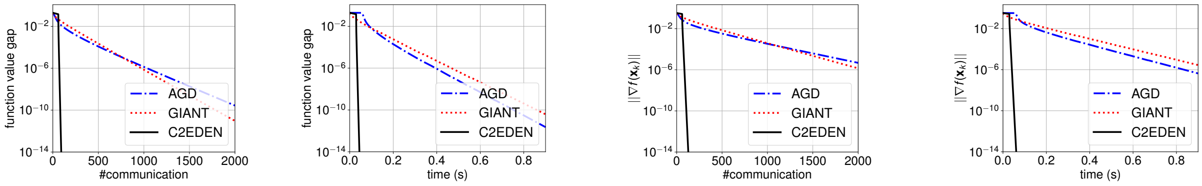
(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 11: The results of the model of ℓ_2-regularized logistic regression on a9a (n=16).



(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 12: The results of the model of ℓ_2-regularized logistic regression on phishing (n=16).



(a) #communication vs. gap (b) time (s) vs. gap (c) #communication vs. $\|\nabla f(x_k)\|$ (d) time (s) vs. $\|\nabla f(x_k)\|$

Figure 13: The results of the model of ℓ_2-regularized logistic regression on splice (n=16).

5 CONCLUSION

In this work, we propose the Communication and Computation EfficiEnt Distributed Newton (C2EDEN) method for distributed optimization. We provide a new mechanism to communicate the second-order information along with a simple yet efficient update rule. We prove the proposed method possesses the fast convergence rate like the classical second-order methods on single machine optimization problems. The empirical studies on both convex and nonconvex optimization problem also support our theoretical analysis.

For the future work, it is very interesting to generalize our ideas to the setting of decentralized optimization. It is also possible to design the stochastic variants of C2EDEN.

ACKNOWLEDGEMENTS

Luo Luo is supported by National Natural Science Foundation of China (No. 62206058) and Shanghai Sailing Program (22YF1402900). John C.S. Lui is supported in part by the RGC's GRF 14200321.

REFERENCES

[1] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

[2] Anestis Antoniadis, Irène Gijbels, and Mila Nikolova. Penalized likelihood regression for generalized linear models with non-quadratic penalties. *Annals of the Institute of Statistical Mathematics*, 63(3):585–615, 2011.

[3] Stephen Boyd. Convex optimization: from embedded real-time to large-scale distributed. In *KDD*, 2011.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

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

[6] Rixon Crane and Fred Roosta. DINGO: Distributed Newton-type method for gradient-norm optimization. *NeurIPS*, 2019.

[7] Nikita Doikov, El Mahdi Chayti, and Martin Jaggi. Second-order optimization with lazy Hessians. *arXiv preprint arXiv:2212.00781*, 2022.

[8] Avishek Ghosh, Raj Kumar Maity, Arya Mazumdar, and Kannan Ramchandran. Escaping saddle points in distributed Newton’s method with communication efficiency and Byzantine resilience. *arXiv preprint arXiv:2103.09424*, 2021.

[9] Rustem Islamov, Xun Qian, and Peter Richtárik. Distributed second order methods with fast rates and compressed communication. In *ICML*, 2021.

[10] Rustem Islamov, Xun Qian, Slavomír Hanzely, Mher Safaryan, and Peter Richtárik. Distributed Newton-type methods with communication compression and Bernoulli aggregation. *arXiv preprint arXiv:2206.03588*, 2022.

[11] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *ICML*, 2020.

[12] Jakub Konevcnỳ, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[13] Ching-pei Lee, Cong Han Lim, and Stephen J. Wright. A distributed quasi-Newton algorithm for empirical risk minimization with nonsmooth regularization. In *KDD*, 2018.

[14] Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G. Andersen, and Alexander Smola. Parameter server for distributed machine learning. In *Big learning workshop on NIPS*, 2013.

[15] Mu Li, David G. Andersen, Alexander J. Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *NIPS*, 2014.

[16] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of FedAvg on non-iid data. In *ICLR*, 2020.

[17] Sulin Liu, Sinno Jialin Pan, and Qirong Ho. Distributed multi-task relationship learning. In *KDD*, 2017.

[18] Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich, and Peter Richtárik. ProxSkip: Yes! local gradient steps provably lead to communication acceleration! finally! In *ICML*, 2022.

[19] Aritra Mitra, Rayana Jaafar, George J. Pappas, and Hamed Hassani. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. In *NeurIPS*, 2021.

[20] Daniel K. Molzahn, Florian Dörfler, Henrik Sandberg, Steven H. Low, Sambuddha Chakrabarti, Ross Baldick, and Javad Lavaei. A survey of distributed optimization and control algorithms for electric power systems. *IEEE Transactions on Smart Grid*, 8(6):2941–2962, 2017.

[21] Yurii Nesterov. Accelerating the cubic regularization of Newton’s method on convex problems. *Mathematical Programming*, 112(1):159–181, 2008.

[22] Yurii Nesterov. *Lectures on convex optimization*. Springer, 2018.

[23] Yurii Nesterov and Boris T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[24] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer, 1999.

[25] Sashank J. Reddi, Jakub Konevcnỳ, Peter Richtárik, Barnabás Póczós, and Alex Smola. AIDE: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

[26] Mher Safaryan, Rustem Islamov, Xun Qian, and Peter Richtárik. FedNL: Making newton-type methods applicable to federated learning. In *ICML*, 2022.

[27] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate Newton-type method. In *ICML*, 2014.

[28] Virginia Smith, Simone Forte, Michael I. Jordan, and Martin Jaggi. L1-regularized distributed optimization: A communication-efficient primal-dual framework. *arXiv preprint arXiv:1512.04011*, 2015.

[29] Saeed Soori, Konstantin Mishchenko, Aryan Mokhtari, Maryam Mehri Dehnavi, and Mert Gurbuzbalaban. DAve-QN: A distributed averaged quasi-Newton method with local superlinear convergence rate. In *AISTATS*, 2020.

[30] César A. Uribe and Ali Jadbabaie. A distributed cubic-regularized Newton method for smooth convex optimization over networks. *arXiv preprint arXiv:2007.03562*, 2020.

[31] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. A survey on distributed machine learning. *ACM Computing Surveys*, 53(2):1–33, 2020.

[32] Shusen Wang, Fred Roosta, Peng Xu, and Michael W. Mahoney. GIANT: Globally improved approximate Newton method for distributed optimization. In *NeurIPS*, 2018.

[33] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *NeurIPS*, 2018.

[34] Ling Yan, Wu-Jun Li, Gui-Rong Xue, and Dingyi Han. Coupled group lasso for web-scale CTR prediction in display advertising. In *ICML*, 2014.

[35] Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl H. Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 47:278–305, 2019.

[36] Haishan Ye, Chaoyang He, and Xiangyu Chang. Accelerated distributed approximate Newton method. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[37] Xiao-Tong Yuan and Ping Li. On convergence of distributed approximate Newton methods: Globalization, sharper bounds and beyond. *Journal of Machine Learning Research*, 21(206):1–51, 2020.

[38] Yuchen Zhang and Xiao Lin. DiSCO: Distributed optimization for self-concordant empirical loss. In *ICML*, 2015.

A VERIFYING ASSUMPTION 2.1 FOR LOGISTIC REGRESSION

We can verify the objective function in problem (20) satisfies Assumption 2.1. The triangle inequality means we only need to prove that both $g(\mathbf{x}) = \ln(1 + \exp(-\mathbf{a}^\top \mathbf{x} b))$ and $R(\mathbf{x})$ have Lipschitz continuous Hessians for given $\mathbf{a} \in \mathbb{R}^d$ and $b \in \{1, -1\}$. Here are the details.

A.1 Lipschitz Continuity of $\nabla^2 g(\cdot)$

The Hessian of $g(\mathbf{x})$ can be written as

$$\nabla^2 g(\mathbf{x}) = \frac{\exp(-\mathbf{a}^\top \mathbf{x} b)\mathbf{a}\mathbf{a}^\top}{(1 + \exp(-\mathbf{a}^\top \mathbf{x} b))^2} = \sigma(\mathbf{a}^\top \mathbf{x} b)(1 - \sigma(\mathbf{a}^\top \mathbf{x} b))\mathbf{a}\mathbf{a}^\top,$$

where we use the fact that $b^2 = 1$ and denote $\sigma(z) = 1/(1 + \exp(-z))$. It is easy to check that $\sigma(z) \in (0, 1)$ for any $z \in \mathbb{R}$. We first show that $\sigma(\cdot)$ is Lipschitz continuous. Without loss of generality, we suppose $z' > z$. Then we have

$$\begin{aligned}|\sigma(z) - \sigma(z')| &= \left|\frac{\exp(-z) - \exp(-z')}{(1 + \exp(-z))(1 + \exp(-z'))}\right| \\ &= \left|\frac{\exp(-z)(1 - \exp(z - z'))}{(1 + \exp(-z))(1 + \exp(-z'))}\right| \leq 1 - \exp(z - z') \\ &\leq z' - z,\end{aligned}$$

where the last step use the fact $\exp(s) \geq 1 + s$ for any $s \in \mathbb{R}$. Thus, we achieve

$$\begin{aligned}&\|\nabla^2 g(\mathbf{x}) - \nabla^2 g(\mathbf{x}')\| \\ \leq&\|\sigma(\mathbf{a}^\top \mathbf{x} b)(1 - \sigma(\mathbf{a}^\top \mathbf{x} b)) - \sigma(\mathbf{a}^\top \mathbf{x}' b)(1 - \sigma(\mathbf{a}^\top \mathbf{x}' b))\|\|\mathbf{a}\mathbf{a}^\top\| \\ \leq&\|2(\sigma(\mathbf{a}^\top \mathbf{x} b) - \sigma(\mathbf{a}^\top \mathbf{x}' b))\|\|\mathbf{a}^\top \mathbf{a}\| \\ \leq&2\|\mathbf{a}\|\|\mathbf{x} - \mathbf{x}'\|\|\mathbf{a}^\top \mathbf{a}\| \\ =&2\|\mathbf{a}\|^3\|\mathbf{x} - \mathbf{x}'\|,\end{aligned}$$

for any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, where the last step is based on the Lipschitz continuity of $\sigma(\cdot)$.

A.2 Lipschitz continuity of $\nabla^2 R(\cdot)$

- For the nonconvex case, we have $R(\mathbf{x}) = \sum_{p=1}^d x_{(p)}^2/(1 + x_{(p)}^2)$. We denote $r(z) = z^2/(1 + z^2)$, then its third order derivative $\dddot{r}(z)$ holds that

  $$\begin{aligned}|\dddot{r}(z)| &= \left\|\frac{24z(z^2 - 1)}{(z^2 + 1)^4}\right\| \\ &\leq 24\left\|\frac{z}{(z^2 + 1)^3}\right\| \leq 24\left\|\frac{z}{(z^2 + 1)}\right\| \leq 12,\end{aligned}$$

  where the last inequality comes from the fact that $|z/(1+z^2)| \leq 1/2$ for any $z \in \mathbb{R}$. Thus the second-order derivative $\ddot{r}(\cdot)$ holds that

  $$|\ddot{r}(z) - \ddot{r}(z')| \leq |\dddot{r}(\mathbf{x}i)||z - z'| \leq 12|z - z'|.$$

  Then we achieve

  $$\begin{aligned}&\|\nabla^2 R(\mathbf{x}) - \nabla^2 R(\mathbf{x}')\| \\ \leq& \max_{p=1,\ldots,d} |\ddot{r}(\mathbf{x}_{(p)}) - \ddot{r}(\mathbf{x}'_{(p)})| \\ \leq&12 \max_{p=1,\ldots,d} |\mathbf{x}'_{(p)} - \mathbf{x}_{(p)}| \leq 12\|\mathbf{x} - \mathbf{x}'\|.\end{aligned}$$

- For the strongly-convex case, we have $R(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|^2$. It is easy to verify that $\nabla^2 R(\mathbf{x}) = \mathbf{I}$, which means $\|\nabla^2 R(\mathbf{x}) - \nabla^2 R(\mathbf{x}')\| = 0$.