Harvesting Curvatures for Communication-Efficient Distributed Optimization

Diogo Cardoso*†, Boyue Li*, Yuejie Chi*, João Xavier†
*Department of Electrical and Computer Engineering, Carnegie Mellon University, USA
†Institute for Systems and Robotics, IST – University of Lisbon, Portugal

***Abstract*—This paper considers the design of distributed optimization algorithms in the server-client setting, with the goal of achieving better communication efficiency, a crucial desiderata in modern distributed machine learning applications. One popular approach to reduce communication is to impose more computation between communication rounds, by letting each client optimize an approximation of the global objective function—constructed using the local objective function and aggregated history information—and exchange the updates with the parameter server for global consensus. In particular, judiciously incorporating second-order information in the local approximation often leads to better communication efficiency when the problem is ill-conditioned. However, existing methods construct the second-order correction term in an isotropic manner, without taking into account the curvature information of the global objective function accumulated over the course of the algorithm. This paper proposes a novel algorithm that refines this idea by constructing a second-order correction term using a BFGS-style update formula, where the kernel matrix is updated recursively using only history gradients to harvest curvature information for accelerating convergence. Numerical experiments demonstrate improved communication efficiency over competitive baseline algorithms on both synthetic and real datasets.**

***Index Terms*—distributed optimization, communication efficiency, server-client model, quasi-Newton methods.**

I. Introduction

Distributed optimization takes a prominent role in modern machine learning applications, due to its ability to process a huge amount of data in parallel while maintaining privacy of the clients. For instance, the emerging paradigm of federated learning (FL) [1] enables training of machine learning models at mobile devices in a distributed manner, through interaction with a parameter server, without accessing users' private data directly. In this paper, we consider the following optimization problem in a server-client model:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \coloneqq \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x}), \tag{1}$$

The work of D. Cardoso was supported by Fundação para a Ciência e Tecnologia through scholarship PD/BD/143146/2019, and the Information and Communication Technologies Institute at Carnegie Mellon University. The work of B. Li and Y. Chi was supported in part by U.S. National Science Foundation under CCF-2007911, Air Force Research Laboratory under FA8750-20-2-0504, and Wei Shen and Xuehong Zhang Presidential Fellowship from Carnegie Mellon University. The work of J. Xavier was supported in part by the Fundação para a Ciência e Tecnologia, Portugal, through the Project LARSyS, under Project FCT Project UIDB/50009/2020.

where N denotes the number of clients or agents, and each client i can access a twice-differentiable convex objective function $f_i : \mathbb{R}^d \to \mathbb{R}$, for $i \in \{1, \dots, N\}$. In particular, we aim to solve (1) where the clients can only communicate with the central server to exchange parameters and gradients, but not the local data or higher-order information directly.

Roughly speaking, distributed optimization algorithms generally alternate between global communication and local computation, where the clients send their local parameters to the server to obtain an averaged global parameter, which is then sent back to the client for refinements using the local data. To achieve better communication efficiency, a popular scheme in practice is to impose more computation on the clients at each iteration to reduce the communication frequency and enable faster convergence. This includes methods that perform multiple local gradient update steps before communicating [1]–[5], as well as those that solve a more sophisticated local approximation to the global objective function [6]–[10]. In this paper, we follow the second approach by devising better local approximations.

In order to enable faster convergence, it is desirable for the client to construct an approximation of the global objective function—based on its local information—as accurately as possible. While exact Hessian information can be hard to compute and communicate, it has been observed that judiciously incorporating, even implicitly, second-order information in the local approximation often leads to better communication efficiency when the problem is ill-conditioned. This gives rise to the consideration of local problems that take the form of second-order approximations of the global objective function at the current parameter estimate. However, existing methods, such as Distributed Approximate NEwton-type method (DANE) [8], construct the second-order correction term in an isotropic manner, without taking into account the curvature information of the global objective function accumulated over the course of the algorithm. Consequently, DANE suffers from slow convergence or even divergence when the local data are highly heterogenous, due to e.g., small sample sizes.

Inspired by the well-known BFGS algorithm [11], this paper proposes a novel algorithm, called Curvature-Enhanced Distributed Approximate NEwton-type method (CEDANE), that optimizes a refined local approximation by constructing the second-order correction term using a BFGS-style [11] update formula, where the kernel matrix is updated recursively using

only history gradients to harvest more curvature information of the global objective function to accelerate convergence. Importantly, the kernel matrix is non-isotropic, and mimics the global Hessian matrix at the current global parameter using only gradient information, with the hope of further improving the communication efficiency especially in the face of heterogeneous or limited local data. Numerical experiments indeed demonstrate improved communication efficiency over competitive baseline algorithms on both synthetic and real datasets.

The rest of the paper is organized as follows. Section II reviews preliminaries and motivates the development of CEDANE. Section III presents the proposed CEDANE algorithm. Section IV shows numerical experiments on both synthetic and real datasets, and we conclude in Section V. Lower-case and upper-case boldface letters denote vectors and matrices, respectively. Let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{y}^\top \boldsymbol{x}$ denote the standard inner-product in $\mathbb{R}^d$. For a vector $\boldsymbol{x} \in \mathbb{R}^d$, $\|\boldsymbol{x}\|_2^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle$; for a positive-definite matrix $\boldsymbol{A}$, $\|\boldsymbol{x}\|_{\boldsymbol{A}}^2 = \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x}$.

II. Preliminaries

This section reviews two iterative distributed optimization algorithms, DANE and BFGS, that inspired the proposed CEDANE algorithm.

A. DANE algorithm

The Distributed Approximate Newton-type Method (DANE) [8], [9] follows the following general recipe to solve (1) at each iteration t, which takes two rounds of communication:

1) *communication:* the clients send their local parameters $\boldsymbol{x}_i^{(t)}$ to the server to compute the global parameter, $\overline{\boldsymbol{x}}^{(t)} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_i^{(t)}$, then the server sends back the global parameter $\overline{\boldsymbol{x}}^{(t)}$ to the clients;
2) *communication:* the clients send the local gradient $\nabla f_i(\overline{\boldsymbol{x}}^{(t)})$ at the global parameter $\overline{\boldsymbol{x}}^{(t)}$ to the server to compute the global gradient, $\nabla f(\overline{\boldsymbol{x}}^{(t)}) = \frac{1}{N}\sum_{i=1}^N \nabla f_i(\overline{\boldsymbol{x}}^{(t)})$, then the server sends the global gradient back to the clients $\nabla f(\overline{\boldsymbol{x}}^{(t)})$;
3) *computation:* the clients perform local updates, i.e. by solving a local optimization problem which can be regarded an approximation to (1), based on the global parameter $\overline{\boldsymbol{x}}^{(t)}$, global gradient $\nabla f(\overline{\boldsymbol{x}}^{(t)})$, and the local objective function $f_i(\boldsymbol{x})$.

In particular, DANE asks each client to solve a local optimization problem given as

$$\boldsymbol{x}_i^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{x}} \Big\{ f(\overline{\boldsymbol{x}}^{(t)}) + \langle \nabla f(\overline{\boldsymbol{x}}^{(t)}), \boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)} \rangle + \frac{1}{\eta} D_{h_i}(\boldsymbol{x}, \overline{\boldsymbol{x}}^{(t)}) \Big\}, \tag{2}$$

where $\eta > 0$,

$$h_i(\boldsymbol{x}) = f_i(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{x}\|_2^2$$

denotes the regularized local objective function with $\mu \geq 0$, and $D_{h_i}(\boldsymbol{x}, \overline{\boldsymbol{x}}^{(t)})$ denotes the Bregman divergence associated with h_i between $\boldsymbol{x}$ and $\overline{\boldsymbol{x}}^{(t)}$, i.e.

$$D_{h_i}(\boldsymbol{x}, \overline{\boldsymbol{x}}^{(t)}) = h_i(\boldsymbol{x}) - h_i(\overline{\boldsymbol{x}}^{(t)}) - \langle \nabla h_i(\overline{\boldsymbol{x}}^{(t)}), \boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)} \rangle.$$

Therefore, the local problem (2) can be viewed as a second-order approximation of the global objective function. In addition, by rearrangements and dropping constant terms, (2) can be equivalently written as the following equation, which is more practical to implement:

$$\boldsymbol{x}_i^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{x}} \Big\{ f_i(\boldsymbol{x}) - \langle \nabla f_i(\overline{\boldsymbol{x}}^{(t)}) - \eta \nabla f(\overline{\boldsymbol{x}}^{(t)}), \boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)} \rangle + \frac{\mu}{2}\|\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\|_2^2 \Big\}. \tag{3}$$

When the local objective functions (1) are quadratic, (3) can be solved analytically by setting the gradient to $\mathbf{0}$, resulting in the following update formula

$$\overline{\boldsymbol{x}}^{(t+1)} = \overline{\boldsymbol{x}}^{(t)} - \eta \underbrace{\left(\frac{1}{N}\sum_{i=1}^N (\nabla^2 f_i(\overline{\boldsymbol{x}}^{(t)}) + \mu \boldsymbol{I})^{-1} \right)}_{\text{Approximates inverse Hessian matrix}} \nabla f(\overline{\boldsymbol{x}}^{(t)}),$$

where the average of the inverse regularized Hessian matrices can be regarded as approximating the inverse of the global Hessian matrix, thus performing an approximate Newton step for the global objective function with η as the step size. Therefore, while only communicating parameters and gradients, DANE implicitly uses some second-order information. Here, the parameter μ can be viewed as a regularizer that controls the well-posedness of the local problem, where increasing μ helps to stabilize the convergence when the local problems are ill-conditioned or heterogenous at an expense of slower convergence.

B. BFGS algorithm

The BFGS algorithm [11] is a quasi-Newton algorithm designed for unconstrained optimization problems, which keeps track of an estimate of the inverse Hessian matrix, $\big(\boldsymbol{B}^{(t)}\big)^{-1}$ for faster convergence without explicitly computing the Hessian matrix. It is straightforward to adapt the BFGS algorithm to the server-client setting, where the parameter is updated at each iteration according to

$$\overline{\boldsymbol{x}}^{(t+1)} = \overline{\boldsymbol{x}}^{(t)} - \eta \big(\boldsymbol{B}^{(t)}\big)^{-1} \nabla f(\overline{\boldsymbol{x}}^{(t)}),$$

where $\eta > 0$ is the step size.[1] In addition, the matrix $\big(\boldsymbol{B}^{(t)}\big)^{-1}$ is updated recursively by

$$\big(\boldsymbol{B}^{(t+1)}\big)^{-1} = \boldsymbol{\Delta}^{(t)} \big(\boldsymbol{B}^{(t)}\big)^{-1} \boldsymbol{\Delta}^{(t)\top} + \frac{\boldsymbol{s}^{(t)} \boldsymbol{s}^{(t)\top}}{\boldsymbol{y}^{(t)\top} \boldsymbol{s}^{(t)}}, \tag{4a}$$

where $\boldsymbol{s}^{(t)}$, $\boldsymbol{y}^{(t)}$, $\boldsymbol{\Delta}^{(t)}$ are defined accordingly as

$$\boldsymbol{s}^{(t)} = \overline{\boldsymbol{x}}^{(t+1)} - \overline{\boldsymbol{x}}^{(t)}, \tag{4b}$$

$$\boldsymbol{y}^{(t)} = \nabla f(\overline{\boldsymbol{x}}^{(t+1)}) - \nabla f(\overline{\boldsymbol{x}}^{(t)}), \tag{4c}$$

$$\boldsymbol{\Delta}^{(t)} = \boldsymbol{I} - \frac{\boldsymbol{s}^{(t)} \boldsymbol{y}^{(t)\top}}{\boldsymbol{y}^{(t)\top} \boldsymbol{s}^{(t)}}. \tag{4d}$$

The estimates $\boldsymbol{B}^{(t)}$ are guaranteed to preserve symmetric positive-definiteness if the global objective function is strongly

[1] Here, we set the step size η as a fixed parameter, rather than performing line search as in contrast to canonical BFGS to avoid the communication overhead in implementing the line search procedure in a distributed environment.

convex and the algorithm starts from a positive-definite initialization $\boldsymbol{B}^{(0)}$. Note that the BFGS updates (4) use only the global parameter and the global gradient, thus can be implemented either at the server end or the client end.

III. Proposed CEDANE Algorithm

This section develops the proposed algorithm, dubbed the Curvature-Enhanced Distributed Approximate NEwton-type method (CEDANE), which is detailed in Algorithm 1. CEDANE follows a similar recipe as DANE, but differs in the designs of the local optimization problem solved at each client. The key idea of CEDANE is to incorporate more curvature information of the global objective function in the local optimization problem without increasing the communication overhead, a strategy that aims to facilitate a faster convergence especially when the local problems are ill-conditioned or heterogeneous.

Inspired by BFGS, at each iteration, CEDANE aims to solve the following local optimization problem at each client

$$\boldsymbol{x}_i^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{x}} \Big\{ f(\overline{\boldsymbol{x}}^{(t)}) + \langle \nabla f(\overline{\boldsymbol{x}}^{(t)}), \boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)} \rangle + \frac{1}{\eta} D_{g_i^{(t)}}(\boldsymbol{x}, \overline{\boldsymbol{x}}^{(t)}) \Big\}, \tag{5}$$

where $D_{g_i^{(t)}}(\boldsymbol{x}, \overline{\boldsymbol{x}}^{(t)})$ denotes the Bregman divergence associated with $g_i^{(t)}$ between $\boldsymbol{x}$ and $\overline{\boldsymbol{x}}^{(t)}$, and $g_i^{(t)}$ is an iteration-varying regularized local objective function given by

$$g_i^{(t)}(\boldsymbol{x}) = f_i(\boldsymbol{x}) + \frac{\mu}{2} \|\boldsymbol{x}\|^2_{\boldsymbol{B}^{(t)}}. \tag{6}$$

Here, $\boldsymbol{B}^{(t)}$ is an approximation to the global Hessian matrix $\nabla^2 f(\overline{\boldsymbol{x}}^{(t)})$ obtained through performing the BFGS-style updates at each client following (4). Plugging in the expression of $D_{g_i^{(t)}}(\boldsymbol{x}, \overline{\boldsymbol{x}}^{(t)})$ into (5), we can rewrite the local problem as

$$\boldsymbol{x}_i^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{x}} \Big\{ f_i(\boldsymbol{x}) - \langle \nabla f_i(\overline{\boldsymbol{x}}^{(t)}) - \eta \nabla f(\overline{\boldsymbol{x}}^{(t)}), \boldsymbol{x} \rangle + \frac{\mu}{2} \|\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\|^2_{\boldsymbol{B}^{(t)}} \Big\}, \tag{7}$$

where the quadratic term is now kernelized by the approximation to the global Hessian matrix $\boldsymbol{B}^{(t)}$, in sharp contrast to DANE where the quadratic term is isotropic (cf. (3)). Therefore, CEDANE incorporates the curvature information of the global objective function without computing nor transmitting the Hessian matrices explicitly.

Again, when the local objective functions are quadratic, solving (7) analytically leads to the following update formula

$$\overline{\boldsymbol{x}}^{(t+1)} = \overline{\boldsymbol{x}}^{(t)} - \eta \underbrace{\left(\frac{1}{N} \sum_{i=1}^{N} \left(\nabla^2 f_i(\overline{\boldsymbol{x}}^{(t)}) + \mu \boldsymbol{B}^{(t)} \right)^{-1} \right)}_{\text{Approximates inverse Hessian matrix}} \nabla f(\overline{\boldsymbol{x}}^{(t)}),$$

where it is postulated that the incorporation of $\boldsymbol{B}^{(t)}$ in CEDANE leads to a better approximation to the inverse of the global Hessian matrix when the local objective functions f_i's are more heterogeneous.

Algorithm 1 CEDANE

1: **Input:** step size $\eta > 0$, regularization $\mu \geq 0$.
2: **Initialization:** $\boldsymbol{x}_i^{(0)}, \boldsymbol{B}^{(0)} = \boldsymbol{I}, \forall i \in \{1, \ldots, N\}$.
3: **for** $t = 0, 1, 2, \ldots$ **do**
4: **for Clients** $1 \leq i \leq N$ in parallel **do**
5: Send $\nabla f_i(\overline{\boldsymbol{x}}^{(t)})$ to the server, which sends back $\nabla f(\overline{\boldsymbol{x}}^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\overline{\boldsymbol{x}}^{(t)})$.
6: Solve the local optimization problem:
$$\boldsymbol{x}_i^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{x}} \Big\{ f_i(\boldsymbol{x}) - \langle \nabla f_i(\overline{\boldsymbol{x}}^{(t)}) - \eta \nabla f(\overline{\boldsymbol{x}}^{(t)}), \boldsymbol{x} \rangle + \frac{\mu}{2} \|\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\|^2_{\boldsymbol{B}^{(t)}} \Big\}.$$
7: Send $\boldsymbol{x}_i^{(t+1)}$ to the server, which sends back $\overline{\boldsymbol{x}}^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i^{(t+1)}$.
8: Update the BFGS matrix $\boldsymbol{B}^{(t)}$ according to (4).
9: **end for**
10: **end for**

IV. Numerical Experiments

In this section, we evaluate the empirical performance of CEDANE for linear regression using synthetic data, and logistic regression using the Gisette dataset [12], both of which are strongly convex problems. The proposed CEDANE algorithm is compared against several competitive baselines including DANE [8], ADMM [13], and BFGS.

We compare the communication efficiency of different algorithms by analyzing the number of communication rounds needed to achieve certain optimality gap to the global minimum $\boldsymbol{x}^\star$, defined as $\big(f(\overline{\boldsymbol{x}}^{(t)}) - f(\boldsymbol{x}^\star)\big)/f(\boldsymbol{x}^\star)$. Throughout, we use m to denote the number of data points at each client, κ to denote the condition number of the global objective function. The step size η is set to 1 for both DANE and CEDANE, and all other hyper-parameters are best-tuned by hand. The experiments are based on code from [14] and can be found at https://github.com/diogo-mcardoso/cedane.

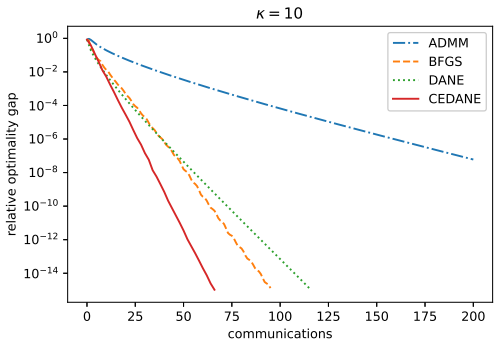
A. Linear regression

The local objective function is defined as

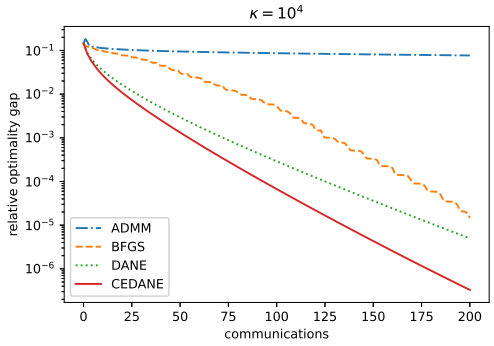
$$f_i(\boldsymbol{x}) = \frac{1}{2m} \|\boldsymbol{A}_i \boldsymbol{x} - \boldsymbol{b}_i\|_2^2,$$

for $\boldsymbol{A}_i \in \mathbb{R}^{m \times d}$, $\boldsymbol{b}_i \in \mathbb{R}^m$, $d = 200$ and $m = 80$. Here, $\boldsymbol{b}_i$ is generated according to $\boldsymbol{b}_i = \boldsymbol{A}_i \boldsymbol{x}^\natural + \boldsymbol{\epsilon}_i$, with $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $\boldsymbol{x}^\natural \in \mathbb{R}^d$ is the parameter vector. Each row of $\boldsymbol{A}_i$ is sampled i.i.d. from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a diagonal matrix and $\boldsymbol{\Sigma}_{ii} = i^{-\log_d \kappa}$, where $\kappa \geq 1$ is the chosen condition number of the problem.

Figure 1 shows the relative optimality gap against communication rounds for $\kappa = 10$ and $\kappa = 10^4$ for various algorithms. It can be seen that CEDANE outperforms DANE and BFGS in both cases, requiring smaller amount of communication to reach the same accuracy, while ADMM significantly lags behind. On the other hand, BFGS, given enough communications, outperforms DANE even without

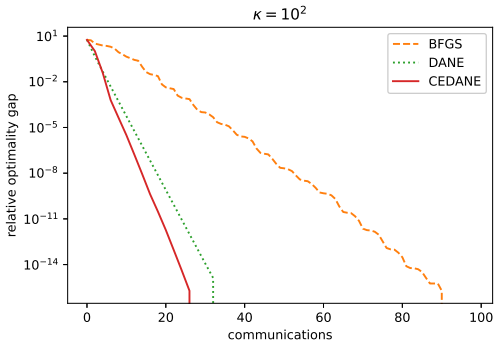


(a) $\kappa = 10$

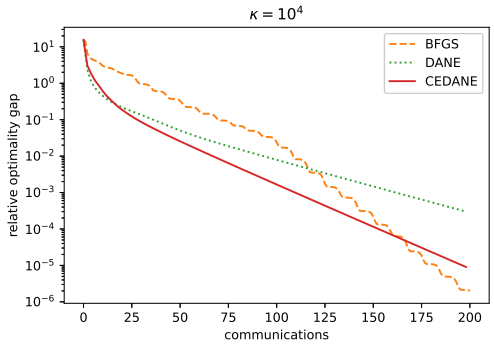


(b) $\kappa = 10^4$

Fig. 1: The relative optimality gap vs. communication rounds for linear regression on synthetic datasets. The left and right panels show the results when condition number $\kappa = 10$ and $\kappa = 10^4$, respectively.



(a) $\kappa = 10^2$



(b) $\kappa = 10^4$

Fig. 2: The relative optimality gap vs. communication rounds for logistic regression on the Gisette dataset with the condition number $\kappa = 10^2$ and $\kappa = 10^4$, respectively.

resorting to a line-search procedure to choose the step size. CEDANE seems to work in a hybrid regime between DANE and BFGS, where it explores local similarity and at the same time, harvests important curvature information of the global objective information to tackle the local-to-global dissimilarity.

B. Logistic regression

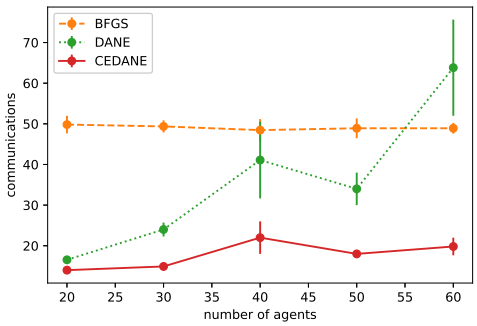
The local objective function is defined as

$$f_i(\boldsymbol{x}) = -\frac{1}{m}\sum_{j=1}^{m} \log\left(\sigma(b_{ij}\boldsymbol{x}^\top \mathbf{a}_{ij})\right) + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2,$$

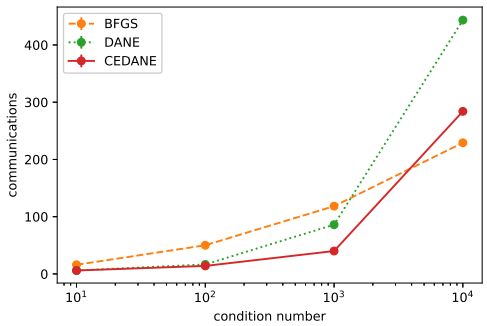
where $(\mathbf{a}_{ij}, b_{ij})$ is the j-th training sample at client i, $\lambda > 0$ is the regularization parameter which controls the condition number, and $\sigma(\cdot)$ is the sigmoid function defined as $\sigma(t) = \frac{1}{1+\exp(-t)}$. The Gisette dataset has 6000 data points equally divided in two different classes, where each data sample has $d = 5000$. We append 1 to the vector $\mathbf{a}_{ij}$ to handle the bias terms. The conditioning of the problem is controlled by parameter λ. The dataset is normalized such that the eigenvalues of the Hessian matrix are upper-bounded by $\lambda+1$ and lower-bounded by λ, and the condition number for the problem is taken approximately to be $\kappa = \frac{\lambda+1}{\lambda}$.

We first evaluate the convergence rate with different conditioning numbers, where Figure 2 shows the relative optimality gap against communication rounds when $\kappa = 10^2$ and $\kappa = 10^4$. CEDANE again outperforms DANE and BFGS for $\kappa = 10^2$, but when $\kappa = 10^4$, we can see that CEDANE converges faster only for the first 150 communications, after which point BFGS harvests enough curvature information to achieve faster convergence. Nevertheless, CEDANE might still be advantageous in terms of communication efficiency when the amount of possible communications is limited or the desired accuracy is moderate.

Next, we analyze the effect on the communication rounds when varying the number of agents and local sample size. Figure 3a shows the number of communication rounds till reach a relative optimality gap of 10^{-7} with different numbers of clients. Because the total sample size is fixed for Gisette dataset, increasing the number of clients decreases the size of local data, which in turn increases the heterogeneity or dissimilarity between the local data. We can see that the performance of DANE is significantly impacted by decreasing the size of the local data because the Hessian matrices are no longer similar to the global Hessian matrix as required by [8]. As expected, BFGS is not impacted by varying the number of clients as it estimates the global Hessian and is less impacted by local heterogeneity. However, by incorporating the BFGS-style approximation in its local problem, CEDANE is barely affected by the reduced local sample size, corroborating the



(a) varying the number of clients



(b) varying the condition number

Fig. 3: Average number of communications needed to achieve 10^{-7} relative optimality gap on logistic regression using the Gisette dataset, averaged over 10 independent runs with random partitions of the dataset and initialization. The left and right panels show the results for (a) different number of clients with $\kappa = 10^2$ and (b) different condition numbers with 20 clients, respectively. Here, the error bars represent the standard deviation, which are less visible on the right panel since they are two orders of magnitude lower than the range of communication rounds.

hypothesis that CEDANE is more robust to heterogeneous local data.

Figure 3b shows the number of communication rounds till reaching a relative optimality gap of 10^{-7} with different conditioning numbers with 20 clients. It can be seen that all algorithms are impacted when the condition number increases. CEDANE is still more robust then DANE and BFGS for most of the conditions tested, being slightly worse than BFGS when $\kappa = 10^4$, similar to previously seen in Figure 2b. It is important to note that the results shown in Figure 3b are done with 20 clients, coinciding with the scenario when DANE has a performance closer to CEDANE according to Figure 3a. A larger gap in performance would be expected for a greater number of clients.

V. Conclusion

In this paper, we presented a novel communication-efficient algorithm for the server-client setting, CEDANE, which constructs a local approximation to the global objective function using non-isotropic second-order curvature information constructed from BFGS-style updates. Compared to other distributed algorithms such as DANE [8] and BFGS, CEDANE achieves faster convergence when the local data are more heterogeneous with the same amount of communication per iteration, which is supported by experiments. However, CEDANE has higher memory requirements because each client has to store the approximate non-isotropic Hessian matrix. It might be possible to improve CEDANE's memory usage by resorting to L-BFGS [11], which we leave for future work. Additional future directions include generalizing the design of CEDANE to the fully-decentralized setting without a parameter server [14], [15], as well as the privacy-preserving setting [16].

References

[1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

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

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[4] S. U. Stich, "Local SGD converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.

[5] S. Cen, H. Zhang, Y. Chi, W. Chen, and T.-Y. Liu, "Convergence of distributed stochastic variance reduced methods without sampling extra data," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3976–3989, 2020.

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

[7] V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi, "Cocoa: A general framework for communication-efficient distributed optimization," *Journal of Machine Learning Research*, vol. 18, p. 230, 2018.

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

[9] J. Fan, Y. Guo, and K. Wang, "Communication-efficient accurate statistical estimation," *Journal of the American Statistical Association*, pp. 1–11, 2021.

[10] S. Wang, F. Roosta, P. Xu, and M. W. Mahoney, "GIANT: Globally improved approximate newton method for distributed optimization," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[11] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[12] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," *Advances in neural information processing systems*, vol. 17, 2004.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[14] B. Li, S. Cen, Y. Chen, and Y. Chi, "Communication-efficient distributed optimization in networks with gradient tracking and variance reduction," *Journal of Machine Learning Research*, vol. 21, pp. 1–51, 2020.

[15] B. Li, Z. Li, and Y. Chi, "DESTRESS: Computation-optimal and communication-efficient decentralized nonconvex finite-sum optimization," *SIAM Journal on Mathematics of Data Science*, vol. 4, no. 3, pp. 1031–1051, 2022.

[16] Z. Li, H. Zhao, B. Li, and Y. Chi, "SoteriaFL: A unified framework for private federated learning with communication compression," *arXiv preprint arXiv:2206.09888*, 2022.