Communication Efficient Framework for Decentralized Machine Learning

Anis Elgabli[†], Jihong Park[†], Amrit S. Bedi[‡], Mehdi Bennis[†], and Vaneet Aggarwal[*]
[†] University of Oulu, Finland [‡] Army Research Lab, USA [*] Purdue University, USA

***Abstract*—In this paper, we propose a fast, privacy-aware, and communication-efficient decentralized framework to solve the distributed machine learning (DML) problem. The proposed algorithm is based on the Alternating Direction Method of Multipliers (ADMM) algorithm. The key novelty in the proposed algorithm is that it solves the problem in a decentralized topology where at most half of the workers are competing the limited communication resources at any given time. Moreover, each worker exchanges the locally trained model only with two neighboring workers, thereby training a global model with a lower amount of communication overhead in each exchange. We prove that GADMM converges faster than the centralized batch gradient descent for convex loss functions, and numerically show that it converges faster and more communication-efficient than the state-of-the-art communication-efficient algorithms such as the Lazily Aggregated Gradient (LAG) and dual averaging, in linear and logistic regression tasks on synthetic and real datasets.**

I. Introduction

Distributed optimization plays a pivotal role in distributed machine learning applications [1], [2], [3], [4] that commonly aims to minimize $\frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{\Theta})$ with N workers. As illustrated in Fig. 1-(a), this problem is often solved by locally minimizing $f_n(\boldsymbol{\theta}_n)$ at each worker and globally averaging their model parameters $\boldsymbol{\theta}_n$'s (and/or gradients) at a parameter server, thereby yielding the global model parameters $\boldsymbol{\Theta}$ [5]. Another way is to formulate the problem as an average consensus problem that minimizes $\frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{\theta}_n)$ under the constraint $\boldsymbol{\theta}_n = \boldsymbol{\Theta}, \forall n$ which can be solved using dual decomposition or Alternating Direction Method of Multipliers (ADMM). ADMM is preferable since standard dual decomposition may fail in updating the variables in some cases. For example, if the objective function $f_n(\boldsymbol{\theta}_n)$ is a nonzero affine function of any component in the input parameter $\boldsymbol{\theta}_n$, then the $\boldsymbol{\theta}_n$-update fails, since the Lagrangian is unbounded from below in $\boldsymbol{\theta}_n$ for most choices of the dual variables [6]. However, using ADMM or dual decomposition, an existence of a central entity is necessary.

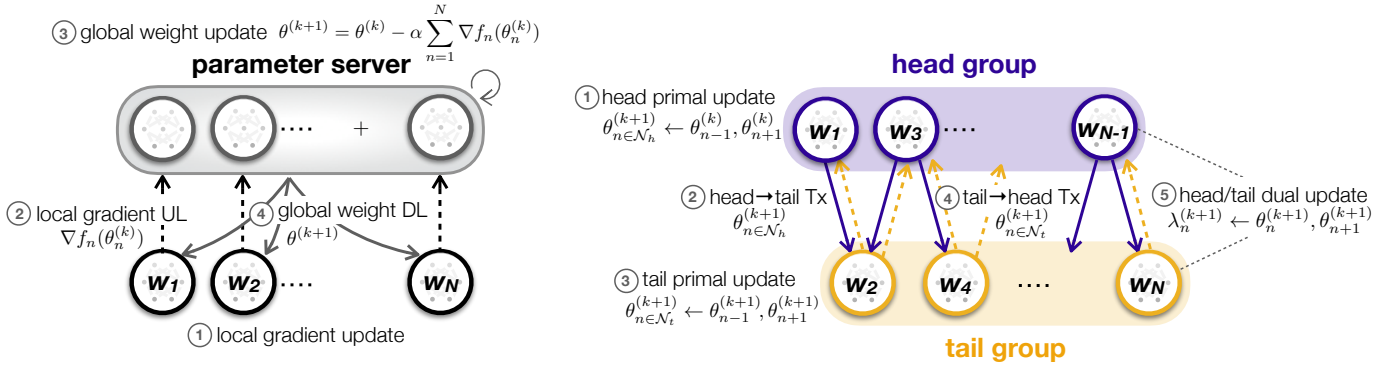
Such a centralized solution is, however, not capable of addressing a large network size exceeds the parameter server's coverage range. Even if the parameter server has a link to each worker, communication resources may become the bottleneck since all workers need to transmit their updates to the server before the server updates the global model and share it with workers. Hence, as the number of workers increases, the uplink communication resources become the bottleneck. Because of this, we aim to develop a fast and communication-efficient decentralized algorithm, and propose *Group Alternating Direction Method of Multipliers (GADMM)*. GADMM solves the problem $\frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{\theta}_n)$ subject to $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n+1}, \forall n \in \{1, \cdots, N-1\}$, in which the workers are divided into two groups (*head* and *Tail*), and each worker in the head (tail) group communicates only with its two neighboring workers from the tail (head) group as shown in Fig. 1-(b). Due to its communication with only two neighbors rather than all the neighbors or a central entity, the communication in each iteration is significantly reduced. Moreover, by dividing the workers into two equal groups, at most half of the workers are competing the communication resources at every communication round.

Despite this sparse communication, we prove that GADMM with a convex f_n enjoys $o(1/k)$ converges rate. We numerically show that its communication overhead is lower than that of state-of-the-art communication-efficient centralized and decentralized algorithms including Lazily Aggregated Gradient (LAG) [7], and dual averaging [8] for linear and logistic regression on synthetic and real datasets.

II. Related Works and Contributions

There are a variety of distributed optimization algorithms proposed in the literature, such as primal methods [9], [10], [11], [12] and primal-dual methods [13], [14], [15]. The performance of distributed optimization algorithms is commonly characterized by their computation time and communication cost. The computation time is decided by the per-iteration complexity of the algorithm. The communication cost is determined by: (i) the number of *communication rounds* until convergence, (ii) the number of *channel uses* per communication round, and (iii) the *bandwidth/power* per channel use. Note that the number of communication rounds is proportional to the number of iterations; *e.g.,* 2 rounds at every iteration k, for uplink and downlink transmissions in Fig. 1-(a) or for head-to-tail and tail-to-head transmissions in Fig. 1-(b). For a large scale network, the communication cost often becomes dominant compared to the computation time, calling for communication efficient distributed optimization [16], [17], [18], [19], [20], [21].

To reduce the bandwidth/power usage per channel use, decreasing communication payload sizes is one popular solution, which is enabled by gradient quantization [22], model parameter quantization [23], [21], and model output exchange for large-sized models via knowledge distillation [24]. To reduce



(a) **Distributed gradient descent** (centralized)

(b) **GADMM** (decentralized)

Fig. 1: An illustration of (a) distributed gradient descent with a parameter server and (b) GADMM without any central entity.

the number of channel uses per communication round, exchanging model updates can be restricted only to the workers whose computation delays are less than a target threshold [25], or to the workers whose updates are sufficiently changed from the preceding updates, with respect to gradients [7], or model parameters [20]. Albeit their improvement in communication efficiency for every iteration k, most of the algorithms in this literature are based on distributed gradient descent, and this limits their required communication rounds to the convergence rate of distributed gradient descent, which is $\mathcal{O}(1/k)$ for differentiable and smooth objective functions.

On the other hand, primal-dual decomposition methods are shown to be effective in enabling distributed optimization [26], [6], [27], [28], [29], [30], among which ADMM is a compelling solution that often provides a fast convergence rate with low complexity [28], [29], [30]. However, all aforementioned algorithms including standard ADMM require a parameter server being connected to every worker, which may induce a costly communication link to some workers.

For decentralized topology, decentralized gradient descent (DGD) has been investigated in [31]. Beyond the GD based approach, several communication-efficient decentralized algorithms were proposed [8], [32]. However, compared to GADMM, they either achieve a slower convergence rate or transmit more parameters per iteration.

To summarize the contribution, we propose a novel algorithm to solve the DML problem optimally for convex functions. The proposed algorithm is shown to be fast and communication-efficient. It enjoys the same convergence rate of standard ADMM, but with significantly less communication overhead. The proposed GADMM algorithm allows (i) only half of the workers to transmit their updated parameters at each communication round, (ii) the workers update their model parameters in parallel, while each worker communicates only with two neighbors which makes it communication-efficient.

The rest of the paper is organized as follows. In section III, we describe the problem formulation. We describe our proposed variant of ADMM (GADMM) and analyze its convergence guarantees in sections IV. In section V, we discuss our simulation results comparing GADMM to the considered baselines. Finally, in section VI, we conclude the paper and briefly discuss future directions.

III. Problem Formulation

We consider a network of N workers where each worker is equipped with the task to learn a global parameter $\boldsymbol{\Theta}$. The aim is to minimize the global convex loss function which is sum of the local convex, proper, and closed functions $f_n(\boldsymbol{\theta}_n)$ for all n. Since the goal is to solve the problem in a distributed manner, we consider the following optimization problem.

$$\boldsymbol{\theta}^\star := \arg \min_{\{\boldsymbol{\theta}_n\}_{n=1}^N} \sum_{n=1}^{N} f_n(\boldsymbol{\theta}_n) \tag{1}$$

$$\text{s.t. } \boldsymbol{\theta}_n = \boldsymbol{\theta}_{n+1}, \quad n = 1, \cdots, N-1. \tag{2}$$

where $\boldsymbol{\theta}_n \in \mathbb{R}^d$ is the model parameter of the n-th worker. $\boldsymbol{\lambda} := [\boldsymbol{\lambda}_1^T, \cdots, \boldsymbol{\lambda}_N^T]^T$ is the collection of dual variables, and ρ is a constant adjusting the penalty for the disagreement between $\boldsymbol{\theta}_n$ and $\boldsymbol{\theta}_{n+1}$.

Here $\boldsymbol{\theta}^\star$ is the optimal and note that $\boldsymbol{\theta}_{n-1}^\star = \boldsymbol{\theta}_n^\star$ and $\boldsymbol{\theta}_n^\star = \boldsymbol{\theta}_{n+1}^\star$ for all n. This implies that each worker n has joint constraints with only two neighbors (except for the two end workers which have only one). Nonetheless, ensuring $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n+1}$ for all $n \in \{1, \cdots, N-1\}$ at the convergence point yields convergence to a global model parameter that is shared across all workers.

IV. Proposed Algorithm: GADMM

We now describe our proposed algorithm. The main idea of the proposed algorithm is presented in Fig. 1-(b). The proposed GADMM algorithm splits the network nodes (workers) connected with a chain into two groups *head* and *tail* such that each worker in the head's group is connected to other workers through two tail workers. It allows updating the parameters in parallel for the workers in the same group. In one algorithm iterate, the workers in the head group update their model parameters, and each head worker transmits its updated model to its directly connected tail neighbors. Then,

tail workers update their model parameters to complete one iteration. In doing so, each worker (except the edge workers) communicates with only two neighbors to update its parameter, as depicted in Fig. 1-(b). Moreover, at any communication round, only half of the workers transmit their parameters, and these parameters are transmitted to only two neighbors.

We now describe the steps of GADMM (Algorithm 1) in more details. Without loss of generality, we consider an even N number of workers under their linear connectivity graph shown in Fig. 1-(b). With that in mind, we start by writing the augmented Lagrangian of the optimization problem in (1)-(2) as.

$$\mathcal{L}_\rho(\{\boldsymbol{\theta}_n\}_{n=1}^N, \boldsymbol{\lambda}) = \sum_{n=1}^{N} f_n(\boldsymbol{\theta}_n) + \sum_{n=1}^{N-1} \langle \boldsymbol{\lambda}_n, \boldsymbol{\theta}_n - \boldsymbol{\theta}_{n+1} \rangle + \frac{\rho}{2} \sum_{n=1}^{N-1} \|\boldsymbol{\theta}_n - \boldsymbol{\theta}_{n+1}\|^2, \tag{3}$$

Let's divide N workers into two groups, head $\mathcal{N}_h = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_3, \boldsymbol{\theta}_5, \cdots, \boldsymbol{\theta}_{N-1}\}$, and tail $\mathcal{N}_t = \{\boldsymbol{\theta}_2, \boldsymbol{\theta}_4, \boldsymbol{\theta}_6, \cdots, \boldsymbol{\theta}_N\}$, respectively. The primal and dual variables under GADMM are updated in the following three steps.

1) At iteration $k+1$, the *primal variables of head workers* are updated as:

$$\boldsymbol{\theta}_n^{k+1} = \arg\min_{\boldsymbol{\theta}_n} \big[f_n(\boldsymbol{\theta}_n) + \langle \boldsymbol{\lambda}_{n-1}^k, \boldsymbol{\theta}_{n-1}^k - \boldsymbol{\theta}_n \rangle + \langle \boldsymbol{\lambda}_n^k, \boldsymbol{\theta}_n - \boldsymbol{\theta}_{n+1}^k \rangle + \frac{\rho}{2} \|\boldsymbol{\theta}_{n-1}^k - \boldsymbol{\theta}_n\|^2 + \frac{\rho}{2} \|\boldsymbol{\theta}_n - \boldsymbol{\theta}_{n+1}^k\|^2 \big], n \in \mathcal{N}_h \setminus \{1\} \tag{4}$$

Since the first head worker ($n = 1$) does not have a left neighbor ($\boldsymbol{\theta}_{n-1}$ is not defined), its model is updated as follows.

$$\boldsymbol{\theta}_n^{k+1} = \arg\min_{\boldsymbol{\theta}_n} \big[f_n(\boldsymbol{\theta}_n) + \langle \boldsymbol{\lambda}_n^k, \boldsymbol{\theta}_n - \boldsymbol{\theta}_{n+1}^k \rangle + \frac{\rho}{2} \|\boldsymbol{\theta}_n - \boldsymbol{\theta}_{n+1}^k\|^2 \big], n = 1 \tag{5}$$

2) After the updates in (4) and (5), head workers send their updates to their two tail neighbors. The *primal variables of tail workers* are then updated as:

$$\boldsymbol{\theta}_n^{k+1} = \arg\min_{\boldsymbol{\theta}_n} \big[f_n(\boldsymbol{\theta}_n) + \langle \boldsymbol{\lambda}_{n-1}^k, \boldsymbol{\theta}_{n-1}^{k+1} - \boldsymbol{\theta}_n \rangle + \langle \boldsymbol{\lambda}_n^k, \boldsymbol{\theta}_n - \boldsymbol{\theta}_{n+1}^{k+1} \rangle + \frac{\rho}{2} \|\boldsymbol{\theta}_{n-1}^{k+1} - \boldsymbol{\theta}_n\|^2 + \frac{\rho}{2} \|\boldsymbol{\theta}_n - \boldsymbol{\theta}_{n+1}^{k+1}\|^2 \big], n \in \mathcal{N}_t \setminus \{N\}. \tag{6}$$

Since the last tail worker ($n = N$) does not have a right neighbor ($\boldsymbol{\theta}_{n+1}$ is not defined), its model is updated as follows.

$$\boldsymbol{\theta}_n^{k+1} = \arg\min_{\boldsymbol{\theta}_n} \big[f_n(\boldsymbol{\theta}_n) + \langle \boldsymbol{\lambda}_{n-1}^k, \boldsymbol{\theta}_{n-1}^{k+1} - \boldsymbol{\theta}_n \rangle + \frac{\rho}{2} \|\boldsymbol{\theta}_{n-1}^{k+1} - \boldsymbol{\theta}_n\|^2 \big], n = N. \tag{7}$$

3) After receiving the updates from neighbors, *every worker locally updates its dual variables* $\boldsymbol{\lambda}_{n-1}$ and $\boldsymbol{\lambda}_n$ as follows

$$\boldsymbol{\lambda}_n^{k+1} = \boldsymbol{\lambda}_n^k + \rho(\boldsymbol{\theta}_n^{k+1} - \boldsymbol{\theta}_{n+1}^{k+1}), n = \{1, \cdots, N-1\}. \tag{8}$$

Algorithm 1 Group ADMM (GADMM)

1: **Input**: $N, f_n(\boldsymbol{\theta}_n)$ for all n, ρ
2: **Initialization**:
3: $\mathcal{N}_h = \{\boldsymbol{\theta}_n \mid n$: odd$\}$, $\mathcal{N}_t = \{\boldsymbol{\theta}_n \mid n$: even$\}$
4: $\boldsymbol{\theta}_n^{(0)} = 0, \boldsymbol{\lambda}_n^{(0)} = 0$ for all n
5: **for** $k = 0, 1, 2, \cdots, K$ **do**
6: **Head worker** $n \in \mathcal{N}_h$**:**
7: **computes** its primal variable $\boldsymbol{\theta}_n^{k+1}$ via (4) in parallel; and
8: **sends** $\boldsymbol{\theta}_n^{k+1}$ to its neighboring workers $n-1$ and $n+1$.
9: **Tail worker** $n \in \mathcal{N}_t$**:**
10: **computes** its primal variable $\boldsymbol{\theta}_n^{k+1}$ via (6) in parallel; and
11: **sends** $\boldsymbol{\theta}_n^{k+1}$ to its neighbor workers $n-1$ and $n+1$.
12: **Every worker updates** the dual variables $\boldsymbol{\lambda}_{n-1}^k$ and $\boldsymbol{\lambda}_n^k$ via (8) locally.
13: **end for**

The convergence analysis of the proposed algorithm and the detailed steps of the optimality proof for convex functions is described in details in [33]. The idea to prove the convergence is related to the proof of Gauss-Seidel ADMM in [6], while additionally accounting for the following three challenges: (i) the additional terms that appear when the problem is a sum of more than two separable functions, (ii) the fact that each worker can communicate with two neighbors only, and (iii) the parallel model parameter updates of the head (tail) workers. We show that the GADMM iterates converge to the optimal solution after addressing all the above-mentioned challenges in the proof.

V. Numerical Results

To validate our theoretical foundations, we numerically evaluate the performance of GADMM in linear regression tasks, compared with the following benchmark algorithms.

- **LAG-PS** [7]: A version of LAG where parameter server selects communicating workers.
- **LAG-WK** [7]: A version of LAG where workers determine when to communicate with the server.
- **Cycle-IAG** [34], [35]: A cyclic modified version of the incremental aggregated gradient (IAG).
- **R-IAG** [7], [36]: A non-uniform sampling version of stochastic average gradient (SAG).
- **GD**: Batch gradient descent.
- **DGD** [31]: Decentralized gradient descent.
- **DualAvg** [8]: Dual averaging.

For the tuning parameters, we use the setup in [7]. For our decentralized algorithm, we consider N workers without any central entity, whereas for centralized algorithms, a uniformly randomly selected worker is considered as a central controller

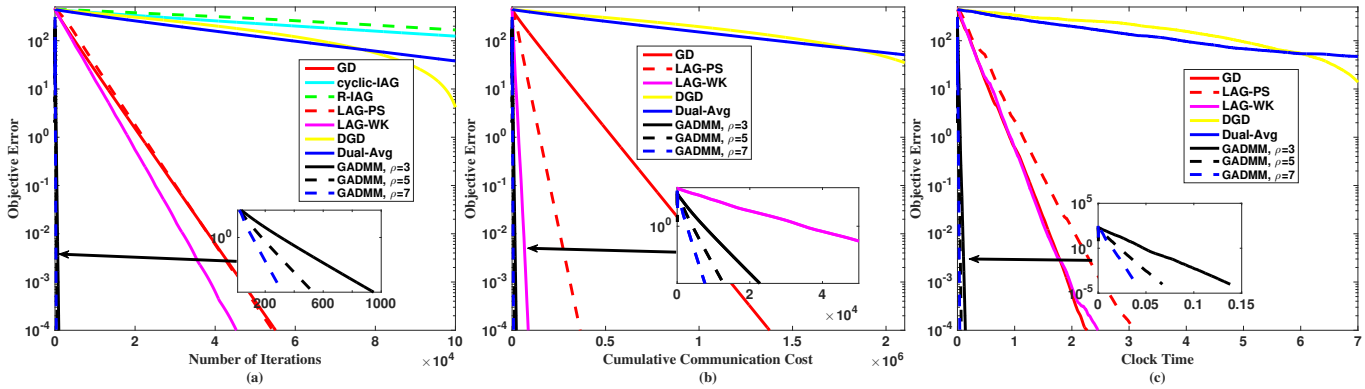


Fig. 2: Objective error, total communication cost, and total running time comparison between GADMM and five benchmark algorithms, in *linear* regression with synthetic ($N = 24$) datasets.

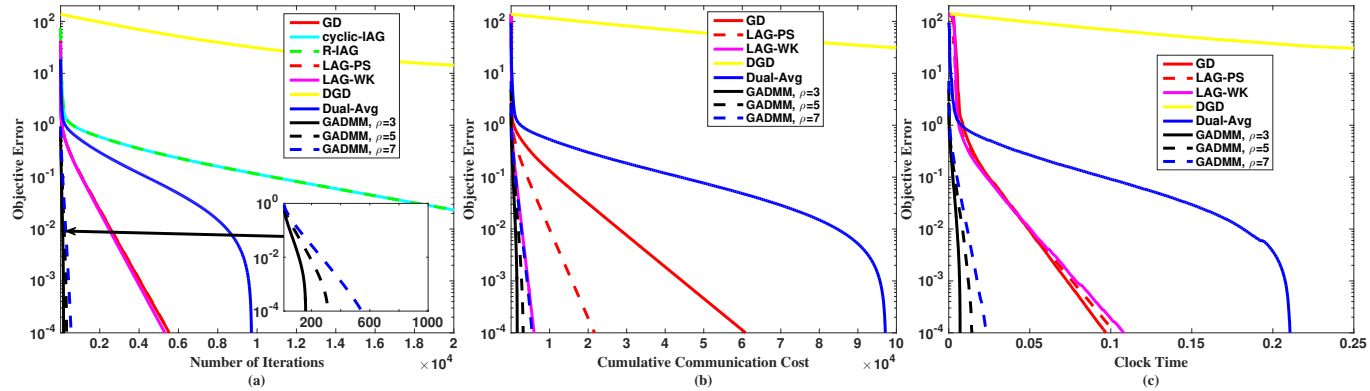


Fig. 3: Objective error, total communication cost, and total running time comparison between GADMM and five benchmark algorithms, in *linear* regression with real ($N = 10$) datasets.

having a direct link to each worker. The performance of each algorithm is measured using:

- the **objective error** $|\sum_{n=1}^{N}\left[f_n(\boldsymbol{\theta}_n^{(k)}) - f_n(\boldsymbol{\theta}^*)\right]|$ at iteration k.
- (ii) The **total communication cost (TC)**. The TC of a decentralized algorithm is $\sum_{t=1}^{T_a}\sum_{n=1}^{N}\mathbf{1}_{n,t}\cdot L_{n,t}^m$, where T_a is the number of iterations to achieve a target accuracy a, and $\mathbf{1}_{n,t}$ denotes an indicator function that equals 1 if worker n is sending an update at t, and 0 otherwise. The term $L_{n,t}^m$ is the cost of the communication link between workers n and m at communication round t. Next, let $L_{n,t}^c$ denote the cost of the communication between worker n and the central controller at t. Then, the TC of a centralized algorithm is $\sum_{t=1}^{T_a}(L_{\text{BC},t}^c + \sum_{n=1}^{N}\mathbf{1}_{n,t}\cdot L_{n,t}^c)$, where $L_{\text{BC},t}^c$ and $L_{n,t}^c$'s correspond to downlink broadcast and uplink unicast costs, respectively. It is noted that the communication overhead in [7] only takes into account uplink costs.
- The total running time (clock time) to achieve objective error a. This metric considers both the communication and the local computation time. We consider $L_{n,t}^m = L_{n,t}^c = L_{\text{BC},t}^c = 1$ unless otherwise specified.

All simulations are conducted using the synthetic and real datasets described in [37], [7]. The synthetic data is generated as described in [7]. We consider 1,200 samples with 50 features, which are evenly split into workers. Next, the real data tests linear regression task with **Body Fat** (252 samples, 14 features). As the real dataset sizes is smaller than the synthetic dataset, we by default consider 10 and 24 workers for the real and synthetic datasets, respectively.

Figs. 2, 3, corroborate that GADMM outperforms the benchmark algorithms by several orders of magnitudes, thanks to the idea of two groups with alternating updates where each worker communicates with only two neighbors. For linear regression with the synthetic dataset, Fig. 2 shows that all variants of GADMM with $\rho = 3, 5$, and 7 achieve the target objective error of 10^{-4} in less than 1,000 iterations, whereas GD, LAG-PS, and LAG-WK (the closest among baselines) require more than 40,000 iterations to achieve the same target error. Furthermore, the TC of GADMM with $\rho = 3$ and $\rho = 5$ are 6 and 9 times lower than that of LAG-WK respectively. We also observe from Figs. 2 and 3 that GADMM outperforms all baselines in terms of the total running time, thanks to the fast convergence. GADMM performs matrix inversion which

is computationally complex compared to calculating gradient. However, the computation cost per iteration is compensated by fast convergence.

VI. Conclusions and Future work

In this paper, we formulate a constrained optimization problem for distributed machine learning applications, and propose a novel decentralized algorithm based on ADMM, termed GADMM to solve this problem optimally for convex functions. GADMM is shown to improve the communication efficiency of each worker. Extensive simulations in linear regression with synthetic and real datasets show significant improvements in communication overhead as compared to the state-of-the-art algorithms.

References

[1] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," *In Proceedings of World Wide Web, Rio de Janeiro, Brazil*, May 2013.

[2] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1223–1231, 2012.

[3] M. Li, D. G. Andersen, and A. Smola, "Distributed delayed proximal gradient methods," *presented at Neural Information Processing Systems Workshop on Optimization for Machine Learning, Lake Tahoe, NV, USA*, December 2013.

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

[5] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," *In Proceedings of Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA*, October 2012.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "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.

[7] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," *Advances in Neural Information Processing Systems*, vol. 31, pp. 5055–5065, 2018.

[8] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic control*, vol. 57, no. 3, pp. 592–606, 2011.

[9] D. Jakovetić, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automation and Control Automa. Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[10] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Automa. Control*, vol. 60, no. 3, pp. 601–615, 2014.

[11] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automation and Control*, vol. 54, no. 1, pp. 48–61, 2009.

[12] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6013–6023, 2015.

[13] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus admm," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2014.

[14] A. Koppel, B. M. Sadler, and A. Ribeiro, "Proximity without consensus in online multiagent optimization," *IEEE Transactions on Signal Processing*, vol. 65, no. 12, pp. 3062–3077, 2017.

[15] A. S. Bedi, A. Koppel, and R. Ketan, "Asynchronous saddle point algorithm for stochastic optimization in heterogeneous networks," *IEEE Transactions on Signal Processing*, vol. 67, no. 7, pp. 1742–1757, 2019.

[16] Y. Zhang, M. J. Wainwright, and J. C. Duchi, "Communication-efficient algorithms for statistical optimization," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1502–1510, 2012.

[17] H. B. McMahan, R. D. Moore, Eider, S. Hampson, and B. A. yArcas, "Communication-efficient learning of deep networks from decentralized data," *In Proceedings of Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA*, April 2017.

[18] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *to appear in Proceedings of the IEEE [Online]. Early access is available at: https://ieeexplore.ieee.org/document/8865093*, November 2019.

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

[20] Y. Liu, W. Xu, G. Wu, Z. Tian, and Q. Ling, "Communication-censored ADMM for decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2565–2579, 2019.

[21] N. Sriranga, C. R. Murthy, and V. Aggarwal, "A method to improve consensus averaging using quantized admm," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019.

[22] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," *Proceedings of Machine Learning Research*, vol. 70, pp. 3329–3337, 2017.

[23] S. Zhu, M. Hong, and B. Chen, "Quantized consensus ADMM for multi-agent distributed optimization," *In Proceedings of International Conference on Acoustics, Speech, and Signal Processing, Shanghai, China*, March 2016.

[24] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *presented at Neural Information Processing Systems Workshop on Machine Learning on the Phone and other Consumer Devices (MLPCD), Montréal, Canada*, 2018. [Online]. Available: http://arxiv.org/abs/1811.11479

[25] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *ArXiv preprint*, vol. abs/1804.05271, 2018.

[26] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," *Advances in Neural Information Processing Systems*, vol. 27, pp. 3068–3076, 2014.

[27] C. Ma, J. Konečnỳ, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takáč, "Distributed optimization with arbitrary local solvers," *Optimization Methods and Software*, vol. 32, no. 4, pp. 813–848, 2017.

[28] R. Glowinski and A. Marroco, "Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires," *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, vol. 9, no. R2, pp. 41–76, 1975.

[29] D. Gabay and B. Mercier, *A dual algorithm for the solution of non linear variational problems via finite element approximation*. Institut de recherche d'informatique et d'automatique, 1975.

[30] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, "Parallel multi-block admm with $o(1/k)$ convergence," *Journal of Scientific Computing*, vol. 71, no. 2, pp. 712–736, 2017.

[31] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[32] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee, "Optimal algorithms for non-smooth distributed optimization in networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 2740–2749.

[33] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "Gadmm: Fast and communication efficient framework for distributed machine learning," 2019.

[34] D. Blatt, A. O. Hero, and H. Gauchman, "A convergent incremental gradient method with a constant step size," *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 29–51, 2007.

[35] M. Gurbuzbalaban, A. Ozdaglar, and P. A. Parrilo, "On the convergence rate of incremental aggregated gradient algorithms," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 1035–1048, 2017.

[36] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.

[37] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml