COMMUNICATION-EFFICIENT DECENTRALIZED FEDERATED LEARNING USING LOCALLY AGGREGATED ADMM

Meng Ma[*] *Yunlong Wang*[†] *Georgios B. Giannakis*[*]

[*]Digital Technology Center and Dept. of ECE, University of Minnesota
[†]Department of Advanced Analytic, IQIVIA

ABSTRACT

Federated learning enables training a shared machine learning model while keeping data distributed in several locations, typically with a central server coordinating all agents. In this paper, we focus on improving the communication efficiency of decentralized federated learning based on alternating direction method of multipliers (ADMM). We propose a local aggregation based ADMM that enjoys faster convergence thus less communication rounds. Moreover, with quantization we can further reduce the communication complexity without comprising the accuracy too much. We test our algorithm on real world applications to showcase the merits.

Index Terms— decentralized federated learning, ADMM, quantization, communication efficient

1. INTRODUCTION

With the recent explosion of deep learning techniques for unprecedented pattern recognition and predictive modeling, researchers are continually looking for ways to gather together datasets from disparate sources to provide sufficient training data for these complex models. In health care industry, for example, due to the fundamental privacy concerns, there is a great risk to store health care records in a centralized location. Typically, institutes only have access to the records of their own patients, making learning a challenging task on this small dataset. Moreover, the data could be unevenly distributed, posing further difficulties for building a proper model.

Federated learning (FL) was proposed to deal with such scenarios. In federated settings, multiple agents collaboratively learn a shared model while keeping all training data private. Existing federated learning strategies typically require the presence of central server that is connected to every agents. The locally trained model parameters are shared with the central server, which aggregates to produce a global model that is sent back to agents. By repeating this process, a global model is effectively trained on all agents' private data. The communication overhead per round scales linearly with the number of agents and size of model parameters. This paradigm is challenged by the increasing number of agents and growing complexity of machine learning models.

Another potential difficulty facing federated learning is privacy. The fact that the central server has access to information from all agents might raise severe concerns of violating regulations. For instance, patient-level data exchange among stakeholder such as insurance companies and treating facilities is prohibited by law [1].

1.1. Problem formulation

The training of a machine learning model is often formulated in finite sum form [2]. Suppose that we have a dataset consisting of N data samples, $\{\mathbf{x}_i, y_i\}_{i=1}^N$, and a proper loss function $l_i : \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}$. Then the training process boils down to minimizing the overall objective

$$\min_{\mathbf{w}\in\mathbb{R}^p} \; f(\mathbf{w}) \quad \text{with} \quad f(\mathbf{w}) := \sum_{i=1}^{N} f_i(\mathbf{w}), \tag{1}$$

where f_i is the local objective of agent i, which is the sum of loss on all samples, i.e., $f_i(\mathbf{w}) = 1/|\mathcal{S}_i| \sum_{i\in\mathcal{S}_i} l(\mathbf{x}_i, y_i; \mathbf{w})$. We use $\mathcal{S}_i \subset \{1, \ldots, N\}$ to denote the set of data sample index of agent i, and $n_i = |\mathcal{S}_i|$ the size of dataset. Notice that the sizes different agent are not necessarily equal.

1.2. Related work

The basic algorithm for solving finite sum problem is *gradient descent* (GD) for smooth f and *subgradient* for nonsmooth functions, which at iteration t performs

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \nabla f(\mathbf{w}^{(t)}), \tag{2}$$

where η_t is the step size or *learning rate*. The computation of the gradient $\nabla f(\mathbf{x}^{(t)})$ involves iterating through every sample, so the complexity scales linearly with N. When the number of samples is too large, this quickly becomes too expensive. A common approach is the *stochastic gradient descent* (SGD), which per iteration performs update $\mathbf{w}$ using partial gradient. In federated setting, this could be realized by randomly selecting a few agents who compute local *stochastic*

This work is supported in part by NSF grant 1901134.

gradient and send to central server to obtain the aggregated global model.

SGD is a simple yet effective method for real world applications [2,3]. The distributed variant,DSGD, is also a popular choice for federated learning [4]. Variance reduce methods, such as SVRG, SARAH, etc., perform better than SGD with some nice properties. As a results, variance reduction methods adaptive to federated setting are also proposed, see [2] and reference therein. To enhance the privacy of federated learning, various ways has been employed to make it difficult to infer identify information, see e.g., [5, 6]. Conventional federated learning may experience severe performance degradation when data are distributed evenly [7].

1.3. Our contribution

In order to comply restrictive privacy regulations, we propose to a decentralized federated learning using ADMM, which forgoes the central server and only allow agent to agent communication. The objective function f can be non-differentiable. We propose to use a locally aggregated ADMM, which converges faster than vanilla ADMM, thus requires less round of communications. Moreover, we can further reduce the communication overhead by transfer quantized values. Finally, we test our algorithm on real datasets to demonstrate its merits.

2. DECENTRALIZED FEDERATED LEARNING USING ADMM

We model communication constraints in the decentralized setting as an undirected graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, with each vertex $i \in \mathcal{V}$ corresponding to one agent, and edge $(i, j) \in \mathcal{E}$ denoting that agents i and j can share information.

The finite sum form (1) can be formulated into consensus form and solved by ADMM. Each agent maintains a local model while a central server updates the global model [8]. Let $\mathbf{w}_i$ be the local model parameter of agent i, and $\mathbf{w}$ the global model. Then finite sum problem can be formulated as

$$\min_{\{\mathbf{w}_i\},\mathbf{w}} \sum_{i=1}^{N} f_i(\mathbf{w}_i) \quad \text{s. to} \quad \mathbf{w}_i = \mathbf{w}, \forall i. \tag{3}$$

The constrained minimization problem turns out to be separable while solved using ADMM, i.e., each agent can only train its local model and the central server is responsible for aggregating all local models and then broadcasting back global model to each agent. This paradigm, however, shares the same issues with GD-based federated learning due to the presence of a central server.

Vanilla ADMM for decentralized FL. To accommodate the privacy regulations, a decentralized ADMM algorithm has been proposed to solve (1), which does not need the central server and only allows information sharing between two trusted neighbors [9]. Mathematically, this translates to

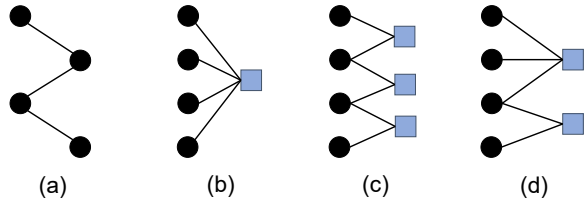

Fig. 1. An example network and its communication graphs. Each circle denotes a agent and each square denotes a server where aggregation of local models is performed.

$$\min_{\mathbf{w}_i,\mathbf{z}_{ij}} \sum_{i=1}^{N} f_i(\mathbf{w}_i) \quad \text{s. to } \mathbf{w}_i = \mathbf{z}_{ij},\ \mathbf{w}_j = \mathbf{z}_{ji},\ \forall (i,j) \in \mathcal{E} \tag{4}$$

where $\mathbf{z}_{ij}$ is an auxiliary variable created for edge (i, j). With some algebra, one can show that the ADMM steps can be carried out without $\mathbf{z}_{ij}$ [9, 10].

Locally aggregated ADMM for decentralized FL. It has been observed that decentralized ADMM performs much worse than its centralized counterpart, because the aggregation of local models are performed locally between neighbors instead of globally [11]. This pairwise aggregation could yield a much slower convergence speed once the network becomes so large that it takes too many rounds of communication for information to propagate between remote agents. Motivated by this, we propose to do local aggregations to accelerate the convergence speed of ADMM.

The idea is to deploy multiple local servers, each of which is connected to a subset of agents and responsible for aggregating their local models. As long as the network is connected, the information of each agent can eventually propagate to the whole network and consensus could be achieved. Figure 1 illustrates this idea, where (a) is an example network of four agents, (b)(c)(d) are the communication graphs of ADMM corresponding to the cases with a central server, without a central server, and with local servers, respectively.

Suppose there are M local servers. Let $\mathbf{z}_j$ be the local model at server j, then we need to ensure $\mathbf{w}_i = \mathbf{z}_j$ if agent i is connected to server j. Denote T the number of agent-server connections. Take Fig. 1(d) for example, $N = 4, M = 3, T = 5$. If we stack all $\mathbf{w}_i$ and $\mathbf{z}_j$ as rows of matrices $\mathbf{W}$ and $\mathbf{Z}$, then (1) is equivalent to

$$\begin{aligned} \min_{\mathbf{W},\mathbf{Z}} \quad & f(\mathbf{W}) = \sum_i f_i(\mathbf{w}_i) \\ \text{s. to} \quad & \mathbf{AW} = \mathbf{BZ} \end{aligned} \tag{5}$$

where $\mathbf{A} \in \{0,1\}^{T\times N}$ and $\mathbf{B} \in \{0,1\}^{T\times M}$ are matrices that select the proper agents and servers so that each row of $\mathbf{AW} = \mathbf{BZ}$ reduces to $\mathbf{w}_i = \mathbf{z}_j$ for some i, j. Given a communication graph like Fig. 1(d), one can build these matrices

row by row: each row of $\mathbf{A}$ ($\mathbf{B}$) is a canonical vector that has one at its i-th (j-th) entry for the connection between agent i and server j. The labels of agents and servers can be arbitrary.

Let d_i (e_j) be the degree of agent i (server j), i.e., the number of connections attached to it. One can prove that by constructing $\mathbf{A}$ and $\mathbf{B}$ this way, it holds that $\mathbf{D} = \mathbf{A}^\top\mathbf{A}$, $\mathbf{E} = \mathbf{B}^\top\mathbf{B}$, and $\mathbf{C} = \mathbf{A}^\top\mathbf{B}$ [12, Lemma 1], where $\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_N)$, $\mathbf{E} = \operatorname{diag}(e_1, \ldots, e_M)$, and $\mathbf{C} \in \{0,1\}^{N\times M}$ is the incidence matrix that has one at its (i,j)-th entry if agent i is connected to server j.

Let $\{\boldsymbol{\lambda}_t\}_{t=1}^T$ be the Lagrangian multipliers, $\boldsymbol{\Lambda} \in \mathbb{R}^{T\times p}$ the matrix by stacking all $\boldsymbol{\lambda}_t$ as rows, and $\mathbf{Y} = \mathbf{A}^\top\boldsymbol{\Lambda}$ whose i-th row is $\mathbf{y}_i \in \mathbb{R}^p$. With some simple manipulations, we can show that the iterations of ADMM with local aggregation reduces to

$$\begin{aligned}\mathbf{w}_i^{(t+1)} &= \operatorname*{argmin}_{\mathbf{w}_i} f_i(\mathbf{w}_i) + \frac{\rho d_i}{2}\|\mathbf{w}_i\|_2^2 + \langle \mathbf{w}_i, \mathbf{y}_i^{(t)} - \rho\mathbf{v}_i^{(t)}\rangle \\ \mathbf{y}_i^{(t+1)} &= \mathbf{y}_i^{(t)} + \rho(d_i\mathbf{w}_i^{(t+1)} - \mathbf{v}_i^{(t+1)})\end{aligned} \tag{6}$$

where $\mathbf{v}_i^{(t)} = \sum_{j=1}^M c_{ij}\mathbf{z}_j^{(t)}$ and $\mathbf{z}_j^{(t+1)} = \frac{1}{e_j}\sum_{i=1}^N c_{ij}\mathbf{w}_i^{(t+1)}$. See [12] for detailed derivation. Notice that agent i can compute $\mathbf{v}_i$ by receiving all local models from its connected servers, which is basically what the summation says. Similarly, server j can update local $\mathbf{z}_j$ by aggregate all local models from connected agents. Thus every step can be implemented using neighbor-wise communication only.

Various experiments have shown that local aggregation can help speed up the convergence of ADMM for solving finite sum problems in a decentralized manner [12], thus reducing the required number of communication rounds and consequently less communication cost.

3. QUANTIZED LOCALLY AGGREGATED ADMM

So far, we have assumed infinite accuracy for communication and computation step in all the discussions. In reality, however, it is rarely this case. Computers can only afford certain bits for each number, and communication links are often subject to noise. To further reduce communication cost, we consider quantized communication in federated setting.

Quantization is the process of mapping an arbitrary real number to some discrete values that can be represented by a finite number of bits. These discrete values are often referred as quantization lattice. In fact, this is how modern computers store numbers internally, typically using 32 or 64 bits to represent an integer. These discrete values do not need to be equally spaced, but we only consider the equal case for simplicity in the sequel. The space between two consecutive values is called *quantization resolution*.

Let $\Delta > 0$ be some quantization resolution. A quantizer is a function Q that maps a real value to some quantized lattice. This mapping can be deterministic or probabilistic. We consider only probabilistic quantization in this paper. In particular, for a real number $x \in [k\Delta, (k+1)\Delta)$, a probabilistic quantized is the function

$$Q(x) = \begin{cases} k\Delta & \text{with probability } k+1-x/\Delta, \\ (k+1)\Delta & \text{with probability } x/\Delta - k. \end{cases} \tag{7}$$

We adopt the convention that quantizing a vector or matrix means quantizing each elements independently. It immediately follows that the quantization error $|Q(x) - x| < \Delta$. For a given resolution Δ and a real value x, it's easy to prove that the probabilistic quantization satisfies

$$\mathbb{E}[Q(x)] = x, \quad \operatorname{var}(Q(x)) < \Delta^2/4.$$

Intuitively, the smaller the resolution is, the less communication overhead is needed, and the less accuracy is achieved. One extreme case would be using only 1 bit information [13]. In such cases, a real value x is quantized to

$$Q(x) = \begin{cases} x_{min} & \text{with probability } \frac{x-x_{min}}{x_{max}-x_{min}}, \\ x_{max} & \text{with probability } \frac{x_{max}-x}{x_{max}-x_{min}}. \end{cases} \tag{8}$$

where $\Delta = x_{max} - x_{min}$. This quantization can immediately achieve 32x or 64x communication efficiency on a typical computer. We gained this efficiency by sacrificing the accuracy of solution, as we show in numerical experiments. So there is a fundamental trade-off between communication cost and solution accuracy.

Quantization has been employed to reduce communication cost of ADMM, see [14] and reference therein. As discussed before, the iterations (6) can be implemented based on neighbor-wise communication only. Values sent and received are subject to quantization before transferred.

$$\begin{aligned}\mathbf{w}_i^{(t+1)} &= \operatorname*{argmin}_{\mathbf{w}_i} f_i(\mathbf{w}_i) + \frac{\rho d_i}{2}\|\mathbf{w}_i\|_2^2 + \langle \mathbf{w}_i, \mathbf{y}_i^{(t)} - \rho\mathbf{v}_i^{(t)}\rangle \\ \mathbf{y}_i^{(t+1)} &= \mathbf{y}_i^{(t)} + \rho(d_i\mathbf{w}_i^{(t+1)} - \mathbf{v}_i^{(t+1)})\end{aligned} \tag{9}$$

where $\mathbf{v}_i^{(t)} = \sum_{j=1}^M c_{ij}Q(\mathbf{z}_j^{(t)})$ and $\mathbf{z}_j^{(t)} = \frac{1}{e_j}\sum_{i=1}^N c_{ij}Q(\mathbf{w}_i^{(t)})$. Notice the update of $\mathbf{w}_i$ does not rely on quantized $\mathbf{y}_i$ because it is also maintained by agent i. The same reason for not quantizing $\mathbf{w}^{(t+1)}$ in the update of $\mathbf{y}^{(t+1)}$.

Theorem 1. *If ∇f_i is linear in $\mathbf{w}_i$, f is strongly convex and has Lipschitz continuous gradient, and the minimization* (5) *admits at least one solution, then iterations* (6) *converge linearly in the mean sense, i.e., $\mathbb{E}[\mathbf{w}_i^{(t)}], \mathbb{E}[\mathbf{y}_i^{(t)}]$ converge to their optimal solutions with linear rate.*

The linear convergence of ADMM with local aggregation has been established [12]. If the assumption of Theorem 1 holds, then taking expectation of [12, Algorithm 1], then the proof carries over to $\mathbb{E}[\mathbf{w}_i^{(t)}], \mathbb{E}[\mathbf{y}_i^{(t)}]$.

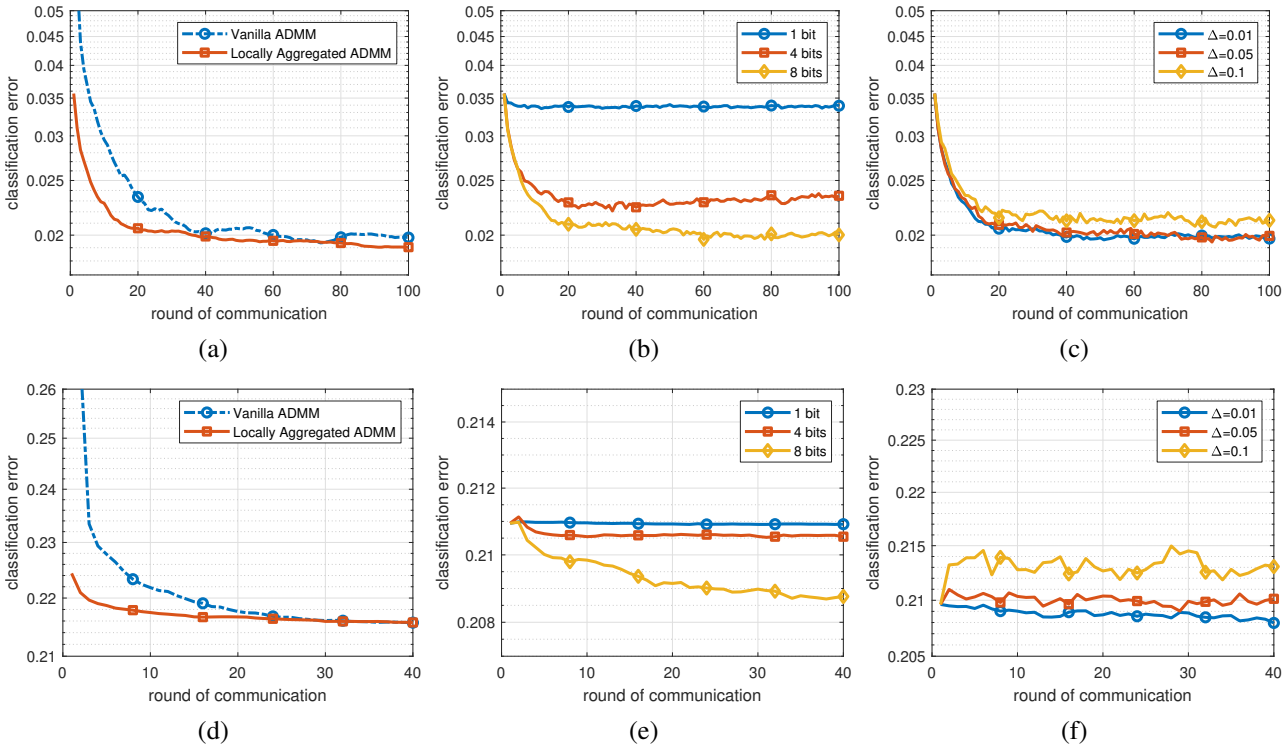

Fig. 2. Test results of ADMM for decentralized federated learning on two datasets: MNIST and Alzheimer Disease.

4. NUMERICAL EXPERIMENTS

In this section, we test our algorithm on some datasets. To demonstrate its merits, we also test vanilla ADMM for decentralized federated learning, as discussed in Section 2. We use the standard linear SVM classifier. As a result, the local objective function is defined as

$$f_i(\mathbf{w}_i) = \frac{1}{2}\|\mathbf{w}_i\|_2^2 + C\sum_{s\in\mathcal{S}_i} \max\{\mathbf{w}_i^\top \tilde{\mathbf{x}}_s, 0\} \qquad (10)$$

where $\tilde{\mathbf{x}}_s = [\mathbf{x}_s^\top, 1]^\top$ and we include the bias term in $\mathbf{w}_i$. Each iteration, we need to solve a subproblem to update $\mathbf{w}_i$, which is essentially fitting a small SVM model on local data of agent i, with a quadratic offset [8]. The dual of this problem turns out to be a constrained quadratic programming that can be solved efficiently, for example, by interior point method.

The first experiment is carried out on the MNIST dataset, which consists of 70,000 images of handwritten digits. We extract all digits 2 and 5 to form a new dataset for binary classification, which consists of 13,303 images of size 28×28. The results is shown in top row of Fig. 2. The left, (a), plots the average classification error of all agents against the number of communication rounds. Since both methods incur the same amount of communication overhead, locally aggregated ADMM achieves much better efficiency. Also, our method shows a much smooth curve than vanilla ADMM, meaning agents achieve consensus faster and better. Fig. 2(b) shows the effects of quantization bits. Using 8 bits we can achieve comparable performance to those without quantization. Figure 2 shows the effects of quantization resolution. With enough quantization bits, the resolution does not have too much effect.

The algorithm was also tested by a classification task on a proprietary clinical dataset consisting of 3,820 patients diagnosed with Alzheimer's Disease (AD) and 14,580 patients diagnosed with mild cognitive impairment (MCI), an early stage symptom of AD. The electronic health records from Jan., 2015 to May 2019 of all patients were collected from 23 hospitals. We obtain similar results, as is shown in bottom row of Fig. 2. Figure 2(d) compares the convergence speed, Fig. 2(e) the effect of quantization bits, and Fig. 2(f) the effects of quantization resolution.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we discussed decentralized federated learning using locally aggregated ADMM, in order to satisfy privacy regulations. We use quantization to improve the communication efficiency. Future work includes to characterize the relation between solution accuracy and quantization resolution. Plus, using adaptive quantization, not equally spaced lattice, could further improve communication efficiency while maintaining the same accuracy.

6. REFERENCES

[1] An Act, "Health insurance portability and accountability," Aug. 1996.

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

[3] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 12, 2019.

[4] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu, "Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent," in *Proc. of Neural Information Processing (NeurIPS)*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, pp. 5330–5340, Curran Associates, Inc.

[5] Reza Shokri and Vitaly Shmatikov, "Privacy-Preserving Deep Learning," in *Proc. of the ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2015, pp. 1310–1321, ACM.

[6] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang, "Deep Learning with Differential Privacy," in *Proc. of the ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2016, pp. 308–318, ACM.

[7] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra, "Federated learning with non-iid data," *arXiv:1806.00582*, 2018.

[8] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Mirection Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[9] Pedro A. Forero, Alfonso Cano, and Georgios B. Giannakis, "Consensus-Based Distributed Support Vector Machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[10] Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin, "On the Linear Convergence of the ADMM in Decentralized Consensus Optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, Apr. 2014.

[11] Meng Ma, Athanasios N. Nikolakopoulos, and Georgios B. Giannakis, "Fast Decentralized Learning via Hybrid Consensus ADMM," in *Proc. of Intl. Conf. on Acoustics, Speech, and Signal Processing*, Calgary, Alberta, Canada, Apr. 2018, IEEE.

[12] Meng Ma, Athanasios N. Nikolakopoulos, and Georgios B. Giannakis, "Hybrid ADMM: A unifying and fast approach to decentralized optimization," *EURASIP Journal on Advances in Signal Processing*, vol. 2018, no. 1, pp. 73, 2018.

[13] Jakub Konečnỳ, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv:1610.02527*, 2016.

[14] Shengyu Zhu, Mingyi Hong, and Biao Chen, "Quantized consensus ADMM for multi-agent distributed optimization," in *Proc. of Intl. Conf. on Acoust., Speech, and Signal Processing*. 2016, pp. 4134–4138, IEEE.