2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)

Quality-Aware Distributed Computation for Cost-Effective Non-Convex and Asynchronous Wireless Federated Learning

Yuxi Zhao, Xiaowen Gong
Department of Electrical and Computer Engineering
Auburn University, Auburn, AL 36849
Email: {yzz0171,xgong}@auburn.edu

***Abstract*—Wireless federated learning (WFL) trains machine learning (ML) models on wireless edge devices in a distributed manner without the need of collecting data from users. In WFL, the quality of a local model update depends on the variance of the local stochastic gradient, determined by the mini-batch data size used to compute the update. In this paper, we study quality-aware distributed computation for WFL with non-convex problems and asynchronous algorithms, using mini-batch size as a "knob" to control the quality of users' local updates. We first characterize performance bounds on the training loss as a function of local updates' quality over the training process, for both non-convex and asynchronous settings. Our findings reveal that the impact of a local update's quality on the training loss 1) increases with the stepsize used for that local update for non-convex learning, and 2) increases when there are more other users' local updates which are coupled with that local update (depending on the update delays) for asynchronous learning. Based on these useful insights, we design channel-aware adaptive algorithms that determine users' mini-batch sizes over the training process, based on the impacts of local updates' quality on the training loss as well as users' wireless channel conditions (which determine the update delays) and computation costs. We evaluate the proposed quality-aware adaptive algorithms using simulations, which demonstrate improved learning accuracy and learning cost.**

***Index Terms*—Quality-aware distributed computation, wireless federated learning.**

I. Introduction

Federated learning (FL) [1] is an emerging and promising ML framework, which performs training of ML models in a distributed manner. Instead of collecting data from a potentially large number of users to a central server in the cloud for training, FL allows the data to remain at users' end devices (such as smartphone), and trains a global ML model on the server by collecting and aggregating model updates locally computed on each user's device based on her local data. One significant advantage of using FL is to preserve the privacy of individual users' data. Moreover, since only the local ML model parameters, instead of the local data, are sent to the server, the communication costs can be greatly reduced. Furthermore, FL can exploit ubiquitous smart devices with substantial computing capabilities, which are often under-utilized. In particular, when FL is used in a wireless edge network, the data samples generated at individual wireless devices can be exploited via local computation and global aggregation based on distributed ML. As a result, *wireless federated learning* (WFL) can achieve collaborative intelligence in wireless edge networks. A general consensus is that WFL can support intelligent control and management of wireless communications and networks (such as in [2]–[4]), and can enable many AI applications based on wireless networked systems.

As is standard, learning accuracy is a key performance metric for FL. The accuracy of the trained machine learning model in FL depends heavily on which users participate in the learning process and the *quality* of their local model updates. Specifically, when distributed stochastic gradient descent (SGD) is used for FL, the quality of a local stochastic gradient in each iteration can be measured by the variance of the gradient, which depends on the *mini-batch size* used to compute the gradient. It is important to observe that the quality of local updates can be treated as a design parameter and used as a *control "knob"* (via the mini-batch size) to be adapted across users and over time. Such quality-aware distributed computation can substantially improve the learning accuracy of WFL.

In this paper, we study quality-aware distributed computation for WFL, with the focuses on *non-convex* problems and *asynchronous* algorithms. The training problem of many practical ML models (e.g., deep neural networks) involves a non-convex loss function. Such a non-convex optimization problem is more difficult to solve than the convex version, due to suboptimal local minima. In addition, asynchronous learning algorithms are usually more efficient than their synchronous counterparts in utilizing users' computing capabilities, as it allows devices to keep computing without waiting for the global update received from the server as in the synchronous algorithms. This benefit of asynchronous learning is more so in a wireless setting, as there can be strong communication stragglers due to heterogeneous and time-varying wireless channels.

In this paper, our goal is to minimize the training loss while taking into account costs and constraints of computation and communication resources. In particular, we investigate how to adaptively determine participating users' mini-batch sizes over the learning process. To this end, several significant challenges need to be addressed: 1) The quality (determined by the mini-batch size) of local stochastic gradient updates can be hetero-

This work was supported by the startup fund of Xiaowen Gong, Intramural Grants Program 190599 of Auburn University, and U.S. NSF grant ECCS-2121215.

ISBN 978-3-903176-37-9 © 2021 IFIP

geneous across users and time-varying, and it is non-trivial to quantify the impacts of local updates' quality on the accuracy of the final learnt model over the learning process. 2) The non-convex and asynchronous settings of FL require new analysis different from their convex and synchronous counterparts. 3) The unique features of wireless edge networks, including time-varying wireless channels, should be taken into account. To achieve a desired tradeoff between the training loss and the training cost, local updates' quality should be determined based on the impacts of local updates on the training loss as well as users' wireless channel conditions and computation costs.

The main contributions are summarized as follows:

- We propose quality-aware distribute computation for FL in wireless edge networks, which controls the *quality* of users' local model updates via the mini-batch sizes used to compute the updates, for non-convex problems and asynchronous algorithms. Our goal is to minimize the training loss as well as users' computation and communication costs in the training process.
- We characterize performance bounds on the training loss as a function of users' local updates' quality (and thus the mini-batch sizes) over the training process, for both non-convex and asynchronous settings. Our findings reveal that the impact of a user's local update's quality on the training loss 1) increases with the stepsize used that local update for non-convex learning, and 2) increases when there are more other users' local updates which are coupled with that local update for asynchronous learning, depending on the update delays.
- Based on the obtained insights above, we develop channel-aware adaptive algorithms that determine users mini-batch sizes over the training process for both non-convex and asynchronous learning, based on the impacts of local updates' quality on the training loss as well as users' wireless channel conditions (which determine the update delays) and computation costs. We characterize the optimal mini-batch sizes, which shows that it is optimal to use larger mini-batch sizes when the local updates' impacts are larger. For the non-convex setting, we also develop a greedy algorithm that selects participating users, which achieve an approximation ratio by exploiting the non-monotone submodular property of the problem.
- We evaluate the proposed quality-aware adaptive algorithms using simulations. The results demonstrate that these algorithms outperform existing schemes in terms of the training loss.

The remainder of this paper is organized as follows. Section II reviews related work. In Section III, we describe quality-aware distributed computation for federated learning. In Section IV and Section V, we study learning accuracy bounds, and dynamic user selection and mini-batch size design based on the training loss bounds, respectively. Simulation results are provided in Section VI.

II. Related Work

Wireless Federated Learning. FL has emerged as a disruptive computing paradigm for ML by democratizing the learning process to potentially many individual users using their end devices. For WFL, the computing and networking environments have salient features, including heterogeneous and time-varying computing and communication resources that need to be accounted for. Recent studies on FL have made effort to take into account these issues [5]–[12]. For example, Tran et al [5] studied FL in wireless networks for devices with different computing and communication capabilities. In [13], Tu et al studied computation offloading based distributed learning where devices have different computing and communication resources. However, all these works have not exploited mini-batch sizes to *control the quality* of users' local model updates, and have not considered the impacts of diverse and dynamic local updates' quality on learning accuracy. A very recent work [14] has studied quality-aware distributed computation for WFL with convex problems and synchronous algorithms. However, it has not considered the non-convex and asynchronous settings which are very different and is the focus of this paper.

Non-Convex and Asynchronous Distributed Learning. With rapid recent advances in ML/AI, distributed ML has also seen substantial research activities in the past decade [15]–[17]. Many prior works have studied various settings of distributed ML [18]–[23], including for non-convex problems and asynchronous algorithms. As many ML optimization problems are non-convex, the convergence of non-convex distributed learning has been studied [19]–[21]. Along a different avenue, asynchronous distributed learning [22]–[26] has received more attention due to its high efficiency for large-scale distributed learning. However, most existing works on non-convex and asynchronous distributed learning have focused on the impacts of learning rate and delay on the convergence of the learning algorithm. In this paper, we theoretically study the impacts of local updates' quality (quantified by the variance and determined by the mini-batch size) on learning accuracy, and the mini-batch size design, which is very different from the prior works.

III. Quality-Aware Distributed Computation for Wireless Federated Learning

In this section, we present the system model of quality-aware distributed computation for FL in a wireless edge network.

Quality-Aware Distributed Computation for FL. Consider the setting where the distributed learning process of FL is carried out by a set of wireless users. The server of FL can reside in the cloud or at the edge (e.g., access point or base station of a wireless network), and the users are connected to the FL server via wireless links.

We consider the following FL problem:

$$\min_{\mathbf{w}} \; F(\mathbf{w}) \triangleq \sum_{i=1}^{N} \frac{D_i}{D} F_i(\mathbf{w}), \tag{1}$$

where $F_i(\mathbf{w})$ is the prediction loss of the model parameter $\mathbf{w}$ based on user i's local dataset, N is the number of users, $\mathcal{D}_i = \{\xi_1^i, \xi_2^i, \ldots, \xi_{D_i}^i\}$ is user i's local dataset for updating the model parameter, and $D \triangleq \sum_{i=1}^{N} D_i$. User i's local loss function $F_i(\mathbf{w})$ is defined by

ISBN 978-3-903176-37-9 © 2021 IFIP

$$F_i(\mathbf{w}) \triangleq \frac{1}{D_i} \sum_{m=1}^{D_i} f_i(\mathbf{w}; \xi_m^i),$$

where $f(\cdot)$ is the per-sample loss function. In each round of FL, K out of N users are selected from the user set $\mathcal{N}$ to compute local updates, communicate their local updates to the server, and receive the updated global model from the server. At round t, a selected user i computes the average gradient g_{t-1}^i of the loss function using a set of D_t^i data samples randomly drawn from her local dataset $\mathcal{D}_i$, based on the global model $\mathbf{w}_{t-1}$ received from the previous round $t-1$, and update her local model as

$$\mathbf{w}_t^i = \mathbf{w}_{t-1} - \eta g_t^i,$$

where

$$g_t^i \triangleq \frac{1}{D_t^i} \sum_{j=1}^{D_t^i} \nabla f(\mathbf{w}, \xi_t^{i,j}),$$

η is the stepsize, and $\xi_t^{i,j}$ is the jth data sample randomly drawn from user i's local dataset $\mathcal{D}_i$. At the end of round t, the server aggregates K users' local models and updates the global model as

$$\mathbf{w}_t = \sum_{i=1}^{K} \frac{D_t^i}{D_t} \mathbf{w}_t^i, \tag{2}$$

where $D_t \triangleq \sum_{i=1}^K D_t^i$.

The *quality* of a user's local update is captured by the variance of the local stochastic gradient, given by

$$q_i \triangleq E\left[\left\|g_t^i - \bar{g}_t\right\|^2\right], \tag{3}$$

where $\bar{g}_t \triangleq E[g_t^i]$. Assume that the loss function f satisfies $E\left\|\nabla f_i\left(\mathbf{w}_t, \xi_m^i\right) - E[\nabla f_i\left(\mathbf{w}_t\right)]\right\|^2 \leq \sigma^2, \forall t$. It can be shown that [27]

$$E\left[\left\|g_t^i - \bar{g}_t\right\|^2\right] \leq \frac{\sigma^2}{D_t^i}.$$

Note that a user's quality is determined by the *mini-batch size* D_t^i used to update her local model. Thus, a local update computed with a larger mini-batch size has higher quality.

In this paper, we assume that users' local data are IID. Our results in the following sections (including the training loss bounds, adaptive mini-batch size design and user selection) can be extended to the case of non-IID data, and will be studied in our future work.

Asynchronous FL. The Fl algorithm can be carried out in an asynchronous manner described as follows. In this case, the learning process consists of rounds, each lasting for a time period of the same length. At the beginning of each round, the server broadcasts the global model. At the end of each round, the server updates the global model using the local models received in the round as (2) (as illustrated in Fig. 1). Note that a user's local update can be received by the server in a round different from the round when the user receives the global model from the server for computing that local update. The update delay τ_i quantifies the difference between the round when user i receives the global model from the server and the round when user i's local update computed from that global model is received by the server. Note that τ_i is an integer, where $\tau_i = 1$ means there is no update delay, and $\tau_i > 1$ means

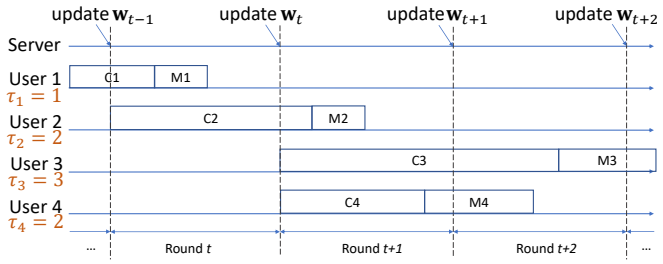

Fig. 1. Schedule of the server's updates and users' computations (C) and communications (M) in asynchronous FL. The server updates and broadcasts the global model at the end of each round and the beginning of the next round, respectively. τ_i is the update delay of user i. We ignore the server's computation and communication times for simplicity.

there is an update delay. The computation time (Ci) is the time it takes for user i to *compute* (update) her local model, and the communication time (Mi) is the time it takes for user i to *communicate* (upload) her updated local model to the server. For example, in Fig. 1, user 2 receives the global model and starts to update her local model at the beginning of round t. After computing the local model, user 2 starts to uploads her local model to the server. Then the server receives user 2's local model $\mathbf{w}_{t+1}^2$ in round $t+1$ and updates the global model as $\mathbf{w}_{t+1} = \mathbf{w}_{t+1}^2$ at the end of round $t+1$.

FL in Wireless Edge Network. A user incurs a computation cost (measured by the computation delay, energy consumption, etc) for computing a local update, which depends on the computation capability of the user's device and the mini-batch size used to compute the update. Let $c_{p,t}^i$ be user i's cost of computing her local update using one data sample in round t. Besides the computation cost, a user also incurs communication cost for communicating local updates to the server (measured by the communication delay, energy consumption, etc), which depends on the user's wireless channel condition. Let $c_{m,t}^i$ be user i's communication cost in round t. Note that the computation cost $c_{p,t}^i$ and the computation cost $c_{m,t}^i$ generally vary across users and over rounds of the FL algorithm.

IV. Training Loss Bound Analysis

In this section, under the quality-aware FL framework of the previous section, we study the training loss bounds for three settings: 1) non-convex problems; 2) asynchronous algorithms; 3) non-convex problems and asynchronous algorithms. We will first characterize the performance bounds as functions of users' mini-batch sizes over the training process. Based on the obtained results, we then discuss the impacts of mini-batch sizes and other system parameters (including stepsize) on the training loss.

A. Case of Non-Convex Learning

We first analyze the training loss bound for non-convex problems with synchronous algorithms. For non-convex optimization, the metrics for convex optimization (e.g., $F(\mathbf{w}_T) - F(\mathbf{w}^*)$) are not suitable, since it is hard to find the global optimum for non-convex optimization problems. Thus, we analyze the ergodic convergence [20], where we randomly select an index $\tilde{t}$ from $\{1, 2, \ldots, T\}$ with probability $\{\eta_t / \sum_{t=1}^T \eta_t\}$, and use $\mathbf{w}_{\tilde{t}}$

ISBN 978-3-903176-37-9 © 2021 IFIP

as the final model of the training process. The expected squared gradient norm $\|\bar{g}_{\tilde{t}}\|^2$ of $\mathbf{w}_{\tilde{t}}$ can be upper bounded as follows.

Theorem 1: Suppose F is L-smooth, and $E\left\|\nabla f_i\left(\mathbf{w}_t, \xi_m^i\right) - E[\nabla f_i(\mathbf{w}_t)]\right\|^2 \leq \sigma^2$, $\forall i, t$, take the stepsize $\eta_t \leq \frac{1}{L}$, $\forall t$, the ergodic convergence is given by

$$\frac{1}{\sum_{t=1}^T \eta_t} \sum_{t=1}^T \eta_t \|\bar{g}_t\|^2 \leq \frac{2E(F(\mathbf{w}_0) - F(\mathbf{w}^*)) + \sum_{t=1}^T L\eta_t^2 \frac{\sigma^2}{D_t}}{\sum_{t=1}^T \eta_t}, \tag{4}$$

where $D_t \triangleq \sum_{i \in S_t} D_t^i$, and D_t^i is the mini-batch size of user i in round t.

Due to space limitation, all the proofs of results in this paper are provided in our online technical report [28].

Remark 1: In (4), $\frac{\sigma^2}{D_t}$ is the upper bound of the variance of the global model update $\mathbf{w}_t$, which is determined by the variances of participating users' local model updates. Thus the weight $L\eta_t^2 / \sum_{t=1}^T \eta_t$ of the variance $\frac{\sigma^2}{D_t}$ of the global update captures the impact of the quality of local updates on the training loss. We also know that a larger stepsize results in a larger change of the model. Thus, given the total number of rounds T and the sum of stepsizes $\sum_{t=1}^T \eta_t$, the larger the stepsize of the round, the larger the impact of local updates' quality on the training loss. Also observe that with any sequence of $\{\eta_t\}$ such that $\sum_{t=1}^T \eta_t$ diverge and $\sum_{t=1}^T \eta_t^2$ converge (e.g., $\eta_t = 1/t$), and with any non-decreasing mini-batch size, the bound converges to 0 as $T \to \infty$.

Taking a closer look at Theorem 1, we can properly choose the stepsize and total mini-batch size in each round and obtain the following convergence rate:

Proposition 1: Suppose F is L-smooth, $E\left\|\nabla f_i\left(\mathbf{w}_t, \xi_m^i\right) - E[\nabla f_i(\mathbf{w}_t)]\right\|^2 \leq \sigma^2$, and $E\left\|\nabla F_i(\mathbf{w}_t)\right\|^2 \leq B^2$, $\forall i, t$, take any mini-batch size $\{D_t\}$ and a constant stepsize $\eta = \frac{\sqrt{D_{\min}}}{L\sqrt{T}}$, where $D_{\min} \triangleq \min\{D_t\}$, the ergodic convergence is given by

$$\frac{1}{T} \sum_{t=1}^T \|\bar{g}_t\|^2 \leq \frac{2LE(F(\mathbf{w}_0) - F(\mathbf{w}^*)) + \sigma^2}{\sqrt{D_{\min} T}}. \tag{5}$$

Proposition 1 shows that when the stepsize is small enough, the convergence rate achieves $O(1/\sqrt{D_{\min} T})$, which is consistent with the result obtained in [19], Corollary 1.

B. Case of Asynchronous Learning

We then analyze the training loss bound when asynchronous algorithms are used for convex problems. In this subsection, for ease of exposition, we focus on using a constant stepsize. Our results can be extended to using a time-varying stepsize.

Theorem 2: Suppose F is L-smooth and μ-strongly convex, $E\left\|\nabla f_i\left(\mathbf{w}_t, \xi_m^i\right) - E[\nabla f_i(\mathbf{w}_t)]\right\|^2 \leq \sigma^2$, and $E\left\|\nabla F_i(\mathbf{w}_t)\right\|^2 \leq B^2$, $\forall i, t$, take the stepsize $\eta \leq \frac{1}{L}$, then the training loss is bounded by

$$\begin{aligned} E[F(\mathbf{w}_T) - F(\mathbf{w}^*)] &\leq (1-\mu\eta)^T (F(\mathbf{w}_0) - F(\mathbf{w}^*)) \\ &+ \sum_{t=1}^T \left[(1-\mu\eta)^{T-t} \left(L^2\eta^3 \sum_{i \in M_t} \frac{D_t^i}{D_t} \Big(\sum_{t'=t-\tau_i+1}^{t-1} \frac{\sigma^2}{D_{t'}} \right.\right. \\ &\left.\left. + \Gamma(\tau_i - 1)B^2 \Big) + \frac{\eta}{2} \frac{\sigma^2}{D_t} \right) \right], \end{aligned} \tag{6}$$

where M_t is the set of users who update at time t, Γ is the maximum update delay.

Remark 2: Theorem 2 shows that the training loss bound consists of two terms. The first term decreases geometrically with the number of rounds T, and is due to that SGD in expectation makes progress towards the optimal solution. The second term of the bound is caused by the randomness of data sampling for computing the update in SGD. Compared to the training loss in the case of synchronous learning [14], each term in the second term not only depends on the total mini-batch size D_t of users who finish uploading their local models in the current round t, but also the total mini-batch size $D_{t'}$ of each past round t' such that $t' \in \{t - \tau_i + 1, \ldots, t-1\}$, where $i \in M_t$. For example, in Fig. 1, the training loss in round $t+1$ not only depends on user 2's mini-batch size, but also on user 1's mini-batch size. This implies that in each round, the larger update delays of users, the worse the training loss.

Remark 3: According to (6), the training loss due to users' update delays in each round is $\sum_{i \in M_t} \frac{D_t^i}{D_t} (\sum_{t'=t-\tau_i+1}^{t-1} \frac{\sigma^2}{D_{t'}} + \Gamma(\tau_i - 1)B^2)$. When there is no update delay, i.e., $\tau_i = 1, \forall i \in M_t$, this term is 0, and the training loss bound degenerates to the case of synchronous learning in [14]. When update delays exist, this term decreases as the mini-batch sizes of users who upload their local models in the past rounds increase.

To obtain some useful insights, we next focus on the special case where only one user uploads her local model in a round. Let user t be the user who uploads her local model in round t. Then, using Theorem 2, the training loss bound is given by

$$\begin{aligned} &E[F(\mathbf{w}_T) - F(\mathbf{w}^*)] \leq (1-\mu\eta)^T (F(\mathbf{w}_0) - F(\mathbf{w}^*)) + \\ &\sum_{t=1}^T \left[(1-\mu\eta)^{T-t} \left(L^2\eta^3 \Big(\sum_{t'=t-\tau_t+1}^{t-1} \frac{\sigma^2}{D_{t'}} + \Gamma(\tau_t - 1)B^2 \Big) + \frac{\eta}{2} \frac{\sigma^2}{D_t} \right) \right]. \end{aligned} \tag{7}$$

To analyze the impact of user t on the training loss, we find the terms determined by D_t and τ_t in (7) as follows.

$$\begin{aligned} I_t = &(1-\mu\eta)^{T-t} \left(L^2\eta^3 \Big(\sum_{t'=t-\tau_t+1}^{t-1} \frac{\sigma^2}{D_{t'}} + \Gamma(\tau_t - 1)B^2 \Big) \right. \\ &\left. + \frac{\sigma^2}{D_t} \left(\frac{\eta}{2} + L^2\eta^3 \sum_{\tau'=1}^{\Gamma} \mathbf{1}_{\tau_{t+\tau'} \geq \tau'} (1-\mu\eta)^{-\tau'} \right) \right), \end{aligned} \tag{8}$$

where $\mathbf{1}$ is an indicator function such that $\mathbf{1} = 1$ when the user who uploads her local model in round $t+\tau'$ receives the global model from the server before round t, and $\mathbf{1} = 0$ otherwise.

Remark 4: From (8), we can see that when user t's update delay τ_t is independent of her mini-batch size D_t, I_t decreases as D_t increases and/or τ_t decreases. This implies that in a given round, a user with a larger mini-batch size or a smaller update delay reduces the training loss. Also observe that user t's mini-batch size affects some later rounds such that users who uploads their local models in those rounds receive the global models before round t.

Remark 5: In (8), the weight of user t's impact on the training loss is $(1-\mu\eta)^{T-t}$. Note that this weight increases with the round number t as $1 - \mu\eta < 1$. Therefore, the update delay and mini-batch size in a later round have larger impacts on

ISBN 978-3-903176-37-9 © 2021 IFIP

the training loss than in an earlier round. This observation has important implications: it is better for a user to have a larger mini-batch size or a smaller update delay in a later round rather than an earlier round to reduce the training loss.

C. Case of Non-Convex and Asynchronous Learning

Next, we analyze the training loss bound for non-convex problems and asynchronous algorithms.

Theorem 3: Suppose F is L-smooth and $E\left\|\nabla f_i\left(\mathbf{w}_t, \xi_m^i\right) - E[\nabla f_i(\mathbf{w}_t)]\right\|^2 \leq \sigma^2$, $\forall i, t$, take the stepsize $\eta_t \leq \frac{1}{L}$, $\forall t$, the ergodic convergence is given by

$$\frac{1}{\sum_{t=1}^T \eta_t} \sum_{t=1}^T \eta_t \|\bar{g}_t\|^2 \leq \frac{2E(F(\mathbf{w}_0) - F(\mathbf{w}^*))}{\sum_{t=1}^T \eta_t} + \frac{\sum_{t=1}^T (L\eta_t^2 \frac{\sigma^2}{D_t} + 2L^2\eta_t \sum_{i\in M_t} \frac{D_t^i}{D_t} \sum_{t'=t-\tau_i+1}^{t-1} \eta_{t'}^2 \frac{\sigma^2}{D_{t'}})}{\sum_{t=1}^T \eta_t}, \tag{9}$$

where the stepsize satisfies $L\eta_t + L^2\Gamma\eta_t \sum_{t'=1}^{\Gamma-1} \eta_{t+t'} \leq 1$, $\forall t$.

Remark 6: From (9), we can see that the convergence rate for the combined non-convex and asynchronous setting shows similar properties as that for each of the two settings. First, the convergence rate not only depends on the total mini-batch size D_t used in the current round t, but also on the mini-batch sizes used in several past rounds. Second, given the total number of rounds T and the sum of stepsizes $\sum_{t=1}^T \eta_t$, the larger the stepsize of the round, the larger the impact of local updates' quality on the training loss. Also observe that the impact of the quality $\frac{\sigma^2}{D_t}$ of the local updates in round t is captured not only by the stepsize η_t in round t but also by the stepsizes in several later rounds.

V. Channel-Aware Adaptive User Selection and Mini-Batch Size Design

In this section, we study how to design users' mini-batch sizes over the training process to minimize the training loss bound for non-convex and asynchronous FL, respectively. For the non-convex setting, we also investigate how to select participating users. In the meanwhile, we take into account users' computation and communication costs. Note that we consider continuous-valued mini-batch size D_t in our theoretical analysis, which can be converted back to the nearest integer values when used in practice.

A. Case of Non-Convex Learning

First we study the optimal user selection and mini-batch size design for non-convex problems with synchronous algorithms. We aim to minimize the sum of the training loss and users' total communication and computation cost. The optimization problem can be formulated as

$$\min_{\{S_t\},\{D_t\}} \ \gamma \frac{1}{\sum_{t=1}^T \eta_t} \sum_{t=1}^T \eta_t \|\bar{g}_t\|^2 + (1-\gamma) \sum_{t=1}^T \sum_{i\in S_t} (c_{p,t}^i D_t^i + c_{m,t}^i),$$
$$s.t.\ D_t^i \leq D_B^i, \forall i, t,$$

where $\{S_t\}$ is the set of selected users in all rounds, $\{D_t\}$ is the set of assigned mini-batch sizes of users in all rounds, D_t is the mini-batch size of the user who updates her local model in round t, $\gamma \in (0, 1]$ is the weight that balances the training loss and the cost, which can be determined according to the server's concern, and D_B^i is user i's maximum possible mini-batch size.

From (4), we rewrite the problem as follows:

$$\min_{\{S_t\},\{D_t\}} \ \sum_{t=1}^T \mathcal{J}_1(S_t, D_t) = \gamma \sum_{t=1}^T \frac{L\eta_t^2 \frac{\sigma^2}{D_t}}{\sum_{t=1}^T \eta_t} + (1-\gamma) \sum_{t=1}^T \sum_{i\in S_t} (c_{p,t}^i D_t^i + c_{m,t}^i),$$
$$s.t.\ D_t^i \leq D_B^i, \forall i, t,$$

Since $\sum_{t=1}^T \eta_t$ is given, we can see that the problem above can be decomposed into T independent problems, each for one of the T rounds. Thus, we focus on finding the optimal S_t and D_t for a single round t. Moreover, we decompose the problem in round t into two subproblems: 1) we first study the optimal mini-batch size design given any user selection; 2) then we study the optimal user selection given the optimal mini-batch size design.

1) Optimal Mini-Batch Size Design: Given any set of selected users, since the total communication cost is fixed, a user with a lower computation cost is preferred over one with a higher computation cost. Hence, the optimal mini-batch sizes are determined in the ascending order of users' computation costs until the minimum value of $\mathcal{J}_1$ is reached. It can be shown that $\mathcal{J}_1$ is a convex function of D_t^i when other users' mini-batch sizes are given. The following result characterizes users' optimal mini-batch sizes.

Theorem 4: Let users in S_t be ordered as $c_{p,t}^1 \leq c_{p,t}^2 \leq \cdots \leq c_{p,t}^{|S_t|}$. Then the users' optimal mini-batch sizes are determined iteratively in this order, where the ith user's optimal mini-batch size is given by

$$D_t^{i*}(S_t) = \min\{D_B^i, \max\{\sqrt{\frac{L\gamma\eta_t^2\sigma^2}{(1-\gamma)c_{p,t}^i \sum_{t-1}^T \eta_t}} - \sum_{j=1}^{i-1} D_t^{j*}, 0\}\}.$$

2) User Selection Algorithm: With the optimal mini-batch size design, the user selection problem in round t can be rewritten as

$$\min_{S_t} \ \mathcal{J}_1(S_t, D_t^*(S_t)) = \gamma \left(\frac{L\eta_t^2\sigma^2}{\sum_{i\in S_t} D_t^{i*}(S_t) \sum_{t=1}^T \eta_t}\right) + (1-\gamma) \sum_{i\in S_t} (c_{p,t}^i D_t^{i*}(S_t) + c_{m,t}^i). \tag{10}$$

First note that problem (10) can be cast as the maximization problem of $-\mathcal{J}_1(S_t, D_t^*(S_t))$. Then we have the following property of the problem.

Lemma 1: $-\mathcal{J}_1(S_t, D_t^*(S_t))$ is a negative non-monotone submodular function.

Note that it is difficult to solve a negative non-monotone submodular maximization problem with performance guarantee (e.g., with an approximation ratio). To overcome this challenge, we transform the above problem into a non-negative non-monotone submodular maximization problem.

First, we find the maximum value of $\mathcal{J}_1(S_t, D_t^*(S_t))$. From the optimal mini-batch size design, we can see that there exists some user j, such that for any user $j' \in S_t$ that has $c_{p,t}^{j'} \geq c_{p,t}^j$, we have $D_t^{j'*}(S_t) = 0$. The objective function

ISBN 978-3-903176-37-9 © 2021 IFIP

$\mathcal{J}_1$ decreases as users' optimal mini-batch sizes are determined iteratively according to Proposition 4 until user j, and does not change when the optimal mini-batch sizes of users after j are determined. Therefore, given a selected user set S_t, $\mathcal{J}_1$ is maximized when only one user's mini-batch size is non-zero. Thus, for any $S \subseteq \mathcal{N}$, the maximum value of $\mathcal{J}_1$ is given by

$$\mathcal{J}_{1,\max} = \frac{\gamma L \eta_{\max}^2 \sigma^2}{\sum_{t=1}^{T} \eta_t} + (1-\gamma)(c_{p,\max} D_{B,\max} + N c_{m,\max}),$$

where $\eta_{\max} = \max\{\eta_t | \forall t\}$, $c_{p,\max} = \max\{c_{p,t}^i | \forall i,t\}$, $D_{B,\max} = \max\{D_B^i | \forall i\}$, and $c_{m,\max} = \max\{c_{m,t}^i | \forall i,t\}$.

Based on the maximum value of $\mathcal{J}_1(S_t, D_t^*(S_t))$, we define a new function $\mathcal{G}(S_t)$ and rewrite the user selection problem in round t as

$$\max_{S_t} \ \mathcal{G}(S_t) \triangleq \begin{cases} -\mathcal{J}_1(S_t, D_t^*(S_t)) + \mathcal{J}_{1,\max}, & \text{if } S_t \neq \emptyset \\ 0, & \text{if } S_t = \emptyset \end{cases}$$

Note that in each round t of the FL algorithm, at least one user is selected (i.e., $|S_t| > 0$) with a positive mini-batch size (i.e., $D_t > 0$). This is because when $S_t = \emptyset$, the global model is not updated so that round t should not be counted as a round of the FL algorithm. Thus, we can define that $\mathcal{G}(\emptyset) \triangleq 0$. We can see that the above two problems are equivalent since $\mathcal{J}_{1,\max}$ is a constant. Next, we focus on finding the solution that maximizes $\mathcal{G}(S_t)$. From Lemma 1, it follows directly that $\mathcal{G}(S_t)$ is a non-negative non-monotone submodular function.

Next, we can apply the DeterministicUSM Algorithm [29] to solve the user selection problem, which is given in Algorithm 1. The approximation ratio of DeterministicUSM is given by the following lemma.

Lemma 2: [29] Algorithm DeterministicUSM is a $\frac{1}{3}$-approximation algorithm for maximizing function $\mathcal{G}(S_t)$.

Given the result above, we then show the approximation ratio of Algorithm 1 for our problem, under the optimal mini-batch size design given in Proposition 4. The proof of the following result is given in the appendix.

Theorem 5: Under the optimal mini-batch size design, with the user selection given by Algorithm 1, the system loss is upper bounded by

$$\sum_{t=1}^{T} \mathcal{J}_1(S_t^*, D_t^*(S^*(t))) \leq \frac{1}{3}\mathcal{OPT} + \mathcal{O}(T),$$

where $\mathcal{OPT}$ is the system loss of the optimal user and mini-batch size selection.

B. Case of Asynchronous Learning

We next study the case of asynchronous algorithms for convex problems. In this paper, we assume that the update delays[1] of users are fixed, regardless of the mini-batch sizes used to compute the local updates. This is a reasonable assumption when the communication times of local updates are relatively large compared to their computation times. The case where update delays depend on the mini-batch sizes is a very challenging problem, and will be studied in our future work. For ease of exposition, in this subsection, we also assume that only one user uploads her local model in one round.

[1]Note that the update delay is an integer and is not the total computation and communication time of the local update.

Algorithm 1: Approximate optimal user selection

```
X_0 ← ∅, Y_0 ← N;
for i = 1, 2, ··· , N do
    a_i ← G(X_{i-1} ∪ {i}) − G(X_{i-1});
    b_i ← G(Y_{i-1} \ {i}) − G(Y_{i-1});
    if a_i ≥ b_i then
        X_i ← X_{i-1} ∪ {i}, Y_i ← Y_{i-1};
    else
        X_i ← X_{i-1}, Y_i ← Y_{i-1} \ {i};
S_t ← X_N;
return S_t.
```

As discussed earlier, the impact of each user t's mini-batch size on the training loss depends on the set of other users' computation and communication periods during which user t uploads her local model. We need to take into account this coupling of users' updates in terms of their impacts on the training loss, based on the update delays over the training process. Thus, we develop an adaptive algorithm that determines the mini-batch size for each user's each update, based on the delay information of only Γ number of updates in the future.

When users' update schedules are given[2], the total communication cost is fixed. Thus it suffices to minimize the sum of the training loss bound and users' total computation cost:

$$\begin{aligned} &\min_{\{D_t\}} \ \gamma E[F(\mathbf{w}_T) - F(\mathbf{w}^*)] + (1-\gamma)\sum_{t=1}^{T} c_{p,t} D_t, \\ &s.t.\ D_t \leq D_B^t, \forall t, \end{aligned} \tag{11}$$

where D_B^t is user t's maximum possible mini-batch size. Since the first term of the training loss bound given in (7) does not depend on users' mini-batch sizes $\{D_t\}$, it suffices to minimize the second term. Furthermore, we rewrite the second term of (6) as the sum of terms that are determined by user t's update delay and mini-batch size. Thus, from (7), we rewrite (11) as

$$\begin{aligned} \min_{\{D_t\}} \ \sum_{t=1}^{T} \mathcal{J}_2(D_t) = \sum_{t=1}^{T} \Big(&\gamma(1-\mu\eta)^{T-t} \Big(L^2\eta^3 \Gamma(\tau_t - 1) B^2 \\ &+ \frac{\eta}{2}\frac{\sigma^2}{D_t} + L^2\eta^3 \frac{\sigma^2}{D_t} \sum_{\tau'=1}^{\Gamma} \mathbf{1}_{\tau_{t+\tau'} \geq \tau'} (1-\mu\eta)^{-\tau'} \Big) \\ &+ (1-\gamma) c_{p,t} D_t \Big), \\ s.t.\ D_t \leq D_B^t, \forall t. \end{aligned} \tag{12}$$

where $\tau_{t+\tau'}$, $\forall \tau' \in \{1, \ldots, \Gamma\}$ are known. Thus, for user t who uploads her local model in round t, the server can find the value of $\sum_{\tau'=1}^{\Gamma} \mathbf{1}_{\tau_{t+\tau'} \geq \tau'} (1-\mu\eta)^{-\tau'}$ based on the update delays of users who upload their local models in next Γ rounds.

Note that problem (12) can be decomposed into T independent problems, each for one of the T users. Thus, we focus on finding the optimal D_t that minimizes $\mathcal{J}_2(D_t)$ for a single

[2]Note that the training loss depends heavily on users' update schedules. We focus on the design of users' mini-batch size given users' update schedules here and will study the joint design of user selection and mini-batch size in future work.

ISBN 978-3-903176-37-9 © 2021 IFIP

user t. It can be shown that $\mathcal{J}_2(D_t)$ is a convex function of D_t. Thus the optimal mini-batch size D_t^* can be found as the local minimum of $\mathcal{J}_2(D_t)$, given as below.

Theorem 6: Given users' update schedules and their update delays, the optimal mini-batch size for each user t who uploads her local model in round t is given by

$$D_t^* = \min\{\sqrt{\frac{\gamma v_t \sigma^2}{(1-\gamma)c_{p,t}}}, D_B^t\},$$

where $v_t = (1-\mu\eta)^{T-t}(\frac{\eta}{2} + L^2\eta^3 \sum_{\tau'=1}^{\Gamma} \mathbf{1}_{\tau_{t+\tau'} \geq \tau'}(1-\mu\eta)^{-\tau'})$.

Theorem 6 shows that the optimal mini-batch size D_t^* is larger in a later round. This is because the weight $(1-\mu\eta)^{T-t}$ of an update on the training loss bound increases with the round number t, so that D_t^* also increases with the round number. Also note that D_t^* increases as the number of users who receive the global models before round t and finish uploading their local models after round t increases. This is because the impact of the mini-batch size of user t on the training loss increases when there are more of those users.

VI. Performance Evaluation

In this section, we conduct synthetic data simulations to validate the theoretical findings. We implement a simulated system consisting of a virtual server and a number of virtual users. For convex optimization problem, we generate 10000 data samples according to the linear model, i.e., $y = \mathbf{w}^T\mathbf{x}$, and use the mean square error function as the loss function, i.e., $f(\mathbf{w}, \xi) = \frac{1}{2}\|y - \mathbf{w}^T\mathbf{x}\|^2$, where $\xi = (\mathbf{x}, y)$ is a data sample. Each data point consists of 10 features and 1 label. For non-convex optimization problems, we generate 10000 data samples for a binary linear classification model of which the data is generated according to $\Pr(y = 1|\mathbf{x} = \boldsymbol{x}) = \lambda(\langle \mathbf{w}, \boldsymbol{x}\rangle)$ [30], where $\lambda(\langle \mathbf{w}, \boldsymbol{x}\rangle) = (1+e^{-\lambda})^{-1}$. Each data point consists of 10 features and 1 label. The stepsize η is set as a constant for all settings.

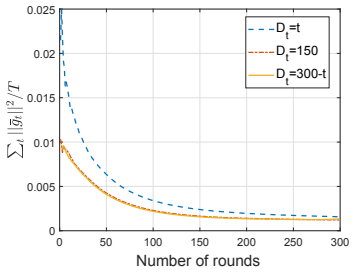

Fig. 6. Impact of mini-batch size on the training loss of synchronous FL for non-convex optimization.

We first evaluate the performance of the case of non-convex loss function with synchronous learning. We compare the training loss using time-invariant, descending and ascending mini-batch sizes to update the global model over rounds. The average mini-batch size over all rounds are the same for above three distributions to achieve fair comparison. Fig. 6 shows that although the mini-batch sizes over time of three distributions are different, they result in the same training loss at the end of training. The case of ascending mini-batch size has the worst learning accuracy in beginning rounds, and the case of descending mini-batch size has the best learning accuracy in beginning rounds. This is because, in beginning rounds, the case of descending mini-batch size uses more data to update the FL model. In our theoretical result, we have shown that using larger mini-batch to update implies better quality and leads to a lower training loss. Moreover, since the weight of local updates' quality (stepsize η) is the same in all rounds, the impact of local updates' quality on the training loss is the same. In ending rounds, as the total mini-batch size over rounds converges to the same for three cases, the training loss converges to the same value.

We next evaluate the performance of the case of convex loss function with asynchronous learning. We first compare the training loss using time-invariant, descending and ascending mini-batch sizes to update the global model. Different from the case of non-convex loss function with synchronous learning, Fig. 2 shows that although the average mini-batch sizes over time of three distributions are the same, different distributions of the mini-batch size result in different training loss at the end of training. The case of ascending mini-batch size has the worst learning accuracy in beginning rounds and results in the best learning accuracy in ending rounds. This conforms the result from Theorem 2 that the update in a later round has a larger impact on the learning accuracy. We also compare the training loss while users' maximum update delays are different ($\Gamma \in \{1, 4, 8\}$). We simulate for 50 local iterations in total, with the mini-batch size in each local iteration set as 25. From Fig. 3, we can see that when users update without delay ($\Gamma = 1$), the system suffers the lowest training loss. The training loss increases as the maximum update delay Γ increases. we can see that the simulation result is consistent with the result given in Theorem 2. FL suffers a larger error caused by the delay of users' updates.

Lastly, we evaluate the training loss of the case of non-convex loss function with asynchronous learning. Same as the other two cases, we first compare the training loss using time-invariant, descending and ascending mini-batch sizes to update the global model when the maximum update delay $\Gamma = 4$. Fig. 4 shows that even with the update delay, the three distributions result in the same training loss at the end of training which is the same as the case of non-convex loss function with synchronous learning. We then compare the training loss while users' maximum update delays are different ($\Gamma \in \{1, 4, 8\}$). In Fig. 5, we can see that the result is similar to that of the case of convex loss function with asynchronous learning.

VII. Conclusion and Future Work

In this paper, we have studied quality-aware distributed computation for WFL with non-convex problems and asynchronous algorithms. We have characterized the performance bounds on the training loss as a function of users' local updates' quality over the training process, for both non-convex and asynchronous settings. The results show that the impact of a local update's quality 1) increases with the stepsize used in the round for non-convex learning, and 2) increases when there are more other users' local updates (depending on the update delays) which are coupled with that local update for asynchronous learning. We have also developed channel-aware

ISBN 978-3-903176-37-9 © 2021 IFIP

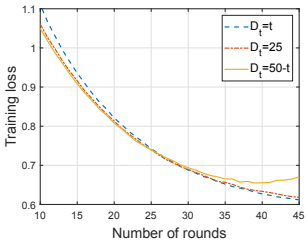

Fig. 2. Impact of mini-batch size on the training loss of asynchronous FL for convex optimization.

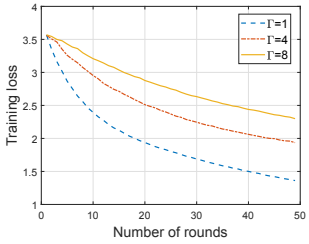

Fig. 3. Impact of maximum update delay on the training loss of asynchronous FL for convex optimization.

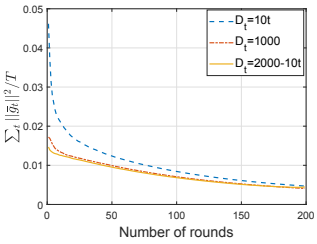

Fig. 4. Impact of mini-batch size on the training loss of asynchronous FL for non-convex optimization.

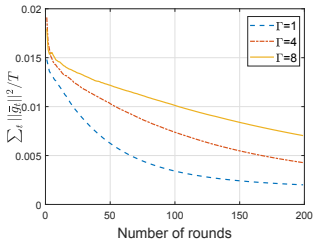

Fig. 5. Impact of maximum update delay on the training loss of asynchronous FL for non-convex optimization.

adaptive algorithms that select participating users and determine their mini-batch sizes. Simulations have been used to evaluate the proposed algorithms.

For future work, one interesting direction is to consider non-convex and asynchronous WFL where users have non-IID local data. In this case, the optimal mini-batch size and user selection design can be very different from in the IID data setting.

REFERENCES

[1] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

[2] B. Zhu, J. Wang, L. He, and J. Song, "Joint transceiver optimization for wireless communication phy using neural network," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1364–1373, 2019.

[3] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1248–1261, 2019.

[4] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1277–1290, 2019.

[5] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2019.

[6] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, 2020.

[7] J. Ren, Y. He, D. Wen, G. Yu, K. Huang, and D. Guo, "Scheduling for cellular federated edge learning with importance and channel awareness," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7690–7703, 2020.

[8] Z. Yang, M. Chen, W. Saad, C. S. Hong, M. Shikh-Bahaei, H. V. Poor, and S. Cui, "Delay minimization for federated learning over wireless communication networks," in *International Conference on Machine Learning, Workshop on Federated Learning*, 2020.

[9] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Transactions on Wireless Communications*, 2020.

[10] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2020.

[11] S. Wang, M. Lee, S. Hosseinalipour, R. Morabito, M. Chiang, and C. G. Brinton, "Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2021.

[12] J. Zhang, N. Li, and M. Dedeoglu, "Federated learning over wireless networks: A band-limited coordinated descent approach," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2021.

[13] Y. Tu, Y. Ruan, S. Wang, S. Wagle, C. G. Brinton, and C. Joe-Wang, "Network-aware optimization of distributed learning for fog computing," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2020.

[14] Y. Zhao and X. Gong, "Quality-aware distributed computation and user selection for cost-effective federated learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2021.

[15] O. Shamir and N. Srebro, "Distributed stochastic optimization and learning," in *Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2014, pp. 850–857.

[16] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 328–339.

[17] S. U. Stich, "Local sgd converges fast and communicates little," in *International Conference on Learning Representations (ICLR)*, 2019.

[18] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations*, 2019.

[19] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 5693–5700.

[20] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[21] H. Yu and R. Jin, "On the computation and communication complexity of parallel sgd with dynamic batch sizes for stochastic non-convex optimization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7174–7183.

[22] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, 2015, pp. 2737–2745.

[23] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[24] F. Niu, B. Recht, C. Ré, and S. J. Wright, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," *arXiv preprint arXiv:1106.5730*, 2011.

[25] L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari, "Asynchronous parallel algorithms for nonconvex optimization," *Mathematical Programming*, pp. 1–34, 2019.

[26] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3043–3052.

[27] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *The Journal of Machine Learning Research*, vol. 13, pp. 165–202, 2012.

[28] Technical report. [Online]. Available: https://www.dropbox.com/s/gmqdlrxdysl1xnt/TR.pdf?dl=0

[29] N. Buchbinder, M. Feldman, J. Seffi, and R. Schwartz, "A tight linear time (1/2)-approximation for unconstrained submodular maximization," *SIAM Journal on Computing*, vol. 44, no. 5, pp. 1384–1402, 2015.

[30] S. Mei, Y. Bai, A. Montanari *et al.*, "The landscape of empirical risk for nonconvex losses," *Annals of Statistics*, vol. 46, no. 6A, pp. 2747–2774, 2018.

ISBN 978-3-903176-37-9 © 2021 IFIP