Dynamic Federated Learning

Elsa Rizk *, Stefan Vlaski †, and Ali H. Sayed ‡
Institute of Electrical Engineering, Ecole Polytechnique Federal de Lausanne
Switzerland
Email: *elsa.rizk@epfl.ch, †stefan.vlaski@epfl.ch, ‡ali.sayed@epfl.ch

Abstract—Federated learning has emerged as an umbrella term for centralized coordination strategies in multi-agent environments. While many federated learning architectures process data in an online manner, and are hence adaptive by nature, most performance analyses assume static optimization problems and offer no guarantees in the presence of drifts in the problem solution or data characteristics. We consider a federated learning model where at every iteration, a random subset of available agents perform local updates based on their data. Under a non-stationary random walk model on the true minimizer for the aggregate optimization problem, we establish that the performance of the architecture is determined by three factors, namely, the data variability at each agent, the model variability across all agents, and a tracking term that is inversely proportional to the learning rate of the algorithm. The results clarify the trade-off between convergence and tracking performance.

Index Terms—federated learning, distributed learning, tracking performance, dynamic optimization, asynchronous SGD, non-IID data, heterogeneous agents

I. INTRODUCTION

We consider a collection of $K$ agents dispersed in space. Each agent $k$ has access to $N_k$ data points denoted by $\{x_{k,n}\}$, where the subscript $k$ denotes the agent index and the subscript $n$ denotes the sample index within the dataset. The objective is to seek the minimizer of the aggregate risk:

$$w^* \triangleq \arg \min_{w \in \mathbb{R}^M} \frac{1}{K} \sum_{k=1}^{K} P_k(w), \quad (1)$$

where the individual risks at the local agents are in turn defined as sample averages over their loss values, i.e.,

$$P_k(w) \triangleq \frac{1}{N_k} \sum_{n=1}^{N_k} Q_k(w; x_{k,n}). \quad (2)$$

We will allow the sought-after model $w^*$ to drift with time and often write $w_{it}$ instead of just $w^*$ to highlight this possibility. Here, the subscript $i$ refers to a time index. The drift in $w^*$ is often the result of variations in the statistical properties of the data $\{x_{k,n}\}$ at the agents, which can change with time as well.

Strategies for the pursuit of solutions to (1) can generally be divided into one of two classes: distributed architectures with a fusion center that collects all data centrally for processing [1]–[3], and fully decentralized strategies that rely on local information exchanges among neighbouring agents over a graph [4]–[7]. Federated learning [8]–[20] offers a midterm solution where data is collected locally at the agents and some processing is also performed locally, while global information is shared between a central processor and the dispersed agents. The architecture helps reduce the amount of communication rounds between the central processor and the agents.

A. Related Works

Several recent works have examined the convergence behaviour of federated learning. They, however, vary in the assumptions on the number of participating agents (all or partial), nature of data (IID or not), operation (synchronous or asynchronous), nature of risk function (convex or non-convex), and bounded conditions on gradient noise [9], [10], [21]–[31] – see Table 1.

Some other works examine the convergence behaviour of a modified version of the original FedAvg algorithm; in [9] FedAvg is modified to include non-uniform epoch sizes among the agents; the authors then provide a convergence proof of the new algorithm, called FedProx, independent of the local solvers. They assume non-convex cost functions with non-IID data, and they quantify the statistical heterogeneity by a dissimilarity measure based on the randomized Kaczmarz method [32], [33] for solving linear system of equations. However, this proof fails to encapsulate the convergence of FedAvg. In [31], a hierarchical version of FedAvg is developed where model aggregation occurs at several levels, and the convergence under non-IID settings for both convex and non-convex functions is studied. The authors of [10] introduce multi-task federated learning to deal with statistical heterogeneity and show their algorithm converges for convex cost functions. They introduce the MOCHA algorithm that extends CoCoA [34] to deal with the challenges introduced by the federated setting, such as stragglers.

While related works have considered dynamic optimization algorithms under varying algorithmic frameworks [35]–[42], we focus in this work on the original FedAvg algorithm with varying step-sizes and under a non-stationary environment.

II. ALGORITHM DERIVATION

Returning to (1), in the absence of communication or computational constraints, the centralized gradient descent step takes the form:

$$w_i = w_{i-1} - \mu \frac{1}{K} \sum_{k=1}^{K} \nabla w_{i} P_k(w_{i-1}), \quad (3)$$

where $i$ is the iteration index. We can distribute the update by splitting the gradient descent step among the $K$ agents. After
TABLE I: List of references on the convergence analysis of federated learning under different assumptions. This work is the only one to tackle the 3 challenges of federated learning.

| References | Algorithm          | Function Type | Data Heterogeneity | Operation | Agent Participation | Other Assumptions |
|------------|--------------------|---------------|--------------------|-----------|---------------------|-------------------|
| [22]       | dist. gradient descent | convex         | non-IID            | synchronous | full                | smooth            |
| [23]       | dist. SGD           | convex         | IID                | synchronous | full                | smooth            |
| [24], [25] | dist. SGD           | non-convex     | IID                | synchronous | full                | smooth            |
| [26]       | dist. SGD           | non-convex     | non-IID            | asynchronous | full                | bounded gradients |
| [27]       | dist. SGD           | convex         | non-IID            | synchronous | full                | -                 |
| [28]       | dist. momentum SGD  | non-convex     | non-IID            | synchronous | full                | bounded gradients |
| [29]       | FedAvg             | convex         | non-IID            | asynchronous | partial            | bounded gradients |
| [30]       | FedAvg             | convex         | non-IID            | asynchronous | partial            | model drift       |

This work

Table and text: introducing local parameters $w_{k,i}$ at each agent, recursion (3) becomes:

$$w_{k,i} = w_{i-1} - \mu \nabla w^T P_k(w_{i-1}),$$

$$w_i = \frac{1}{K} \sum_{k=1}^{K} w_{k,i-1}. \quad (4)$$

The drawback of this formulation is that evaluation of $\nabla w^T P_k(w_{i-1})$ at each agent $k$ and for every iteration $i$ requires a total of $N_k$ evaluations of the gradients $\nabla Q_k(w_{i-1}; x_{k,n})$. A popular approach for reducing the per-iteration computational cost is the utilization of stochastic gradient approximations. We will construct the gradient approximation as the average cost is the utilization of stochastic gradient approximations.

We will construct the gradient approximation as the average $\nabla w^T P_k(w_{i-1})$ of a federated learning scenario [8], we allow for a participation of only $L$ agents at every iteration, and sample the set of indices of participating agents $L_i$ from $\{1, \ldots, L\}$ without replacement, transforming the combination step (10) to:

$$w_i = \frac{1}{L} \sum_{\ell \in L_i} w_{\ell,E_i}. \quad (12)$$

With the local incremental update step (11) and partial-participation combination step (12) we arrive at Algorithm 1.

This algorithm bears similarity to the original FedAvg [8] algorithm, but differs in the fact that we allow for varying local epoch sizes $E_k$ and the normalization of gradient directions by epoch size. The advantage of this construction is in its applicability to settings where agents have varying capabilities so that some agents are able to run multiple epochs while others are not. Instead of forcing capable agents to only run a single epoch, not taking advantage of their computational resources, or forcing slow agents to perform multiple epochs, risking a straggler effect, this model allows every agent to contribute precisely as much as they are able to. The normalization of gradients by the number of local epochs $E_k$, as our analysis will show, is necessary to ensure an unbiased solution despite asymmetric agent contribution, essentially by ensuring that (6) is an unbiased estimate of the gradient of $P_k(\cdot)$. In the absence of step-size normalization, agents with...
more participation would be able to bias the limiting point of the algorithm towards their own local minimizers.

**Algorithm 1** (Dynamic Federated Averaging)

initialize $w_0$

for each iteration $i = 1, 2, \cdots$ do

Select set of participating agents $\mathcal{L}_i$ by sampling $L$ times from $\{1, \ldots, K\}$ without replacement.

for each agent $k \in \mathcal{L}_i$ do

initialize $w_{k,-1} = w_{i-1}$

for each epoch $e = 1, 2, \cdots, E_k$ do

Find indices of the mini-batch sample $\mathcal{B}_{k,e}$ by sampling $B_k$ times from $\{1, \ldots, N_k\}$ without replacement.

$g = \frac{1}{B_k} \sum_{b \in \mathcal{B}_{k,e}} \nabla w_i^T Q(w_{k,e-1}; x_{k,b})$

$w_{k,e} = w_{k,e-1} - \mu \frac{1}{B_k} g$

end for

end for

$w_i = \frac{1}{L} \sum_{k \in \mathcal{L}_i} w_{k,E_k}$

end for

III. CONVERGENCE ANALYSIS

A. Modeling Conditions

To facilitate the performance analysis of the dynamic federated averaging algorithm in a non-stationary environment, we assume convexity and smoothness of gradients.

**Assumption 1.** The functions $P_k(\cdot)$ are $\nu-$strongly convex, and $Q_k(\cdot; x_{k,n})$ are convex:

\[
P_k(w_2) \geq P_k(w_1) + \nabla w_i^T P_k(w_1)(w_2 - w_1) + \frac{\nu}{2} \|w_2 - w_1\|^2, \tag{13}\]

\[
Q_k(w_2) \geq Q_k(w_1) + \nabla w_i^T Q_k(w_1)(w_2 - w_1). \tag{14}\]

They also have $\delta-$Lipschitz gradients:

\[
\|\nabla w_i^T P_k(w_2) - \nabla w_i^T P_k(w_1)\| \leq \delta \|w_2 - w_1\|, \tag{15}\]

\[
\|\nabla w_i^T Q_k(w_2; x) - \nabla w_i^T Q_k(w_1; x)\| \leq \delta \|w_2 - w_1\|. \tag{16}\]

We also impose an assumption on the drift of the minimizer $w^o_i$, namely that it follows a random walk model.

**Assumption 2.** We assume that the true model $w^o_i$ follows a random walk:

\[
w^o_i = w^o_{i-1} + q_i, \tag{17}\]

where $q_i$ denotes some zero mean random variable independent of $w^o_j$ for any $j < i$ and with bounded variance, i.e., $\mathbb{E}\|q_i\|^2 = \sigma_q^2$.

As it will turn out, the tracking performance of Algorithm 1 will be largely determined by the drift parameter $\sigma_q^2$ on the global model $w^o_i$. Nevertheless, we need to additionally assume that the individual minimizers $w^o_{k,i} \triangleq \arg\min_{w \in \mathbb{R}^M} P_k(w)$ do not drift too far.

**Assumption 3.** For all $i$, the distance of each local model $w^o_{k,i}$ to the global model $w^o_i$ is bounded, i.e.:

\[
\mathbb{E}\|w^o_{k,i} - w^o_i\|^2 \leq \sigma_{w,k}^2. \tag{18}\]

B. Error Recursion

Iterating the local update steps (11) and combining via (12), we find:

\[
w_i = w_{i-1} - \mu L \sum_{\ell \in \mathcal{L}_i} \frac{1}{E_{\ell}} \sum_{b \in \mathcal{B}_{\ell,b}} \nabla w_i^T P_k(w_{i-1}) - \mu s_i - \mu d_i, \tag{19}\]

where we introduced:

\[
s_i \triangleq \frac{1}{L} \sum_{\ell \in \mathcal{L}_i} \sum_{b \in \mathcal{B}_{\ell,b}} \nabla w_i^T P_k(w_{i-1}) - \frac{1}{K} \sum_{k=1}^K \nabla w_i^T P_k(w_{i-1}), \tag{20}\]

\[
d_i \triangleq \frac{1}{L} \sum_{\ell \in \mathcal{L}_i} \sum_{b \in \mathcal{B}_{\ell,b}} \left[ \nabla w_i^T P_k(w_{i-1}; x_{\ell,b}) - \nabla w_i^T P_k(w_{\ell,\epsilon}; x_{\ell,b}) \right]. \tag{21}\]

The term $s_i$ results from the stochastic approximation of the true gradient of (1) by only utilizing a subset of agents and a subset of data at every iteration. The term $d_i$ results from the incremental implementation of (11). The gradient noise term $s_i$ will turn out to be the dominant factor in the performance of the algorithm, but can be bounded as follows.

**Lemma 1.** The gradient noise defined in (20) satisfies:

\[
\mathbb{E}\{s_i\|w_{i-1}\| = 0, \tag{22}\]

\[
\mathbb{E}\{\|s_i\|^2\|w_{i-1}\| \leq \beta_s^2 \mathbb{E}\|w_{i-1}^o - w_{i-1}\|^2 + \sigma_s^2 + \epsilon^2, \tag{23}\]

where we defined:

\[
\beta_s^2 \triangleq \frac{1}{KL} \sum_{k=1}^K \left( 6r_{s,k} + 2r_e \right) \delta^2, \tag{24}\]

\[
\sigma_s^2 \triangleq \frac{3}{KL} \sum_{k=1}^K r_{s,k} \mathbb{E}\|\nabla w_i^T P_k(w_{i-1}; x_k) \|^2 - \nabla w_i^T P_k(w_{i-1}) \|^2, \tag{25}\]

\[
\epsilon^2 \triangleq \frac{2}{KL} \sum_{k=1}^K r_{s,k} \mathbb{E}\|\nabla w_i^T P_k(w_{i-1}) \|^2, \tag{26}\]

\[
r_{s,k} \triangleq \frac{N_k - B_k}{(N_k - 1) B_k E_k}, \quad r_e \triangleq \frac{K - L}{K - 1}. \tag{27}\]

**Proof.** Omitted due to space limitations.
From (23), we observe that the bound on the gradient noise variance consists of two absolute components, $\sigma_{\epsilon}^2$ and $\epsilon^2$. The term $\sigma_{\epsilon}^2$ corresponds to an average of the data variability $\mathbb{E}\|\nabla_{w^i} P_k(w_{i-1}; x_k) - \nabla_{w^i} P_k(w_{i-1})\|^2$ at each agent, weighted by the factor $\tau_{s,k}$. Agents with high data variability will incur a higher variance by employing a mini-batch approximation, instead of a full gradient update, but can mitigate this effect by increasing the mini-batch size $B_k$, and hence reducing $\tau_{s,k}$. In the limit case where agent $k$ is performing a full gradient update, we have $B_k = N_k$, and hence $\tau_{s,k} = 0$ and no noise contribution from agent $k$. The second absolute noise term $\epsilon^2$ stems from model variability among different agents. In particular, $\mathbb{E}\|\nabla_{w} P_k(w_{i-1})\|^2$ measures the suboptimality of the average model $w_{s,k-1}^i$ for the local cost $P_k(\cdot)$ at agent $k$, and in light of the Lipschitz gradient condition is proportional to $\mathbb{E}\|w_{i-1}^o - w_{i-1,k}^o\|^2$; it stems from the incremental step introduced in (11). In this case, the model variability term is multiplied by a participation factor $\tau_s$, which quantifies the fraction of agents participating in the federated update at every iteration, and vanishes whenever all agents participate.

**Theorem 1.** Consider the iterates $w_i$ generated by the dynamic federated averaging algorithm. For sufficiently small step-size $\mu$, it holds that $\mathbb{E}\|w_i^o - w_i\|^2$ converges exponentially fast:

$$\mathbb{E}\|w_i^o - w_i\|^2 \leq O(\gamma^i) + O(\mu) (\sigma_{\epsilon}^2 + \epsilon^2) + O(\mu^{-1})\sigma_{\epsilon}^2,$$

where $\gamma < 1 - \nu \mu + O(\mu^2) \in [0, 1).

**Proof.** Omitted due to space limitations.

We observe that, in addition to the model and data variability terms discussed above, the performance of the algorithm is further determined by the drift parameters of the random-walk model $\sigma_{\gamma}^2$. A reduction in the step-size results in slower decay of the transient term $O(\gamma^i)$ and increased tracking loss $O(\mu^{-1})\sigma_{\epsilon}^2$, while reducing the effect of the gradient noise proportional to $O(\mu)$.

**IV. EXPERIMENTAL ANALYSIS**

**A. Experimental Setup**

We test the dynamic federated averaging algorithm on the logistic risk function with $\ell_2$-norm regularization (29) using synthetic data. The local risk is given by:

$$P_k(w) = \frac{1}{N_k}\sum_{i=0}^{N_k-1} \ln(1 + e^{-\gamma h^T w}) + \rho\|w\|^2.$$  

The data is generated as follows: We first generate a random $w_i^o$ and apply the random walk model (17) with $q_i \sim N(0, \sigma_{\gamma}^2)$. Then, we model the change in the true parameters $w_i^o$ across agents by adding randomly sampled constants $c_k \sim N(0, \sigma_{w}^2)$, i.e., $w_{i,k}^o = w_i^o + c_k$. Then, for each agent $k$, $N_k$ random 2-dimensional features are sampled from a Gaussian distribution with a randomly generated variance, for each time $i$. We generate random feature vectors $h_{k,i}$ and assign them labels $\gamma_{k,i} = \text{sign}(h_{k,i}^T w_{i,k}^o)$. We assume we have $K = 20$ agents, with $L = 7$ active agents at each time. We set the batch sizes $B_k$ and epoch sizes $E_k$ to different values in the range [10, 20] and [1, 10], respectively.

We examine the effect of different hyperparameters on the behaviour of the algorithm. We validate the theoretical results by changing the step size; we also look at the effect of varying the variances $\sigma_{\epsilon}^2$ and $\sigma_{\gamma}^2$. We plot the averaged mean-square-deviation (MSD) curves in log domain after performing 50 passes over the data (Figure 1). For the first experiment, we set $\sigma_{\epsilon}^2 = 0.1$, and we vary the step size by factor of 10 in the stationary ($\sigma_{w}^2 = 0$) and non-stationary case ($\sigma_{w}^2 = 0.01$). We observe in Figure 1a that as $\mu$ increases, the MSD increases and the convergence becomes faster; while, in Figure 1b we plot the final MSD value reached for each $\mu$, and we observe that there is an optimal step size $\mu^* = 0.1$ that achieves the best MSD. For the second experiment, we vary $\sigma_{\gamma}^2$ while fixing $\sigma_{\epsilon}^2 = 0$ and $\mu = 0.01$. The plots in Figure 1c indicate that the MSD increases with $\sigma_{\gamma}^2$, but the convergence rate is not affected, which is expected. Finally, in the third experiment we fix $\sigma_{\epsilon}^2 = 0.1$ and vary $\sigma_{\gamma}^2$. A similar trend is observed (Figure 1d); $\sigma_{\gamma}^2$ only affects the MSD and not the convergence.

**V. CONCLUSION**

The work presented in this paper consists of developing a convergence analysis for a modified version of the FedAvg algorithm under three major challenges: non-IID data, asynchronous operation under drift, and partial agent participation. We were able to guarantee the convergence of the algorithm and identified three major components that affect its convergence: step size, agent heterogeneity, and drift variance. We were able to illustrate the theoretical results with a series of experiments.
