IEEE Control Systems Letters paper presented at
2023 American Control Conference (ACC)
San Diego, CA, USA. May 31 - June 2, 2023

Federated Learning Using Variance Reduced Stochastic Gradient for Probabilistically Activated Agents

Mohammadreza Rostami and Solmaz S. Kia, *Senior Member, IEEE*

***Abstract*— This paper proposes an algorithm for Federated Learning (FL) with a two-layer structure that achieves both variance reduction and a faster convergence rate to an optimal solution in the setting where each agent has an arbitrary probability of selection in each iteration. The first layer of our algorithm corresponds to the model parameter propagation across agents done by the server. In the second layer, each agent does its local update with a stochastic and variance-reduced technique called Stochastic Variance Reduced Gradient (SVRG). We leverage the concept of variance reduction from stochastic optimization when the agents want to do their local update step to reduce the variance caused by stochastic gradient descent (SGD). The special attention in this paper is on FL operation where the agents' participation in the update process in each round is probabilistic and non-uniform. We provide a convergence bound for our algorithm which improves the rate from $O(\frac{1}{\sqrt{K}})$ to $O(\frac{1}{K})$ by using a constant step-size when the cost is strongly convex. For non-convex costs, we establish a $O(\frac{1}{\sqrt[3]{K^2}})$ to a stationary point using a vanishing stepsize. We demonstrate the performance of our algorithm using numerical simulations.**

I. Introduction

In recent years, with the technological advances in modern smart devices, each phone, tablet, or smart home system, generates and stores an abundance of data, which, if harvested collaboratively with other users' data, can lead to learning models that support many intelligent applications such as smart health and image classification [1], [2]. Standard traditional machine learning approaches require centralizing the training data on one machine, cloud, or in a data center. However, the data collected on modern smart devices are often of sensitive nature that discourages users from relying on centralized solutions. Federated Learning (FL) [3], [4] has been proposed to decouple the ability to do machine learning from the need to store the data in a centralized location. The idea of Federated Learning is to enable smart devices to collaboratively learn a shared prediction model while keeping all the training data on the device.

Figure 1 shows a schematic representation of an FL architecture. In FL, collaborative learning without data sharing is accomplished by each agent receiving a current model weight from the server. Then, each participating learning separately updates the model by implementing a stochastic gradient descent (SGD) [5] using its own locally collected datasets. Then, the participating agents send their locally calculated model weights to a server/aggregator, which often combines the models through a simple averaging, as in FedAvg [4], to be sent back to the agents. The process repeats until a satisfactory model is obtained. Federated learning relies heavily on communication between learner agents (clients) and a moderating server. Engaging all the clients in the learning procedure at each time step of the algorithm results in huge communication cost. On the other hand, poor channel quality and intermittent connectivity can completely derail training. For resource management, in the original popular FL algorithms such as FedAvg in [4], at each round of the algorithm, a batch of agents are selected uniformly at random to receive the updated model weights and perform local learning. FedAvg and similar FL algorithms come with convergence guarantees [6]–[9] under the assumption of availability of the randomly selected agents at each round. However, in practice due to factors such as energy and time constraints, agents' availability is not ubiquitous at all times. Thus, some works have been done to solve this problem via device scheduling [10]–[14]. Nevertheless, the agents' availability can be a function of unforeseen factors such as communication channel quality, and thus is not deterministic and known in advance.

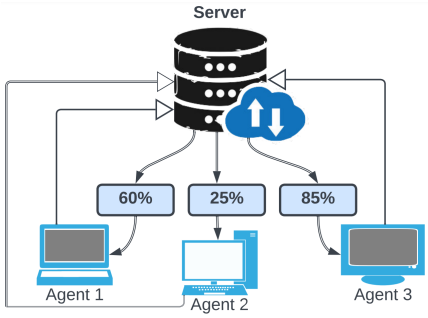

Fig. 1: Federated Learning structure with non-uniform probability of agent selection in each iteration.

To understand the effect of an agent's stochastic availability on the FL, recent work such as [15] proposed to move from random batch selection to an FL model where the agents availability and participation at each round are probabilistic, see Fig. 1. In this paper, we adopt this newly proposed framework and contribute to devising an algorithm that achieve faster convergence and lower error covariance. Our focus will be on incorporating a variance reduction procedure into the local SGD procedure of participating learner agents at each round. The randomness in SGD algorithms induces variance of the gradient, which leads to decay learning rate and sub-linear convergence rate. Thus, there has been an effort to reduce the variance of the stochastic gradient, which

The authors are with the Department of Mechanical and Aerospace Engineering, University of California Irvine, Irvine, CA 92697, {mrostam2,solmaz}@uci.edu. This work was supported by NSF, under CAREER Award ECCS-1653838.

resulted in the so-called *Stochastic Variance Reduced Gradient* (SVRG) methods. It is shown that SVRG allows using a constant learning rate and results in linear convergence in expectation.

In this paper, we incorporate an SVRG approach in an FL algorithm where agents' participation in the update process in each round is probabilistic and non-uniform. Through rigorous analysis, we show that the proposed algorithm has a faster convergence rate. In particular, we show that our algorithm results in a practical convergence in expectation with a rate $O(\frac{1}{K})$, which is an improvement over the sublinear rate of $O(\frac{1}{\sqrt{K}})$ in [15]. We demonstrate the effectiveness of our proposed algorithm through a set of numerical studies and by comparing the rate of convergence, covariance, and the circular error probable (CEP) measure. Our results show that our algorithm drastically improves the convergence guarantees, thanks to the decrease in the variance, which results in faster convergence.

Organization: Section II introduces our basic notation, and presents some of the properties of smooth functions. Section III presents the problem formulation and the structure behind it. Section IV includes the proposed algorithm and its scheme. Section V contains our convergence analysis for the proposed algorithm and provides its convergence rate. Section VI presents simulations and Section VII gathers our conclusions and ideas for future work. For the convenience of the reader, we provide some of the proofs in the Appendix.

II. Preliminaries

In this section, we introduce our notations and terminologies used throughout the paper. We let $\mathbb{R}$, $\mathbb{R}_{>0}$, $\mathbb{R}_{\geq 0}$, denote the set of real, positive real numbers. Consequently, when $x \in \mathbb{R}$, $|x|$ is its absolute value. For $\mathbf{x} \in \mathbb{R}^d$, $\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}}$ denotes the standard Euclidean norm. We let $\langle ., . \rangle$ denotes an inner product between two vectors for two vectors x and $y \in \mathbb{R}^d$. A differentiable function $f\colon \mathbb{R}^d \to \mathbb{R}$ is Lipschitz with constant $\mathsf{L} \in \mathbb{R}_{>0}$, or simply L-Lipschitz, over a set $\mathcal{C} \subseteq \mathbb{R}^d$ if and only if $\|f(x) - f(y)\| \leq \mathsf{L}\|x - y\|$, for $x, y \in \mathcal{C}$. Furthermore, if the function is differentiable, we have $f(y) \leq f(x) + \nabla f^\top (y - x) + \frac{\mathsf{L}}{2}\|y - x\|^2$ for all $x, y \in \mathcal{C}$ [16]. Lastly, we recall Jensen's inequality, which states [17]:

$$\left\| \frac{1}{N} \sum\nolimits_{n=1}^{N} x_n \right\|^2 \leq \frac{1}{N} \sum\nolimits_{n=1}^{N} \|x_n\|^2. \tag{1}$$

III. Problem Statement

This section formalizes the problem of interest. Consider a set of N agents (clients) that communicate with a server to learn parameters of a model that they want to fit into their collective data set. Each agent has its own local data which can be distributed either uniformly or non-uniformly. The learning objective is to obtain the learning function weights $\theta \in \mathbb{R}^d$ from

$$\min_{\theta \in \mathbb{R}^d} f(\theta) := \frac{1}{N} \sum_{n=1}^{N} f_n(\theta), \quad f_n(\theta) = \frac{1}{L_n} \sum_{i=1}^{L_n} f_{n,i}(\theta), \tag{2}$$

where f_n is possibly a convex or non-convex local learning loss function. At each agent $n \in \{1, \cdots, N\}$, f_n depends on

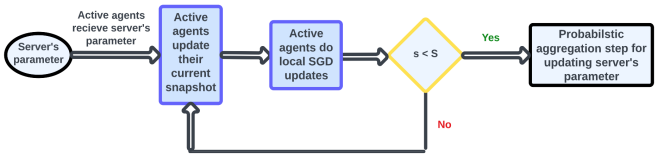

Fig. 2: SVRG update steps.

training data set $\{(q_{n,i}, \hat{y}_{n,i})\}_{i=1}^{L_n} \subset \mathbb{R}^{1\times d} \times \mathbb{R}$ (supervised learning). Examples include [18]

- square loss $f_{n,i}(\theta) = \|(\hat{y}_{n,i} - q_{n,i}\theta)\|^2$,
- log loss $f_{n,i}(\theta) = \log(1 + \mathrm{e}^{-\hat{y}_{n,i} q_{n,i} \theta})$.

Assumption 1 (Assumption on L-smoothness of local cost functions): *The local loss functions have L-Lipschitz gradients, i.e., for any agent $n \in \{1, \cdots, N\}$ we have*

$$\|\nabla f_n(\theta) - \nabla f_n(\bar{\theta})\| \leq \mathsf{L}\|\theta - \bar{\theta}\| \tag{3}$$

for any $\theta, \bar{\theta} \in \mathbb{R}^d$ and $\mathsf{L} \in \mathbb{R}_{>0}$.

This assumption is technical and common in the literature.

Problem (2) should be solved in the framework of FL in which agents maintain their local data and only interact with the server to update their local learning parameter vector based on a global feedback provided by the server. The server generates this global feedback from the local weights it receives from the agents. In our setting, at each round k of the FL algorithm, each agent $n \in \{1, \cdots, N\}$ becomes active to perform local computations and connect to the server with a probability of p_n^k. We denote the active state by $\mathbf{1}_n^k \in \{0, 1\}$; thus,

$$p_n^k = \text{Prob}(\mathbf{1}_n^k = 1).$$

The activation probability at each round can be different.

IV. Federated Learning with Variance Reduction

To solve (2), we design the FedAvg-SVRG Algorithm 1, which has a two-layer structure. In this algorithm, each agent has its own probability to be active or passive in each round which is denoted by p_n^k for agent n at iteration k.

Algorithm 1 is initialized with θ^0 by the server. We denote the number of the FL iterations by K. At each round $k \in \{1, \cdots, K\}$, the set of active agents is denoted by $\mathcal{A}^k$ (line 5), which is the set of agents for which $\mathbf{1}_n^k = 1$. Then, each active agent receives a copy of the learning parameter θ^k from the server. Afterward, active agents perform their local updates according to steps 7 to 18. For resource management local update in FL algorithms, e.g., [15], follow an SGD update. However, the SGD update suffers from a high variance because of the randomized search of the algorithm, so instead of using the SGD update step, which results in a decaying step size and slow convergence, we use the SVRG update step which is stated in lines 7 to 18. In the SVRG update, we calculate the full batch gradient of the agents at some points, which are referred to as *snapshots*. Then, between every two snapshots, each agent does its local update. A schematic of SVRG update steps is shown in Fig. 2.

Algorithm 1 FedAvg-SVRG with non-uniformly agent sampling

1: **Input:** δ, K, θ^0, $\{p_n^k\}$, S, M
2: **Output:** θ^k
3: **for** $k \leftarrow 0, ..., K-1$ **do**
4: Determine the active agents (sample $\mathbf{1}_n^k \sim p_n^k$)
5: $\mathcal{A}^k \leftarrow$ set of active agents
6: **for** $n \in \mathcal{A}^k$ **do**
7: $\tilde{w}_{n,0}^k = \theta^k$
8: **for** $s \leftarrow 0, ..., S-1$ **do**
9: $\tilde{w} = \tilde{w}_{n,s}^k$
10: $\tilde{\mu} = \frac{1}{L_n} \sum_{i=1}^{L_n} \nabla f_{n,i}(\tilde{w})$
11: $w_{n,s,0}^k = \tilde{w}$
12: **for** $m \leftarrow 0, ..., M-1$ **do**
13: Randomly pick # from $\in \{1, ..., L_n\}$
14: $w_{n,s,m+1}^k = w_{n,s,m}^k - \delta v_{n,s,m}^k$
15: where $v_{n,s,m}^k = \nabla f_{\#}(w_{n,s,m}^k) - \nabla f_{\#}(\tilde{w}) + \tilde{\mu}$
16: **end for**
17: set $\tilde{w}_{n,s+1}^k = w_{n,s,M}^k$
18: **end for**
19: **end for**
20: $\theta^{k+1} = \theta^k + \frac{1}{N} \sum_{n=1}^{N} \frac{\mathbf{1}_n^k}{p_n^k} (w_{n,S-1,M}^k - \tilde{w}_{n,0}^k)$
21: **end for**

We denote the number of snapshots in our SVRG method by S. We let M be the number of local SVRG updates in between two snapshots for each active agent before aggregation. Line 10 of Algorithm 1 corresponds to computing the full batch gradient of each agent at the snapshot points, then in line 12, each agent does its local update with substituted gradient term denoted as $v_{n,s,m}^k = \nabla f_{\#}(w_{n,s,m}^k) - \nabla f_{\#}(\tilde{w}) + \tilde{\mu}$. Note the gradient substituted term in the SVRG update is an unbiased estimator. After completing the SVRG update, each agent updates its snapshot, which is mentioned in line 17 [19] [5]. In the end, in line 20, the model parameter gets updated. It should be noted that the weight for updating the model parameter denoted by $\frac{\mathbf{1}_n^k}{p_n^k}$ makes the gradient to be unbiased when the model parameter wants to be updated because, by this fraction, agents with a low probability of being selected for each iteration still have an adequate impact on a model parameter when they play a part at each iteration. Unlike SGD, the stepsize δ for the SVRG update does not have to decay in line 14. Hence, it gives rise to a faster convergence as one can choose a large stepsize.

V. Convergence analysis

In this section, we study the convergence bound for the proposed algorithm which is applicable for both convex and non-convex cost functions.

Assumption 2 (Assumption on unbiased stochastic gradients):

$$\mathbb{E}[\nabla f_{\#}(w)|w] = \nabla f_n(w) \tag{4}$$

for any w *and* $\# \in \{1, ..., L_n\}$. *As a result, our substituted gradient term denoted by* $v_{n,s,m}^k = \nabla f_{\#}(w_{n,s,m}^k) - \nabla f_{\#}(\tilde{w}) + \tilde{\mu}$ *becomes unbiased where* $\tilde{\mu} = \frac{1}{L_n} \sum_{i=1}^{L_n} \nabla f_{n,i}(\tilde{w})$.

Assumption 3 (Bound on the substituted gradient term):

$$\mathbb{E}[||v_{n,s,m}^k(w)||^2] \leq G^2, \forall\, w, n, s, m \quad \textit{for some } G > 0. \tag{5}$$

Assumption 4 (Assumption on μ-strongly convex local cost functions): *The local cost functions are strongly convex with parameter* μ, *i.e.,*

$$f_n(\theta_2) \geq f_n(\theta_1) + \nabla f_n(\theta_1)^T(\theta_2 - \theta_1) + \frac{\mu}{2}||\theta_2 - \theta_1||^2. \tag{6}$$

Also, we should point out that $\mathbf{1}_n^k$ and $\mathbf{1}_{n'}^k$ are independent for $n \neq n'$, and the agent activation for each iteration is independent of random function selection. In other words, $\mathbf{1}_n^k$ and $\nabla f_{\#}(w)$ are completely independent.

Theorem 5.1 (Convergence bound for the proposed algorithm for both convex and non-convex cost functions): *Let Assumptions 1 and 2 hold. Then Algorithm 1 results in*

$$\begin{aligned}
&\frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right] \leq \frac{2}{\delta KSM}(f(\theta^0) - f^\star) \\
&+ \frac{\delta^2 \mathsf{L}^2 (M-1)}{KSMN} \\
&\left[\sum_{k=0}^{K-1} \sum_{n=1}^{N} \sum_{s=0}^{S-1} \sum_{m=0}^{M-1} \sum_{m'=0}^{m-1} \mathbb{E}\left[\left\|v_{n,s,m'}^k\right\|^2\right]\right] \\
&+ \frac{\delta \mathsf{L}}{KN} \sum_{k=0}^{K-1} \sum_{n=1}^{N} \frac{1}{p_n^k} \sum_{s=0}^{S-1} \sum_{m=0}^{M-1} \mathbb{E}\left[\left\|v_{n,s,m}^k\right\|^2\right].
\end{aligned} \tag{7}$$

Furthermore, if assumption (5) holds, then Algorithm 1 results in

$$\begin{aligned}
&\frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right] \leq \frac{2}{\delta KSM}(f(\theta^0) - f^\star) \\
&+ \delta^2 \mathsf{L}^2 (M-1)^2 G^2 \\
&+ \frac{\delta \mathsf{L} SMG^2}{KN} \sum_{k=0}^{K-1} \sum_{n=1}^{N} \frac{1}{p_n^k}.
\end{aligned} \tag{8}$$

where $f^\star$ *is the optimal solution to* (2).

Due to space limitation the proof of Theorem 5.1 appears elsewhere. The Auxiliary lemmas to establish the proof are given in the appendix.

Remark: According to (8) a rate of convergence of the algorithm is determined by $\min\{\frac{1}{\delta KSM}, \delta^2, \frac{\delta}{KSM}\}$. In order to select the convergence rate of our algorithm we can derive it by choosing $\delta = \frac{1}{K^\epsilon}$. Then, the rate of convergence is chosen from $\min\{\frac{1}{K^{1-\epsilon}}, \frac{1}{K^{2\epsilon}}, \frac{1}{K^{1+\epsilon}}\}$. By selecting $\epsilon = \frac{1}{3} = \max\{1-\epsilon, 2\epsilon\}$ the best convergence rate can be obtained, which is of order $O(\frac{1}{\sqrt[3]{K^2}})$. Thus, using the decaying stepsize ($\delta = \frac{1}{\sqrt[3]{K}}$) allows us obtain $O(\frac{1}{\sqrt[3]{K^2}})$ convergence to the optimal point for both convex and non-convex cost functions.

Proposition 5.1: If in additional to the assumptions made in Theorem 5.1, Assumption (4) holds, then Algorithm 1 satisfies

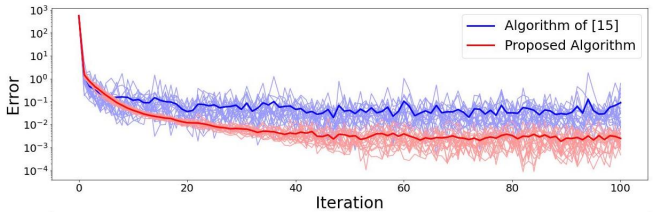

Fig. 3: Error of cost function for the first case ($S = 5$ and $M = 2$) over 20 Monte Carlo iterations (thicker line correspond to the mean of Monte Carlo iterations, vertical axis is limited for the purpose of better visualization).

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right] \leq \frac{2}{\delta KM}(f(\theta^0)-f^\star) + \frac{8D\delta^2\mathsf{L}^3(M-1)^2(1-\alpha^K)}{K(1-\alpha)} + \frac{8D\delta\mathsf{L}^2(M-1)}{KN}\sum_{k=0}^{K-1}\alpha^k\sum_{n=1}^{N}\frac{1}{p_n^k}, \tag{9}$$

where D is a constant and $\alpha = h(\mu, \mathsf{L}, \delta, M) < 1$ with a proper choice of constant δ.

Due to space limitation the proof of Theorem 5.1 appears in [20]. For details regarding α and D see [20]. By incorporating a SVRG approach in our FL algorithm, Proposition 5.1 for strongly convex costs guarantees that we can use a fixed size step-size δ and achieve a convergence rate of $O(\frac{1}{K})$. The improvement is due to the fact that the SVRG update step does not need to have a decaying stepsize throughout the learning process. Thus, using a constant and larger stepsize leads the algorithm to faster convergence. This is an improvement over the existing algorithm [15] in which they guarantee $O(\frac{1}{\sqrt{K}})$ as the convergence rate of the algorithm by using the SGD method for their local update step.

VI. Numerical Simulations

In this section, we analyze and demonstrate the performance of the Algorithm 1 by solving a regression problem (quadratic loss function). In this study, we compare the performance of our algorithm to that of the FedAvg in [15]. We used a real medical insurance data set of 900 persons in the form of $(y, v) \in \mathbb{R} \times \mathbb{R}^{1\times 5}$. Then, to observe the effect of stochastic optimization, we distribute the data among 18 agents. Thus, each agent owns 50 quadratic costs. In other words, we seek to solve the following convex optimization problem:

$$\min_{\theta\in\mathbb{R}^{10}} f(\theta) = \frac{1}{N}\sum_{n=1}^{N} f_n(\theta), \tag{10}$$
$$f_n(\theta) = \frac{1}{L_n}\sum_{i=1}^{L_n} f_{n,i}(\theta), \quad f_{n,i}(\theta) = \|q_{n,i}\theta - \hat{y}_{n,i}\|^2,$$

where in our problem, N, and L_n are 18 and 50, respectively. Here, $\hat{y}_{n,i} \in \mathbb{R}$, and θ is the learning parameter (weight) which is a column vector with 5 elements.We conduct 20 Monte Carlo simulation in all of which we initialize our algorithm at $\theta^0 = [0.5, ..., 0.5]^\top$, and we use the fixed step-size $\delta = \frac{1}{\sqrt{100}}$ in all rounds. We also simulate the FedAvg algorithm of [15] with the same initialization but using the decaying stepsize of $\frac{1}{\sqrt{K}}$ as mentioned in [15]. For our algorithm we consider two cases: (1) $(S = 5, M = 2)$ and (2) $(S = 10, M = 5)$.

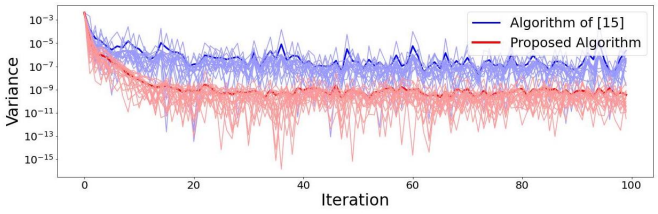

Fig. 4: Variance of cost function for the first case ($S = 5$ and $M = 2$) over 20 Monte Carlo iterations (thicker line correspond to the mean of Monte Carlo iterations, vertical axis is limited for the purpose of better visualization).

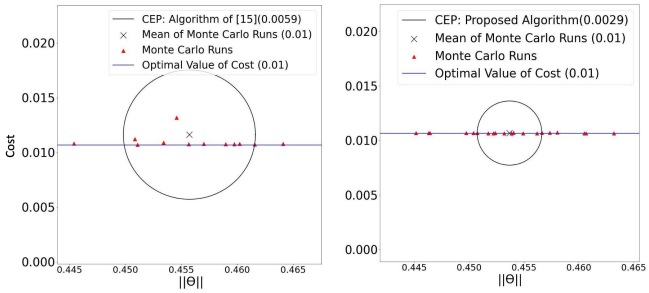

Fig. 5: CEP graph for the second case ($S = 5$ and $M = 2$) over 20 Monte Carlo iterations. The results for our algorithm is in the right while the result for algorithm of [15] is shown on the left.

The simulation results for the first case are shown in Fig. 3–Fig.5, while the results for the second case are shown in Fig. 6–Fig.8. Figures 3 and 6 show that in both cases our algorithm has a faster convergence to the optimal cost (the value is 0.01).

Figures 4 and 7 show the variance caused by the two algorithms. The variance of our algorithm is significantly lower than that of the algorithm of [15]. In order to show the variance of our algorithm, we put a logarithmic axis on y axis. Also, the variance of our algorithm decreases as the number of iterations increases as opposed to the algorithm of [15], which suffers from a high variance.

Figure 5 and 8 show the circular error probable (CEP) to observe the variance in the last iteration ($K = 100$) for our algorithm and the FedAvg algorithm in [15]. CEP is a measure used in navigation filters. It is defined as the radius of a circle, centered on the mean, whose perimeter is expected to include the landing points of 50% of the rounds; said otherwise, it is the median error radius [21]. Here, then, CEP demonstrates how far the means of the Monte Carlo runs are from 50% of the Monte Carlo iterations for both algorithms. As a result, less radius means less variance from the mean of the Monte Carlo runs. This plot shows not only our algorithm reaches a closer neighborhood to the optimal cost, but also, it has less CEP radius in comparison to that of the algorithm of [15]; this is another indication that our algorithm has less variance compared to the FedAvg algorithm in [15]. For our algorithm, the CEP radius in the

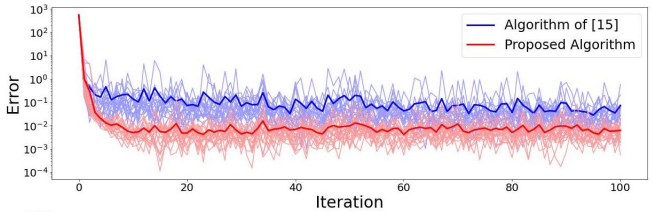

Fig. 6: Error of cost function for the second case ($S = 10$ and $M = 5$) over 20 Monte Carlo iterations (thicker line correspond to the mean of Monte Carlo iterations, y axis is limited for the purpose of better visualization).

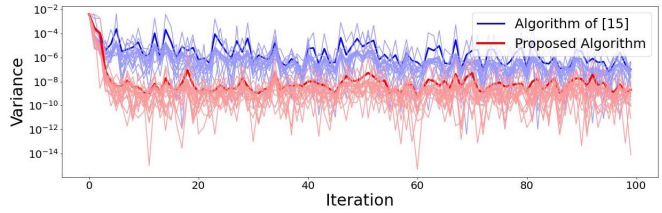

Fig. 7: Variance of cost function for the second case ($S = 10$ and $M = 5$) over 20 Monte Carlo iterations (thicker line correspond to the mean of Monte Carlo iterations, y axis is limited for the purpose of better visualization).

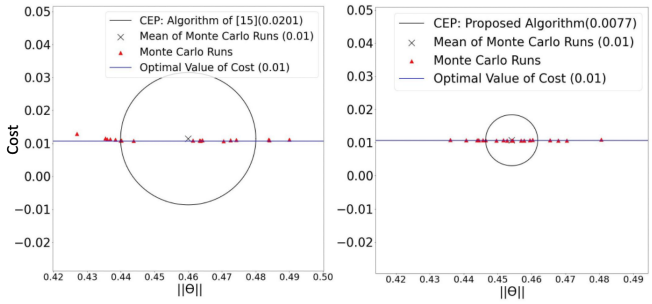

Fig. 8: CEP graph for the second case ($S = 10$ and $M = 5$) over 20 Monte Carlo iterations. The results for our algorithm is in the right while the result for algorithm of [15] is shown on the left.

first and the second cases are respectively 0.0029 and 0.0077, while these values of the algorithm of [15] are respectively 0.0059 and 0.0201.

To complete our simulation study, we also compare the convergence performance of our algorithm to that of the FedAvg of [4], which uses a uniform agent selection. Figure 9 demonstrates the results when we use the batch size of 5 of the FedAvg of [4] and use the parameters corresponding to the first case for our algorithm. As we can see, our algorithm outperforms the FedAvg of [4] both in mean and variance.

VII. Conclusions

We have proposed an algorithm in the FL framework in the setting where each agent can have a non-uniform probability of becoming active (getting selected) in each FL round. The algorithm possesses a doubly-layered structure as the original FL algorithms. The first layer corresponds to distributing the server parameter to the agents. At the second layer, each

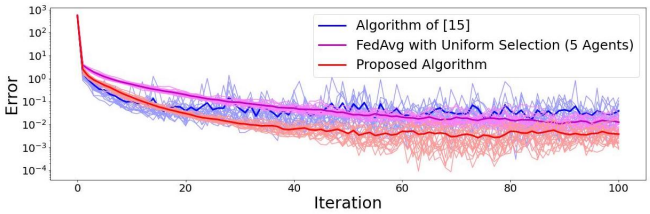

Fig. 9: Comparing the results of the cost function for three algorithms over 20 Monte Carlo iterations (thicker line correspond to the mean of Monte Carlo iterations, y axis is limited for the purpose of better visualization).

agent updates its copy of the server parameter through an SVRG update. Then after each agent sends back its update, the server parameter gets updated. By leveraging the SVRG technique from stochastic optimization, we constructed a local updating rule that allowed the agents to use fixed stepsize. We characterized an upper bound for the gradient of the expected value of the cost function, which showed our algorithm converges to the optimal solution with the rate of no less than $O(\frac{1}{K})$ for strongly convex costs. This showed an improvement over the existing results that only have a convergence rate of $O(\frac{1}{\sqrt{K}})$. We demonstrated the performance of our algorithm through a detailed simulation study. We used various statistical measures to show our algorithm's faster convergence and low variance compared to existing state-of-the-art FL algorithms. Future work will investigate the extension of the result to allow non-uniform selection of snapshots inside the SVRG update for computing the full batch gradient of the agents.

References

[1] D. C. Nguyen, Q. V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. A. Dobre, and W. J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–37, 2022.

[2] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[3] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, 2019.

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics*, (Lauderdale, FL), pp. 1273–1282, 2017.

[5] R. Xin, S. Kar, and U. A. Khan, "Decentralized stochastic optimization and machine learning: A unified variance-reduction framework for robust performance and fast convergence," *tspm*, pp. 102–113, 2020.

[6] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," 2019.

[7] A. Mitra, R. Jaafar, G. J. Pappas, and H. Hassani, "Achieving linear convergence in federated learning under objective and systems heterogeneity," *arXiv preprint arXiv:2102.07053*, 2021.

[8] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, pp. 429–450, 2020.

[9] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal of Selected Areas in Communications*, no. 6, pp. 1205–1221, 2019.

[10] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317–333, 2019.

[11] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[12] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards flexible device participation in federated learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 3403–3411, PMLR, 2021.
[13] H. Yang, M. Fang, and J. Liu, "Achieving linear speedup with partial worker participation in non-iid federated learning," *arXiv preprint arXiv:2101.11203*, 2021.
[14] 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*, vol. 20, pp. 269–283, 2021.
[15] J. Perazzone, S. Wang, M. Ji, and K. Chan, "Communication-efficient device scheduling for federated learning using stochastic optimization," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, (Piscataway, N,J), pp. 1449–1458, IEEE, 2022.
[16] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.
[17] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas (Second Edition)*. Princeton reference, Princeton University Press, 2009.
[18] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st ed., 2017.
[19] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems* (C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds.), vol. 26, (57 Morehouse Lane Red Hook, N,Y), Curran Associates, Inc., 2013.
[20] M. Rostami and S. Kia, "Federated learning using variance reduced stochastic gradient for probabilistically activated agents," *arXiv preprint arXiv:2210.14362*, 2023.
[21] J. C. Spall and J. L. Maryak, "A feasible bayesian estimator of quantiles for projectile accuracy from non-iid data," *Journal of the American Statistical Association*, vol. 87, no. 419, pp. 676–681, 1992.

APPENDIX

This appendix gives the auxiliary lemmas that are used in the main proofs.

Lemma A.1: Algorithm 1 results in

$$\frac{\delta^2 \mathsf{L}}{2N^2}\,\mathbb{E}\left[\left\|\sum_{n=1}^{N}\frac{\mathbf{1}_n^k}{p_n^k}\sum_{s=0}^{S-1}\sum_{m=0}^{M-1}v_{n,s,m}^k\right\|^2\middle|\theta^k\right] \leq NSM\sum_{n=1}^{N}\frac{1}{p_n^k}\sum_{s=0}^{S-1}\sum_{m=0}^{M-1}\mathbb{E}\left[\left\|v_{n,s,m}^k\right\|^2\middle|\theta^k\right] \quad \text{(A.11)}$$

Proof: By using Jensen's inequality we get the following inequalities:

$$\begin{aligned}
&\mathbb{E}\left[\left\|\sum_{n=1}^{N}\frac{\mathbf{1}_n^k}{p_n^k}\sum_{s=0}^{S-1}\sum_{m=0}^{M-1}v_{n,s,m}^k\right\|^2\middle|\theta^k\right]\\
&\leq N\sum_{n=1}^{N}\mathbb{E}\left[\left\|\frac{\mathbf{1}_n^k}{p_n^k}\sum_{s=0}^{S-1}\sum_{m=0}^{M-1}v_{n,s,m}^k\right\|^2\middle|\theta^k\right]\\
&= N\sum_{n=1}^{N}\frac{\mathbb{E}[\mathbf{1}_n^k|\theta^k]}{(p_n^k)^2}\,\mathbb{E}\left[\left\|\sum_{s=0}^{S-1}\sum_{m=0}^{M-1}v_{n,s,m}^k\right\|^2\middle|\theta^k\right]\\
&\leq NS\sum_{n=1}^{N}\frac{\mathbb{E}[\mathbf{1}_n^k|\theta^k]}{(p_n^k)^2}\sum_{s=0}^{S-1}\mathbb{E}\left[\left\|\sum_{m=0}^{M-1}v_{n,s,m}^k\right\|^2\middle|\theta^k\right]\\
&\leq NSM\sum_{n=1}^{N}\frac{\mathbb{E}[\mathbf{1}_n^k|\theta^k]}{(p_n^k)^2}\sum_{s=0}^{S-1}\sum_{m=0}^{M-1}\mathbb{E}\left[\left\|v_{n,s,m}^k\right\|^2\middle|\theta^k\right]\\
&\leq NSM\sum_{n=1}^{N}\frac{1}{p_n^k}\sum_{s=0}^{S-1}\sum_{m=0}^{M-1}\mathbb{E}\left[\left\|v_{n,s,m}^k\right\|^2\middle|\theta^k\right],
\end{aligned}$$

which concludes the proof. ∎

Lemma A.2: Algorithm 1. results in

$$\begin{aligned}
&-\delta\,\mathbb{E}\left[\left\langle\nabla f(\theta^k),\frac{1}{N}\sum\nolimits_{n=1}^{N}\nabla f_n(w_{n,s,m}^k)\right\rangle\right]\\
&\leq\frac{\delta^3\mathsf{L}^2(M-1)}{2N}\sum_{n=1}^{N}\sum_{m'=0}^{m-1}\mathbb{E}\left[\left\|v_{n,s\cdot m'}^k\right\|^2\right]\\
&\quad-\frac{\delta}{2}\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right] \quad \text{(A.12)}
\end{aligned}$$

Proof: We start by noting that

$$\begin{aligned}
&-\delta\,\mathbb{E}\left[\left\langle\nabla f(\theta^k),\frac{1}{N}\sum\nolimits_{n=1}^{N}\nabla f_n(w_{n,s,m}^k)\right\rangle\right]=\\
&-\delta\,\mathbb{E}\left[\left\langle\nabla f(\theta^k),\frac{1}{N}\sum\nolimits_{n=1}^{N}\nabla f_n(w_{n,s,m}^k)-\nabla f(\theta^k)\right.\right.\\
&\left.\left.+\nabla f(\theta^k)\right\rangle\right]\\
&=\delta\,\mathbb{E}\left[\left\langle\nabla f(\theta^k),-\frac{1}{N}\sum_{n=1}^{N}\nabla f_n(w_{n,s,m}^k)+\nabla f(\theta^k)\right\rangle\right]\\
&-\delta\,\mathbb{E}\left[\left\langle\nabla f(\theta^k),\nabla f(\theta^k)\right\rangle\right]\\
&\leq\frac{\delta}{2}\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right]-\delta\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right]\\
&+\frac{\delta}{2}\,\mathbb{E}\left[\left\|\nabla f(\theta^k)-\frac{1}{N}\sum_{n=1}^{N}\nabla f_n(w_{n,s,m}^k)\right\|^2\right]\\
&=\frac{\delta}{2}\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right]-\delta\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right]\\
&+\frac{\delta}{2}\,\mathbb{E}\left[\left\|\frac{1}{N}\sum_{n=1}^{N}\left[\nabla f_n(\theta^k)-\nabla f_n(w_{n,s,m}^k)\right]\right\|^2\right]\\
&\leq\frac{\delta}{2N}\sum_{n=1}^{N}\mathbb{E}\left[\left\|\nabla f_n(\theta^k)-\nabla f_n(w_{n,s,m}^k)\right\|^2\right]\\
&-\frac{\delta}{2}\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right]\\
&\leq\frac{\delta\mathsf{L}^2}{2N}\sum_{n=1}^{N}\mathbb{E}\left[\left\|\theta^k-w_{n,s,m}^k\right\|^2\right]-\frac{\delta}{2}\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right]\\
&=\frac{\delta\mathsf{L}^2}{2N}\sum_{n=1}^{N}\mathbb{E}\left[\left\|\sum_{m'=0}^{m-1}\delta v_{n,s,m'}^k\right\|^2\right]-\frac{\delta}{2}\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right]\\
&\leq\frac{\delta^3\mathsf{L}^2}{2N}\sum\nolimits_{n=1}^{N}\mathbb{E}\left[\left\|\sum_{m'=0}^{m-1}v_{n,s,m'}^k\right\|^2\right]\\
&-\frac{\delta}{2}\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right]\\
&\leq\frac{\delta^3\mathsf{L}^2(M-1)}{2N}\sum\nolimits_{n=1}^{N}\sum_{m'=0}^{m-1}\mathbb{E}\left[\left\|v_{n,s,m'}^k\right\|^2\right]\\
&-\frac{\delta}{2}\,\mathbb{E}\left[\left\|\nabla f(\theta^k)\right\|^2\right],
\end{aligned}$$

which concludes the proof. ∎