Efficient Fully Distributed Federated Learning with Adaptive Local Links

Evangelos Georgatos  
Comp. Eng. and Informatics Dept.  
University of Patras  
Patras, Greece  
egeorgatos@ceid.upatras.gr

Christos Mavrokefalidis  
Comp. Eng. and Informatics Dept.  
University of Patras  
Patras, Greece  
maurokef@ceid.upatras.gr

Kostas Berberidis  
Comp. Eng. and Informatics Dept.  
University of Patras  
Patras, Greece  
berberid@ceid.upatras.gr

Abstract—Nowadays, data-driven, machine and deep learning approaches have provided unprecedented performance in various complex tasks, including image classification and object detection, and in a variety of application areas, like autonomous vehicles, medical imaging and wireless communications. Traditionally, such approaches have been deployed, along with the involved datasets, on standalone devices. Recently, a shift has been observed towards the so-called Edge Machine Learning, in which centralized architectures are adopted that allow multiple devices with local computational and storage resources to collaborate with the assistance of a centralized server. The well-known federated learning approach is able to utilize such architectures by allowing the exchange of only parameters with the server, while keeping the datasets private to each contributing device. In this work, we propose a fully distributed, diffusion-based learning algorithm that does not require a central server and propose an adaptive combination rule for the cooperation of the devices. By adopting a classification task on the MNIST dataset, the efficacy of the proposed algorithm over corresponding counterparts is demonstrated via the reduction of the number of collaboration rounds required to achieve an acceptable accuracy level in non-IID dataset scenarios.

Index Terms—Server-less Federated Learning, Distributed Learning, Diffusion, Adaptive Weights

I. INTRODUCTION

Fueled by high data and computational power availability, Machine Learning (ML) has a transformative impact on our lives by offering high accuracy tools for tasks like natural language processing, object recognition, medical diagnosis and more [1], [2], [3], [4]. The ML tasks are traditionally implemented in a centralized way either by standalone platforms or via cloud-based architectures, where all the data reside and the computations are performed. However, on the one hand, the advent of a new breed of intelligent devices with limited resources [5] and, on the other hand, time-sensitive applications with high reliability requirements, including AR/VR, self-driving vehicles [4], etc., gave rise to the so-called edge ML paradigm [4]. The aim of Edge ML is that ML tasks are performed closer to the end-user where the data are truly available.

Moving towards this paradigm, edge devices (e.g., smart phones, vehicles, base stations) have access to a (possibly limited in size dataset and they are interested in learning a particular model to be used for inference in classification or regression tasks. The devices form a network and collaboratively learn the intended model by exchanging relevant information. The well-known Federated Learning (FL) [6] is one of the first approaches to consider edge ML by relying on a, so-called, parameter server that exchanges model parameters with the edge devices but not their data for privacy reasons.

Currently, there are considerable research efforts that aim to address many of the challenges identified in the frame of FL [7]. Notably, one major challenge in FL is the existence of non Independently and Identically Distributed (non-IID) datasets among the devices, namely, the datasets of particular devices are not representative of the desired task, e.g., in a classification task, data for certain classes are missing. As a remedy to this problem, it has been proposed in literature that FL may be improved by exchanging, apart from parameters, gradient information as well [6], [8], [9]. Additionally, the central server can be a single point of failure, prone to adversarial attacks and a bottleneck when a large number of devices are involved. There have been some works towards avoiding such a server altogether by adopting fully distributed approaches like [10], [11], [12] and references therein.

In this paper, we propose a fully distributed, data-driven learning algorithm and a novel adaptive combination rule. In particular, the highlights of the paper are as follows. Capitalizing on the distributed (adaptive) signal processing literature [13], [14], a distributed learning algorithm of the diffusion-type is introduced where combinations of information among neighboring devices are performed via constant weights. Additionally, a novel adaptive combination rule is devised for the proposed diffusion-based algorithm based on gradient information leading to improved learning rate convergence especially for the challenging non-IID dataset case, with low and constant communication overhead that does not depend on the formed neighborhood sizes as in [11], [12]. Finally, the efficacy of the proposed distributed algorithm with constant and adaptive combinations is demonstrated via a classification task in the MNIST dataset [15] in terms of non-IID datasets and varying number of participating devices.

In the rest of the paper, Sec. II presents the system model and formulates the problem under consideration. In Sec. III,
the proposed fully-distributed algorithm is described along with the adaptive combination rule and a discussion on the communication complexity is provided. Finally, in Sec. IV, the performance of the proposed algorithm is assessed via extensive simulations and Sec. V concludes the paper.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, first, the underlying system model for the network of the cooperating devices (or agents) will be presented and, then, the optimization problem that is to be solved by the network, will be formulated.

A. System Model

In this work, a strongly connected network of $N$ agents is assumed. The network is modeled as a graph $G(N, E)$, where $N$ is the set of nodes that represent the agents and $E$ is the set of edges that connect pairs of agents being able to exchange information (as an example see Fig. 1). The neighborhood of agent $k \in N$ is depicted by the set $N_k$ and consists of all agents $j$ for which the edge $(k, j) \in E$, including agent $k$. Similarly, $N_k'$ is the neighborhood of $k$, excluding this time the agent $k$.

B. Problem Formulation

The aim of the network is to solve in a collaborative way a stochastic optimization problem for a task of common interest by allowing each agent to interact with its neighborhood. In such problems, agent $k$ aims at minimizing a stochastic risk function of the form

$$F_k(w) = \mathbb{E}\{l(w; x_k)\},$$

(1)

where $l(w; x_k)$ is the loss function, $w \in \mathbb{R}^M$ is a vector consisting of the model parameters, relevant to the task at hand (e.g., in a classification problem, these parameters could be the weights of a neural network), $x_k$ refers to the stochastic data and $\mathbb{E}\{\cdot\}$ refers to statistical expectation over the distribution of the data. Then, the network aims at minimizing the overall problem

$$\min_w F(w), \text{ where } F(w) = \sum_{k=1}^N p_k F_k(w),$$

(2)

where $p_k \geq 0$ sum up to one.

Under the assumption that the statistical distribution of the data is unknown, agent $k$ utilizes dataset $D_k$ with $|D_k| = D_k$ elements and substitutes the stochastic risk function in (1) with the corresponding empirical risk function that approximates the statistical expectation with an arithmetic mean of the loss function $l(\cdot)$ over the $D_k$ elements of the dataset. In this case, $p_k = D_k/D$ (where $D = \sum_k D_k$) which is simplified to $p_k = 1/N$ when the data are uniformly distributed among the agents, i.e., $D_k = D$, $\forall i, j$. Finally, it is noted that under the IID and non-IID scenarios, the data elements of all $D_k$’s are produced according to same and different underlying statistical distributions [3], [7], respectively. In the latter case, considering a classification task as an example, this may lead to datasets that do not capture any data for certain classes [11].

III. FULLY DISTRIBUTED FEDERATED LEARNING

In this section, first, fully distributed federated learning is addressed by proposing a distributed diffusion-based algorithm with constant combination weights. Then, an adaptive combination rule is proposed in which each agent utilizes its neighbors’ gradient information to determine the combination weights at each communication round. It is noted that this rule, although designed here for the proposed diffusion algorithm, it can also be applied to other distributed algorithms like the one (of the consensus type) proposed in [11]. Finally, the communication cost of the proposed algorithm with constant and adaptive weights is analysed in terms of the required information exchanges and compared with other relevant variants.

A. Distributed Diffusion-based Federated Learning

Driven by previous works on distributed parameter estimation [15], the proposed algorithm adopts the diffusion approach for enabling the agents to collaboratively solve (2) and builds upon the so-called Adapt-then-Combine (ATC) strategy. This strategy has been shown to provide enhanced performance and stability guarantees when compared to the Combine-then-Adapt alternative [13], [15].

Before describing the proposed algorithm in detail, let us first present the two main steps involved generally in the ATC strategy, which are the following.

$$\psi_{k,t} = w_{k,t-1} - \mu_k \nabla F_k(w_{k,t-1})$$

(3)

$$w_{k,t} = \sum_{l \in N_k} a_{lk} \psi_{l,t}$$

(4)

In the first step (see (3)), agent $k$ adapts its local parameter vector $w_{k,t-1}$ using a Stochastic Gradient Descent (SGD) update and produces an intermediate vector $\psi_{k,t}$. To achieve this, an approximation $\nabla F_k$ of the true gradient $\nabla F_k$ is employed with a step size $\mu_k$. Then, in the second step (see (4)), after receiving the corresponding intermediate parameter vectors from its neighbors, agent $k$ employs a convex combination of all intermediate vectors using the weights $a_{lk}$, $l \in N_k$, which are non-negative and add up to one, and, thus, the new parameter vector $w_{k,t}$ is created. Note that in the context of...
Algorithm 1 Distributed Diffusion-based FL

1: procedure DIFFUSION LEARNING($N_k, T, E, \{\mu_l\}, B$)
2: \hspace{1em} Initialize: $w_{k,0}, t = 0$
3: \hspace{1em} for each epoch $e = 1, 2, ..., E$ do
4: \hspace{2em} for each communication round $n = 1, 2, ..., T$ do
5: \hspace{3em} $t \leftarrow t + 1$
6: \hspace{3em} $\psi_{k,t} \leftarrow \text{Adapt}(w_{k,t-1})$ \hspace{1em} // Adapt Step
7: \hspace{3em} if ConstantWeights then
8: \hspace{4em} Send($\psi_{k,t}$)
9: \hspace{4em} Receive($\{\psi_{l,t}\}_{l \in N_k}$)
10: \hspace{4em} Use (5) to set $a_{lk}$’s
11: \hspace{3em} else if AdaptiveWeights then
12: \hspace{4em} $\delta_{k,t} = \psi_{k,t} - w_{k,t-1}$
13: \hspace{4em} Send($\delta_{k,t}, \psi_{k,t}$)
14: \hspace{4em} Receive($\{\delta_{l,t}\}_{l \in N_k}, \{\psi_{l,t}\}_{l \in N_k}$)
15: \hspace{4em} Use (2) to set $a_{lk}$’s
16: \hspace{3em} end if
17: \hspace{3em} $w_{k,t} \leftarrow \sum_{l \in N_k} a_{lk} \psi_{l,t}$ \hspace{1em} // Combine Step
18: \hspace{3em} end for
19: \hspace{2em} end for
20: end procedure

21: procedure ADAPT($w_{k,t-1}$)
22: \hspace{1em} $B \leftarrow$ mini-batches of size $b$
23: \hspace{1em} $\psi_{k,t} \leftarrow \text{SGD}(\mu_k, B, w_{k,t-1})$
24: end procedure

B. Adaptive Gradient-based Combination Rule

The proposed distributed algorithm, as other learning algorithms in the machine learning literature, relies on SGD updates that employ an approximate gradient that is computed over the available dataset. In order to approximate gradient to be an unbiased estimate of the true gradient, the SGD algorithm assumes (and requires) an IID sampling of the data comprising the involved dataset [9].

However, this assumption might be problematic in the FL or fully distributed learning cases as such data sampling could be difficult to achieve. The main reason of non-IID datasets in distributed settings is the fact that they are probably captured by agents with different geographical and temporal particularities [7]. The existence of non-IID data has undesirable effects during the training phase (when a model is learned by solving, e.g., minimization problems like (2)), leading to reduced convergence speed or even diverging from the desired model [9]. This behavior is a consequence of the close relation between each agent’s local model $w_k$ and the statistical distribution of its data. A large number of local updates will lead towards a local minimum of the loss function $F_k(w_k)$ that will be inconsistent with the desired global solution. This inconsistency is accumulated during training leading to an increased number of communication rounds before convergence is achieved.

Taking this behavior into account, the convergence speed of the learning procedure can be improved via more sophisticated combination rules for determining the weights $a_{lk}$. To this end, an adaptive rule is proposed building upon the approach devised in [9] for the classical FL case where a parameter server exists. The main idea for this rule is that gradient information is taken into account at each communication round in order to assess each agents’ “divergence” from the optimal minimization path. In the frame of FL, this is achieved by the parameter server via comparing its approximate global gradient with local approximate gradients sent by the agents and weighing the contribution of each agent accordingly. However, this is not possible in fully distributed learning settings. Instead, during a communication round, agent $k$ approximates the global gradient in terms of its neighborhood by aggregating the involved gradients as follows.

$$
\nabla F_k^q(w_{k,t}) = \sum_{l \in N_k} a_{lk} \nabla F_l(w_{l,t-1}) = -\sum_{l \in N_k} a_{lk} \frac{\delta_{l,t}}{\mu_l}.
$$

The $a_{lk}$’s are defined as in (5), while it is reminded that the $\delta_{l,t}$’s, $l \in N_k$, have been sent to agent $k$.

The comparison of the approximate global gradient in (6) with the local gradient of agent $l \in N_k$ is performed in terms of the angle between the involved vectors. This is determined using the well-known dot-product equation as follows.

$$
\theta_{k,l}(t) = \arccos \left( \frac{-\delta_{l,t}^T \nabla F_k^q(w_{l,t})}{\|\delta_{l,t}\| \|\nabla F_k^q(w_{l,t})\|} \right),
$$

where $(\cdot)^T$ is the transpose operator and $\|\cdot\|$ is the Euclidean norm.
For the eventual determination of the weights, the remaining steps are the same as the ones in [9] and they are presented in the following for completeness. Thus, first, a smoothed version of $\theta_{k,t}(t)$ is used to deal with its random nature.

$$\tilde{\theta}_{k,t}(t) = \frac{t-1}{t} \tilde{\theta}_{k,t}(t-1) + \frac{1}{t} \theta_{k,t}(t), \quad t \geq 1$$

(8)

Then, the adaptive combination rule for determining the time-varying weights $a_{lk}(t)$ is given as

$$a_{lk}(t) = \frac{D_l e^f(\tilde{\theta}_{k,t}(t))}{\sum_{j=1}^{N_k} D_l e^f(\tilde{\theta}_{j,t}(t))},$$

where $N_k = |N_k|$, $f(x) = a(1 - e^{-a(x-1)})$ is a variant of the Gompertz function and $a$ is a scalar [9]. Note that (9) does not depend on the dataset sizes when the datasets are balanced, i.e., of the same size.

Finally, it is mentioned that the proposed adaptive combination rule can be also applied to other fully distributed FL algorithms like the one in [11] which is of the consensus type. In this case, a minor modification is required as agent $k$ combines information from agents in $N_k$, i.e., excluding itself. Therefore, the computations of (6)-(9) can be adjusted appropriately.

C. Communication Cost Analysis

The proposed algorithm and the consensus-based one in [11] require from agent $k$ to broadcast to its neighborhood the local model (with $M$ parameters) at each communication round when the constant weights are used. On the other hand, when the adaptive weights are employed in both algorithms, agent $k$ broadcast both the local model and the local approximate gradient, thus, $2M$ parameters need to be transmitted. In the literature, other relevant methods can be found that utilize gradient information like [11] and [12] and references therein.

To the best of our knowledge, in all cases, the communication overhead is much higher and depends on the neighborhood size of agent $k$, thus, leading to the transmission of $O(MN_k)$ parameters, where $N_k = |N_k|$. For this reason, they are not considered in the following simulations.

IV. SIMULATIONS

In this section, the performance of the proposed diffusion-based FL algorithm employing the constant and adaptive combination rules, is studied. In the simulations, the consensus-based federated averaging algorithm of [11] (properly modified so as to incorporate the same weighting rules) is also presented. Furthermore, for benchmarking purposes, the performance of centralized and individual training is also depicted, namely, when all data are gathered in a central server which performs the training and when each agent operates individually with no cooperation, respectively. For the evaluation of the performance, the task of classification on the well-known MNIST dataset is considered (consisting of 60K training and 10K testing images of handwritten digits, in which the classifier is a fully connected neural network with two hidden layers as in [10] and cross-entropy loss is used. For all implementations, MATLAB $^\text{©}$ 2021b and the deep learning toolbox have been used.

The performance is investigated via Monte-Carlo simulations adopting a challenging non-IID scenario and the mean testing accuracy over the network of agents is depicted versus the epochs. At each instance (out of 10 in total), the status of every agent (i.e., whether it has an IID dataset or not) and which classes compose the non-IID datasets, are selected in random. Subsequently, for each IID agent, the dataset is randomly selected from the whole training set of MNIST images, while, for the non-IID agents, the corresponding datasets are randomly selected by considering only the selected classes per agent. At the end of each epoch, mean accuracy is measured using the MNIST testing images. For the simulations, two networks of size $N = 4$ (see the simple linear network in [11]) and $N = 20$ (see [18] and Fig. 1), are considered. Furthermore, $\mu_k = 0.01$ and $D_k = 600$, for $k$, while $b = 10$, $T = 6$, $E = 30$, and $a = 5$ in [9].

In Fig. 2 the results for the network with the $N = 4$ agents is depicted. As expected, the distributed algorithms exhibit improved performance when compared with the no cooperation alternative. Furthermore, the proposed algorithm with adaptive weights demonstrates faster convergence speed versus the other variants, meaning that it reaches a certain accuracy level with less epochs, which leads to less communication rounds (it is reminded that $T = 6$ rounds per epoch are employed here). Similarly, in Fig. 3 the results for the larger network with $N = 20$ agents, is depicted. Here, the network is composed of richer neighborhoods and more cooperating agents. Thus, better testing accuracy can is achieved. Moreover, in this larger network, the algorithms with the adaptive rules are better than the ones with the constant rule, while the diffusion-based one with adaptive weights demonstrates faster convergence, especially, during the first epochs.

Finally, in Figs. 4 and 5 the performance of the algorithms is demonstrated for two specific distributions of the non-IID agents on the network with $N = 20$. In particular, in Fig. 4 the five central agents (namely, 5, 11, 12, 13, 14) are non-IID and observe at most five (out of ten) classes. Similarly, in Fig. 5 five edge nodes (namely, 1, 20, 9, 8, 19) are now non-IID. As observed, the adaptive rule is crucial on the performance of the distributed algorithms while the behavior of the diffusion-based and consensus-based counterparts is reversed. This is an interesting remark that was not observed in the previous mean results and deems further investigation in the near future.

V. CONCLUSIONS

In this work, the problem of fully distributed diffusion-based FL has been studied. Also, a novel adaptive combination rule has been devised that speeds up the convergence rate of learning, especially under the challenging non-IID dataset scenario. The performance of the proposed algorithm has been assessed in comparison with other counter-parts of the same communication complexity in the context of a typical classification task using the well-known MNIST dataset.
Fig. 2. $N = 4$ agents with non-IID datasets.

Fig. 3. $N = 20$ agents with non-IID datasets.

Fig. 4. 5 central agents (out of 20) with non-IID datasets.

Fig. 5. 5 edge agents (out of 20) with non-IID datasets.

REFERENCES

[1] W. G. Hatcher and W. Yu, “A survey of deep learning: platforms, applications and emerging research trends.” IEEE Access, vol. 6, pp. 24411–24432, 2018.

[2] Z.- Q. Zhao, P. Zheng, S.- t. Xu, and X. Wu, “Object detection with deep learning: A review,” IEEE transactions on neural networks and learning systems, vol. 30, no. 11, pp. 3212–3232, 2019.

[3] K. Suzuki, “Overview of deep learning in medical imaging,” Radiological physics and technology, vol. 10, no. 3, pp. 257–273, 2017.

[4] J. Park et al., “Wireless network intelligence at the edge,” Proceedings of the IEEE, vol. 107, no. 11, 2019.

[5] J. Tang, D. Sun, S. Liu, and J.- L. Gaudiot, “Enabling deep learning on iot devices,” Computer, vol. 50, no. 10, pp. 92–96, 2017.

[6] B. McMahan et al., “Communication-efficient learning of deep networks from decentralized data,” in Artificial intelligence and statistics. PMLR, 2017, pp. 1273–1282.

[7] P. Kairouz et al., “Advances and open problems in federated learning,” Foundations and Trends® in Machine Learning, vol. 14, no. 1–2, 2021.

[8] 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.

[9] H. Wu and F. Wang, “Fast-convergent federated learning with adaptive weighting,” IEEE Transactions on Cognitive Communications and Networking, vol. 7, no. 4, pp. 1078–1088, 2021.

[10] Y. Qu et al., “Serverless federated learning for uav networks: Architecture, challenges, and opportunities,” arXiv e-prints, 2021.

[11] S. Savazzi, M. Nicoli, and V. Rampa, “Federated learning with cooperating devices: A consensus approach for massive iot networks,” IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4641–4654, 2020.

[12] Y. Esfandiari et al., “Cross-gradient aggregation for decentralized learning from non-iid data,” in International Conference on Machine Learning. PMLR, 2021, pp. 3036–3046.

[13] A. H. Sayed, “Adaptation, learning, and optimization over networks,” Foundations and Trends in Machine Learning, vol. 7, pp. 311–801, 2014.

[14] I. Plata-Chaves, N. Bogdanović, and K. Berberidis, “Distributed diffusion-based lms for node-specific adaptive parameter estimation,” IEEE Transactions on Signal Processing, vol. 63, no. 13, 2015.

[15] Y. Lecun et al., “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] J. Chen and A. H. Sayed, “Diffusion adaptation strategies for distributed optimization and learning over networks,” IEEE Transactions on Signal Processing, vol. 60, no. 8, pp. 4289–4305, 2012.

[17] A. H. Sayed, “Chapter 9 - diffusion adaptation over networks,” in Academic Press Library in Signal Processing: Vol. 3. Elsevier, 2014.

[18] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” Proceedings of the IEEE, vol. 95, no. 1, pp. 215–233, 2007.