Fast and Communication Efficient Decentralized Learning with Local Updates

Peyman Gholami [1] **Hulya Seferoglu** [1]

Abstract

Gossip and random walk-based learning are widely considered decentralized learning algorithms. Gossip algorithms (both synchronous and asynchronous) suffer from high communication cost, while random-walk based learning experiences high convergence time. In this paper, we design a fast and communication-efficient asynchronous decentralized learning mechanism DIGEST by taking advantage of both Gossip and random-walk ideas, and focusing on stochastic gradient descent (SGD). DIGEST is an asynchronous decentralized learning mechanism building on local-SGD, which is originally designed for communication efficient centralized learning. We analyze the convergence of DIGEST and prove that it approaches to the optimal solution asymptotically for both iid and non-iid data distributions. We evaluate the performance of DIGEST for logistic regression and a deep neural network ResNet20. The simulation results confirm that multi-stream DIGEST has nice convergence properties; its convergence time outperforms the baselines when data distribution is non-iid.

1. Introduction

Decentralized algorithms have been extensively studied in the literature, with Gossip algorithms receiving the lion's share of research attention (Boyd et al., 2006b; Nedic & Ozdaglar, 2009; Koloskova et al., 2019; Aysal et al., 2009; Duchi et al., 2012; Kempe et al., 2003; Xiao & Boyd, 2003; Boyd et al., 2006a; Koloskova et al., 2020; Scaman et al., 2019; Giaretta & Girdzijauskas, 2019). In Gossip algorithms, each node (edge or end user device) has its own locally kept model on which it effectuates the learning by talking to its neighbors. This makes Gossip attractive from a failure-tolerance perspective. However, this comes at the expense of a high network resource utilization. All nodes in a Gossip algorithm in a synchronous mode perform a model update and wait for receiving model updates from their neighbors. When a node completes receiving updates from its neighbors, it aggregates them. Thus, there should be data communication among all nodes after each model update, which is a significant communication overhead.

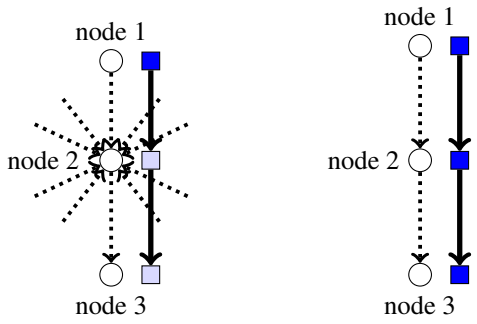
Asynchronous Gossip algorithms, where nodes communicate asynchronously and without waiting for others are promising to reduce idle nodes and eliminate the stragglers, *i.e.,* delayed nodes (Lian et al., 2018; Assran et al., 2019; Li et al., 2018; Avidor & Tal-Israel, 2022; Nadiradze et al., 2020). However, nodes still rely on iterative Gossip averaging of their models, so updates propagate gradually across the network. Such delayed updates, also referred as gradient staleness in asynchronous Gossip may lead to high error floors (Dutta et al., 2021), or require very strict assumptions to converge to the optimum solution (Lian et al., 2018). Moreover, such methods must be implemented with caution to prevent the occurrence of deadlocks (Assran et al., 2019). The need for multiple rounds of Gossip averaging, in both synchronous and asynchronous algorithms, to distribute a node's update to all other nodes tends to diminish the updates after each averaging. The "diminishing updates" are more emphasized when a model passes through high degree nodes, and detrimental for the convergence in when data distribution heterogeneous across the nodes.

In both synchronous and asynchronous Gossip, models propagate over the nodes and updated by each node gradually as seen Fig. 1(a). This may lead to a notion that we name "diminishing updates", where a node's update (e.g., node 1 in Fig. 1(a)), even though crucial for convergence, may be averaged and mixed with other models in the next node (e.g., node 2 in Fig. 1(a)). The diminishing updates are more emphasized when a model passes through high degree nodes, and detrimental for the convergence when data distribution is heterogeneous across the nodes.

If Gossip algorithms are one side of the spectrum of decentralized learning algorithms, the other side is random-walk based decentralized learning (Bertsekas, 1996; Ayache & Rouayheb, 2021; Sun et al., 2018; Needell et al., 2014; Spiridonoff et al., 2021). The random-walk algorithms advocate activating a node at a time, which would update the global model with its local data. Then, the node selects one of its

[1]Department of Electrical and Computer Engineering, University of Illinois Chicago, Chicago, Illinois, United States. Correspondence to: Peyman Gholami <pghola2@uic.edu>.

Workshop of Federated Learning and Analytics in Practice, colocated with 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA.



(a) Gossip (b) DIGEST

Figure 1: Spread of information in a decentralized network.

neighbors randomly and sends the updated global model. The selected neighbor becomes a newly activated node, so it updates the global model using its local data. This continues until convergence. Random-walk algorithms reduce the communication cost as well as computation with the cost of increased convergence time due to idle times at nodes.

The goal of this work is to take advantage of both Gossip and random-walk ideas to design a fast and communication-efficient decentralized learning. Our key intuitions are; (i) Nodes do not need to communicate as much as Gossip to update their models, i.e., a sporadic exchange of model updates is sufficient; (ii) the diminishing updates inherent to Gossip algorithms can be eliminated by employing a global model (detailed in Section 3.2); and (iii) Nodes do not need to wait idle as in random walk. Thus, we design a fast and communication-efficient asynchronous decentralized learning mechanism DIGEST by particularly focusing on stochastic gradient descent (SGD). DIGEST is an *asynchronous decentralized learning* algorithm building on local-SGD algorithms, which are originally designed for communication efficient *centralized learning* (Stich, 2019; Wang & Joshi, 2021; Lin et al., 2020). DIGEST works as follows. Each node keeps updating its local model all the time as in local-SGD. Meanwhile, there is an ongoing stream of global model update among nodes. We note that the exchanged models are global models as each node adds its own local updates to the received model. A node that has the global model selects the next node randomly among its neighbors for global model transmission. After all the nodes update their models with a global model, DIGEST pauses global model exchange, while local SGD computations still continue. The global model exchange is repeated at every H iterations. We name this algorithm single-stream DIGEST. We further improve the convergence time of single-stream DIGEST by enabling multiple streams of global model updates, which is multi-stream DIGEST. We analyze the convergence of single- and multi-stream DIGEST, and prove that both algorithms approach to the optimal solution asymptotically. The simulation results confirm that DIGEST has nice convergence properties. The convergence time of multi-stream DIGEST outperforms the baselines in non-iid setting.

2. Preliminaries

Network Topology. We model the underlying network topology with a connected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (nodes) and $\mathcal{E}$ is the set of edges. The vertex set contains V nodes, *i.e.,* $|\mathcal{V}| = V$, and $|.|$ shows the size of the set. The computing capabilities of nodes are arbitrary and heterogeneous. If node i is connected to node j through a communication link and can transmit data, then link $\{i, j\}$ is in the edge set, *i.e.,* $\{i, j\} \in \mathcal{E}$. The set of the nodes that node i is connected to and can transmit data is called the neighbors of node i, and the neighbor set of node i is denoted by $\mathcal{N}_i$. We do not make any assumptions about the behavior of the communication links; there can be an arbitrary, but finite amount of delay over the links.

Data. We consider a setup where nodes have access to a subset of data samples $\mathcal{D}$. Each node v has a local dataset $\mathcal{D}_v$, where $D_v = |\mathcal{D}_v|$ is the size of the local dataset and $D = \sum_{v=1}^{V} D_v$. The distribution of data across nodes is not identical and independently distributed (non-iid).

Stochastic Optimization. Assume that the nodes in the network jointly minimize a d-dimensional function $f : \mathbb{R}^d \to \mathbb{R}$. The goal of the nodes is to converge on a model $\mathbf{x}^*$, which minimizes the empirical loss over D samples, *i.e.,* $\mathbf{x}^* := \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left[f(\mathbf{x}) := \frac{1}{D} \sum_{i=1}^{D} f_i(\mathbf{x}) \right]$, where $f_i(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ is the loss function of $\mathbf{x}$ associated with the data sample i. The optimum solution is denoted by f^*. The loss function on local dataset $\mathcal{D}_v$ at node v is $f^v(\mathbf{x}) = \frac{1}{D_v} \sum_{i\in\mathcal{D}_v} f_i(\mathbf{x})$.

Notation. The notation table is in Appendix A of (Gholami & Seferoglu, 2023).

3. Design of DIGEST

In this section, we provide the design principle of multi-stream DIGEST. The single stream version is a straightforward extension and its details are provided in (Gholami & Seferoglu, 2023).

3.1. Tree Construction and Multiple Streams

Multi-stream DIGEST operates over a rooted tree. Thus, our first step is to create a rooted tree from our undirected graph G. We use a classical distance vector routing algorithm such as Bellman-Ford so that each node v learns its delay distance d_{vu}^G to node u in a decentralized manner and via message passing. We define the radius of node v as R_v^G, which is the largest distance from node v; i.e., $R_v^G = \max_u\{d_{vu}^G\}$. The root of the network is the node with the smallest R_v^G, i.e., $r = \arg\min_v\{R_v^G\}$, where r is the root node. The shortest delay tree ST_r rooted with r is constructed in a decentralized manner as each node keeps d_{vu}^G information.

After the tree is constructed, multiple streams are created to exchange the global model in the network. First, the root

Algorithm 1 DIGEST on node $v \in \mathcal{V}$ with multiple streams.

1: **Initialization:** $\mathbf{x}_0^v = \mathbf{x}_0$, $\mathbf{x}_{-1}^v = \mathbf{x}_0$, $queue = ()$.
2: **for** m in $\mathcal{M}_v$ **do**
3: $\tilde{\mathbf{x}}_0[m] = \mathbf{x}_0$, $\tilde{\mathbf{x}}_{-1}[m] = \mathbf{x}_0$, $visited[m] = \{\}$, $pre\_node[m] = v$, $\mathcal{S}_T^v[m] = \{0\}_{0<t\leq T}$, $s_1^{v_m}[m] = 1$.
4: **for** t in $0, ..., T-1$ **do**
5: Sample i_t^v uniformly from $\mathcal{D}_v$.
6: Start computing the gradient $\nabla f_{i_t^v}(\mathbf{x}_t^v)$.
7: $\mathbf{x}_{t+1}^v = \mathbf{x}_t^v - \sum_{z \in u_t^v} \eta_z \nabla f_{i_z^v}(\mathbf{x}_z^v)$
8: **if** $queue \neq ()$ **then**
9: **for** any $message$ in $queue$ **do**
10: $(\tilde{\mathbf{x}}_t[m], visited[m], pre\_node[m], m) \leftarrow message$
11: $s_{t+1}^v[m] = 1$
12: Remove $message$ from $queue$
13: **for** m in $\mathcal{M}_v$ **do**
14: **if** $s_{t+1}^v[m] = 1$ **then**
15: $\tilde{\mathbf{x}}_{t+1}[m] = \tilde{\mathbf{x}}_t[m] + \frac{D_v}{D}(\mathbf{x}_{t+1}^v - \mathbf{x}_{-1}^v) + (\mathbf{x}_{-1}^v - \tilde{\mathbf{x}}_{-1}[m])$
16: $\mathbf{x}_{t+1}^v = \tilde{\mathbf{x}}_{t+1}[m]$
17: $\mathbf{x}_{-1}^v = \mathbf{x}_{t+1}^v$ ▷ Last updated model at v
18: $\tilde{\mathbf{x}}_{-1}[m] = \tilde{\mathbf{x}}_{t+1}[m]$ ▷ Last updated model at node v corresponding to stream m
19: **if** $\mod(t, H_m) = 0$ or $visited[m] \neq \mathcal{V}_m$ **then**
20: Send $message = \tilde{\mathbf{x}}_{t+1}[m], visited[m], pre\_node[m], r$ to a neighboring node.
21: **else**
22: $s_{t+H-mod(t,H_m)}^v[m] = 1$
23: $visited[m] = \{\}$

node creates a number of streams which is equal to the number of its children. Each of these streams has a range, which starts from the root node and ends at a child node if the child node itself has more than one child. In that case, the child node behaves exactly as a root node, and creates multiple streams towards its children by following the same rule that we just described for the root node. Eventually, there will be M streams in the tree, and the set of the streams that go through node v is $\mathcal{M}_v$. The set of nodes that are in the range of stream m is $\mathcal{V}_m$. The set of the streams between root r and node v is defined as P_v^r.

3.2. Algorithm Design

Multi-stream DIGEST is summarized in Alg. 1. The following are the key properties of Alg. 1.

There are multiple global models in different streams, *i.e.,* $\tilde{\mathbf{x}}_t[m]$ corresponds to the global model in stream m out of M streams. There are $|\mathcal{M}_v|$ models stored in each node, *i.e.,* $\tilde{\mathbf{x}}_{-1}[m]$ to represent the global model corresponding to the last synchronization of stream m at node v. We define $visited[m]$, $pre\_node[m]$, and $s_t^v[m]$ for each stream m.

Each node v has a $queue$ to store all the messages that a node receives from its neighbors. It is initialized as an empty queue at the start. Whenever node v receives a $message$ from one of its neighbors, it is added in the queue. Each node can receive up to $|\mathcal{M}_v|$ messages related to different streams, so the size of the $queue$ is $|\mathcal{M}_v|$. In each message there is a stream index m (line 10).

Node v extracts all the messages in its queue (line 9-12). Then, it updates its global and local models in line 15, 16 if $s_{t+1}^v[m] = 1$. The global model is updated using the most recent local updates of node v and global updates of other streams (line 15). The global model synchronization continues until all nodes in $\mathcal{V}_m$ are visited for stream m. Then, global model update is paused until a new synchronization round, which starts at every H_m iteration. The policy for selecting H_m is explained in the next section.

4. Convergence Analysis of DIGEST

We use the following assumptions for the convergence analysis of single- and multi-stream DIGEST.

1. **Smooth local loss.** f^v is continuously differentiable and its gradient is L-Lipschitz for $1 \leq v \leq V$, *i.e.,* $\|\nabla f^v(\mathbf{y}) - \nabla f^v(\mathbf{x})\| \leq L\|\mathbf{y} - \mathbf{x}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.
2. **Bounded local variance.** The variance of the stochastic gradient is bounded for all nodes, *i.e.,* $0 \leq t < T$, $1 \leq v \leq V$, $\mathbb{E}_{i_t^v} \|\nabla f_{i_t^v}(\mathbf{x}_t^v) - \nabla f^v(\mathbf{x}_t^v)\|^2 \leq \sigma^2$.
3. **Bounded diversity.** The diversity of the local loss functions and global loss function is bounded, *i.e.,* $0 \leq t < T$, $1 \leq v \leq V$, $\|\nabla f^v(\mathbf{x}_t^v) - \nabla f(\mathbf{x}_t^v)\|^2 \leq \zeta^2$.
4. **Bounded lag.** We assume bounded lag, *i.e.,* $\max\{l_t^v - t\} \leq E, \quad 0 \leq t < T, 1 \leq v \leq V$.
5. **Bounded synchronization interval.** For single-stream DIGEST, we assume that the interval between two subsequent global model synchronizations is bounded, *i.e.,* $gap(\mathcal{S}_T^v) \leq H$, $1 \leq v \leq V$, where $gap(\mathcal{S}_T^v)$ shows the maximum gap between two subsequent 1s in $\mathcal{S}_T^v$. For multi-stream DIGEST, we assume different bounds for each stream, i.e., $gap(\mathcal{S}_T^v[m]) \leq H_m$ for $m \in \mathcal{M}_v$.
6. **Convexity.** f is μ-(strongly) convex, *i.e.,* $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|^2$.

Theorem 4.1. *Let assumptions 1-5 hold, with a constant and small enough learning rate $\eta \leq \frac{1}{30LA}$ (potentially depending on T), the convergence rate of single- and multi-stream DIGEST is as follows:*

Non-convex: $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\hat{\mathbf{x}}_T)\|^2$ *is*

$$O\Bigg(\frac{FLA}{T} + \sigma\sqrt{\frac{\rho LF}{T}} + \Big(\frac{LF\sqrt{\sigma^2 A + \zeta^2 A^2}}{T}\Big)^{\frac{2}{3}}\Bigg),$$

where $\hat{\mathbf{x}}_T = \sum_{v=1}^{V}\sum_{t=0}^{T-1} \frac{D_v}{D}\mathbf{x}_t^v$.

Convex: *Under assumption 6 for $\mu \geq 0$, $\mathbb{E}\, f(\hat{\mathbf{x}}_T) - f^*$ is*

$$O\Bigg(\frac{RLA}{T} + \sigma\sqrt{\frac{\rho R}{T}} + \big(\frac{R\sqrt{L(\sigma^2 A + \zeta^2 A^2)}}{T}\big)^{\frac{2}{3}}\Bigg),$$

where $\hat{\mathbf{x}}_T = \sum_{v=1}^{V}\sum_{t=0}^{T-1}\frac{D_v}{D}\mathbf{x}_t^v$.

Strongly-convex: *Under assumption 6 for $\mu > 0$, $\mathbb{E}\, f(\hat{\mathbf{x}}_T) - f^*$ is*

$$\tilde{O}\Bigg(RLA\exp(\frac{-\mu T}{LA}) + \frac{\rho\sigma^2}{\mu T} + \frac{L(\sigma^2 A + \zeta^2 A^2)}{\mu^2 T^2}\Bigg),$$

where $\hat{\mathbf{x}}_T = \frac{1}{DW_T}\sum_{v=1}^{V}\sum_{t=0}^{T-1} D_v\omega_t\mathbf{x}_t^v$, $\omega_t = (1-a\eta)^{-(t+1)}$, $W_T = \sum_{t=0}^{T-1}\omega_t$.

$\tilde{O}$ hides constants and poly-logarithmic factors, T represent the wall clock time, $F := f(\mathbf{x}_0) - f^*$, $R := \|\mathbf{x}_0 - \mathbf{x}^*\|^2$, $A := H' + E$, and $\rho := \sum_{v=1}^{V}(\frac{D_v}{D})^2$. The convergence rate of single-stream DIGEST follows when $H' = H$, and the convergence rate of multi-stream DIGEST is obtained by putting $H' = \max_v \sum_{m\in P_v^r} H_m$ in A. □

Proof. The proof of Theorem 4.1 is provided in (Gholami & Seferoglu, 2023). □

Remark 4.2. For strongly-convex case, in iid data distribution over nodes, i.e., $\zeta = 0$, the convergence rate to the optimum value f^* is $\tilde{O}(\frac{\rho}{T})$ given that $H' + E = \tilde{O}(\rho T)$ is satisfied, where $\rho = \sum_{v=1}^{V}(\frac{D_v}{D})^2$ is a data concentration coefficient that can take values between $\frac{1}{V} \leq \rho < 1$. In non-iid data distribution over nodes ($\zeta \neq 0$), linear speed up $O(\sqrt{\frac{\rho}{T}})$ is achieved when $H' + E = \tilde{O}(\sqrt{\rho T})$ holds.

Theorem 4.1 and Remark 4.2 show a nice trade-off between convergence rate and communication overhead. It determines how much communication is needed to achieve a linear speed-up. Remark 4.2 also shows the impact of non-iid data distribution, which requires smaller H', hence more communications to converge and achieve linear speed-up.

Remark 4.3. Corollary 4.2 shows that the linear speed up is achieved when $T = \tilde{\Omega}(\frac{H'}{\rho})$ and $T = \tilde{\Omega}(\frac{H'^2}{\rho})$ for iid and non-iid data, respectively. When the network is larger, single-stream DIGEST needs longer H' (which is equal to H) to visit all the nodes, which requires larger T (convergence time). But in multi-stream DIGEST, H' defined as $H' = \max_v \sum_{m\in P_v^r} H_m$ could be as low as R_r^G, which is the radius of root node r (or maximum delay toward any node from root node r). As R_r^G does not necessarily increase with the size of the network, multi-stream DIGEST is plausible even for large networks.

Remark 4.4. Lets assume that the network can be covered in H iteration using single/multi-stream approach. DIGEST can efficiently perform synchronization while nodes are doing local-SGD, i.e., network topology, spectral gap or the maximum and minimum degrees in the network topology don't affect the convergence rate. This is one advantage of using DIGEST in comparison to previous works on asynchronous decentralized learning like (Nadiradze et al., 2020) where the convergence rate in non-convex setting is $O\Big(\frac{F}{\sqrt{HVT}} + \frac{\sqrt{H}(\sigma^2+H\zeta^2)}{\sqrt{VT}} + \frac{V d_{max} L^2 H^3 G^2}{d_{min}\lambda^2 T}\Big)$. Here, we observe that the minimum degree (d_{min}), maximum degree (d_{max}), and spectral gap (λ) of the network graph are part of the result, so affects the convergence.

5. Evaluation of DIGEST

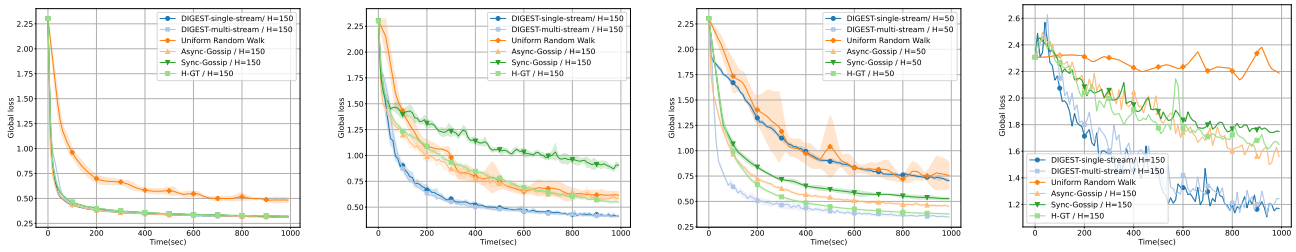
5.1. Convergence Properties

We evaluate DIGEST as compared to baselines; (i) Uniform Random-Walk (URW) (Ayache & Rouayheb, 2021); (ii) Gradient tracking (GT) with local-SGD (Liu et al., 2023): It is an algorithm that is developed to overcome data heterogeneity across nodes in a decentralized optimization problems; (iii) Async-Gossip Lian et al. (2018) with local-SGD; (iv) Sync-Gossip Lian et al. (2018) with local-SGD. Our codes are provided in (dig, 2023).

We consider two network topologies; an Erdős–Rényi graph of $V = 10$ and $V = 100$ nodes with 0.3 as the probability of connectivity. We assume that each iteration of Local SGD takes 0.1 second. The communication delay between every two neighbors is assumed to have exponential distribution where its average is randomly chosen from 0 to 5 seconds.

We use two data distribution: (i) iid-balanced, and (ii) non-iid-unbalanced. In iid-balanced case, data is shuffled and equally divided and placed in nodes. Non-iid-unbalanced has two features: (i) Non-iid, which is realized by sorting data according to their labels, and distribute them in the sorted order. Thus, the data distributed over nodes will be non-iid; (ii) Unbalanced, which means that each node may have different amount of data. We use geometric series to realize unbalanced data across nodes. For example, if a node u has $D_u = \delta$ data, the next nodes get $\delta\rho$, $\delta\rho^2$, etc. data, where ρ is determined by taking into account the size of the total dataset D.

We first examine the convergence performance of logistic regression. We run the optimization using tuned constant learning rate for each algorithm. Fig. 2 shows the convergence behavior of our algorithms as well as the baselines for *MNIST* dataset in 10-nodes and 100-nodes topologies. URW generally underperforms as compared to other methods due to its approach of conducting only one local-SGD operation per iteration on a single node. As a consequence, it does not have a linear speed-up with increasing number of nodes. In certain situations involving non-iid data distribution, URW may exhibit better performance than some other methods as shown in Figs. 2b, 2c. This is because URW is not affected by non-iidness as it uniformly incorporates data from all nodes. DIGEST, Sync-Gossip, and Async-Gossip have similar performance in iid data distribution in Fig. 2a. On the other hand, we observe that Gossip based algorithms are suffering from slow convergence in non-iid setting as shown



(a) 10-nodes / iid-balanced / Logistic Regression / *MNIST*

(b) 10-nodes / noniid-unbalanced / Logistic Regression / *MNIST*

(c) 100-nodes / noniid-unbalanced / Logistic Regression / *MNIST*

(d) 10-nodes / noniid-unbalanced / DNN (ResNet-20) / *CIFAR-10*

Figure 2: Convergence results in terms of global loss over wall-clock time.

in Figs. 2b, 2c. We also observe that GT algorithms enhance the performance of gossip-based algorithms by incorporating a mechanism to overcome non-iidness. However, this algorithm demands twice the communication overhead compared to sync-Gossip, resulting in more communication overhead, which can degrade its convergence performance in terms of wall-clock time. In comparison, DIGEST have better convergence behavior thanks to its very design of spreading information uniformly in the network to handle non-iidness. It is evident that when the network is larger, one-stream DIGEST method is unable to cover the entire network as quickly as required, highlighting the need to utilize multi-stream DIGEST to overcome this limitation. This observation is supported in Fig. 2c, where all streams have the same $H_m = H, m \in \mathcal{M}$ in multi-stream DIGEST.

Next, we evaulate the convergence performance of our algorithms for ResNet-20 (He et al., 2015) as the DNN model. The dataset is *CIFAR-10* (Krizhevsky, 2009). We have set the batch size to 36 per node, and the learning rate is decayed by a constant factor after completing 50% and 75% of the training time. The initial value of the learning rate is separately tuned for each algorithm. We have set the momentum value to 0.9 and the weight decay to 10^{-4}. We observe that in non-iid settings, where communication and model distribution across the network become crucial, DIGEST outperforms Gossip-based algorithms, Fig. 2d.

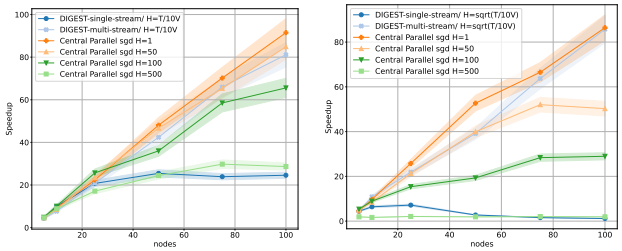
5.2. Speed-up

In this section, we evaluate the speed up performance of our DIGEST algorithms as well as the baseline; centralized parallel SGD. We consider the following cost function

$$f(x) = \begin{cases} (x-1)^2 & x \geq 1, \\ \frac{(x-1)^2}{2} & x < 1. \end{cases} \tag{1}$$

We employ Local-SGD at node v with gradients affected by a normal noise, i.e., $\nabla f_{i_t^v}(\mathbf{x}_t^v) = \nabla f(\mathbf{x}_t^v) + n_t^v$, where $n_t^v \sim \mathcal{N}(\zeta_v, \sigma^2)$, $\sum_{v=1}^{V} \zeta_v = 0$. To create the speed-up curve, we divide the expected error of a single node SGD by the expected error of each method at the last iteration T for different number of nodes. As in linear speed-up, error decreases linearly with the increasing number of workers, so we expect to see a straight line on the graph. The speed-up curve is illustrated in Fig. 3. The central parallel SGD averages all nodes‘ updates at every H steps, and updates the model in all nodes. It is worth noting that the central parallel SGD with $H = 1$ is the best speed-up that can be achieved in this scenario.

We set the learning rate to 0.001, and $|\zeta_v| = 5$ for $v \in \mathcal{V}$, $\sigma = 5$, and $T = 10^4$. Note that in iid setting with a less restrictive constraint on H, larger H can still leads to linear speed-up when compared to non-iid setting. Moreover, it is seen that single stream DIGEST has linear speed-up to a certain limit; however, as the number of nodes increases and single-stream DIGEST cannot traverse the entire network fast enough, linear speed-up is not maintained. On the other hand, multi-stream DIGEST achieves linear speed up and achieves a very close performance to the best possible scenario, which is centralized parallel SGD with $H = 1$.



(a) iid / balanced

(b) non-iid / unbalanced

Figure 3: Speed-up curves for DIGEST.

6. Acknowledgments

This work was supported in part by ARL under Grant W911NF-2120272, and in part by NSF under Grant CCF-1942878, Grant CNS-2148182, and Grant CNS-2112471.

7. Conclusion

We designed fast and communication-efficient decentralized learning mechanisms; single- and multi-stream DIGEST to exploit the convergence rate and communication overhead tradeoff. We proved both algorithms converge to the optimal solution asymptotically for both iid and non-iid data. The simulation results confirms that the convergence rate of DIGEST is better than or comparable to the baselines.

References

Digest codes, 2023. Available at https://www.dropbox.com/s/sowfdwfj0chs1z0/DIGEST-codes.zip?dl=0 and https://github.com/Anonymous404404/DigestCode.git.

Assran, M., Loizou, N., Ballas, N., and Rabbat, M. G. Stochastic gradient push for distributed deep learning. 2019.

Avidor, T. and Tal-Israel, N. Locally asynchronous stochastic gradient descent for decentralised deep learning. *ArXiv*, abs/2203.13085, 2022.

Ayache, G. and Rouayheb, S. E. Private weighted random walk stochastic gradient descent. *IEEE Journal on Selected Areas in Information Theory*, 2(1):452–463, 2021. doi: 10.1109/JSAIT.2021.3052975.

Aysal, T., Yildiz, M. E., Sarwate, A., and Scaglione, A. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, 57:2748–2761, 2009.

Bertsekas, D. P. A new class of incremental gradient methods for least squares problems. *SIAM J. Optim*, 7:913–926, 1996.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006a. doi: 10.1109/TIT.2006.874516.

Boyd, S. P., Ghosh, A., Prabhakar, B., and Shah, D. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52:2508–2530, 2006b.

Duchi, J. C., Agarwal, A., and Wainwright, M. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57:592–606, 2012.

Dutta, S., Joshi, G., Ghosh, S., Dube, P., and Nagpurkar, P. Slow and stale gradients can win the race. *IEEE Journal on Selected Areas in Information Theory*, 2:1012–1024, 2021.

Gholami, P. and Seferoglu, H. Digest: Fast and communication efficient decentralized learning with local updates. *ArXiv*, abs/2307.07652, 2023.

Giaretta, L. and Girdzijauskas, S. Gossip learning: Off the beaten path. In *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1117–1124, Los Alamitos, CA, USA, dec 2019. IEEE Computer Society. doi: 10.1109/BigData47090.2019.9006216. URL https://doi.ieeecomputersociety.org/10.1109/BigData47090.2019.9006216.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Kempe, D., Dobra, A., and Gehrke, J. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pp. 482, USA, 2003. IEEE Computer Society. ISBN 0769520405.

Koloskova, A., Stich, S., and Jaggi, M. Decentralized stochastic optimization and gossip algorithms with compressed communication. *ArXiv*, abs/1902.00340, 2019.

Koloskova, A., Loizou, N., Boreiri, S., Jaggi, M., and Stich, S. A unified theory of decentralized SGD with changing topology and local updates. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5381–5393. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/koloskova20a.html.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Li, Y., Yu, M., Li, S., Avestimehr, A. S., Kim, N. S., and Schwing, A. G. Pipe-sgd: A decentralized pipelined sgd framework for distributed deep net training. *ArXiv*, abs/1811.03619, 2018.

Lian, X., Zhang, W., Zhang, C., and Liu, J. Asynchronous decentralized parallel stochastic gradient descent. In Dy, J. G. and Krause, A. (eds.), *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3049–3058. PMLR, 2018.

Lin, T., Stich, S. U., and Jaggi, M. Don't use large mini-batches, use local sgd. *ArXiv*, abs/1808.07217, 2020.

Liu, Y., Lin, T., Koloskova, A., and Stich, S. U. Decentralized gradient tracking with local steps, 2023.

Nadiradze, G., Sabour, A., Davies, P., Markov, I., Li, S., and Alistarh, D. Decentralized sgd with asynchronous, local and quantized updates. *arXiv: Learning*, 2020.

Nedic, A. and Ozdaglar, A. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54:48–61, 2009.

Needell, D., Srebro, N., and Ward, R. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pp. 1017–1025, Cambridge, MA, USA, 2014. MIT Press.

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal convergence rates for convex distributed optimization in networks. *Journal of Machine Learning Research*, 20(159):1–31, 2019. URL http://jmlr.org/papers/v20/19-543.html.

Spiridonoff, A., Olshevsky, A., and Paschalidis, I. Communication-efficient SGD: From local SGD to one-shot averaging. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=UpfqzQtZ58.

Stich, S. U. Local sgd converges fast and communicates little. *ArXiv*, abs/1805.09767, 2019.

Sun, T., Sun, Y., and Yin, W. On markov chain gradient descent. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 9918–9927, Red Hook, NY, USA, 2018. Curran Associates Inc.

Wang, J. and Joshi, G. Cooperative sgd: A unified framework for the design and analysis of local-update sgd algorithms. *Journal of Machine Learning Research*, 22(213): 1–50, 2021. URL http://jmlr.org/papers/v22/20-147.html.

Xiao, L. and Boyd, S. P. Fast linear iterations for distributed averaging. *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, 5:4997–5002 Vol.5, 2003.