Convergence and Communication Trade-offs in Centralized and Decentralized SGD

Handong Xue, Shutong Peng

***Abstract*—This report studies the trade-offs between centralized mini-batch SGD and decentralized SGD (D-SGD) algorithms in distributed optimization settings. A detailed theoretical analysis characterizes stochastic variance reduction and convergence rates, with emphasis on the role of the communication topology in D-SGD. In D-SGD, the spectral properties of the mixing matrix play a key role in the rate of consensus thus also overall convergence. Convergence behavior and communication cost of centralized SGD and D-SGD were evaluated across three topologies: ring, toroidal grid, and fully connected networks. Experiments on synthetic non-IID datasets with convex and strongly convex setting demonstrate that while centralized and fully connected D-SGD achieve comparable convergence speed, grid-based D-SGD offers a favorable balance between convergence and communication cost. These findings offer insights into larger-scale, non-convex settings such as neural network training [1], where communication topology and variance reduction remain critical to convergence efficiency.**

I. Introduction

Distributed stochastic gradient descent (SGD) is a cornerstone of large-scale machine learning [2], enabling the training of models across multiple nodes in parallel. In traditional centralized systems, a parameter server coordinates updates by aggregating gradients from all worker nodes. While this design simplifies model synchronization, it introduces scalability bottlenecks due to its reliance on global communication and centralized coordination.

Decentralized SGD (D-SGD) offers an alternative that eliminates the need for a central server [3]. Instead, each node maintains a local copy of the model and communicates only with neighboring nodes in a predefined network topology. This decentralized approach distributes the communication load, making it well-suited for edge computing and in scenarios where communication is slow or unreliable.

The performance of D-SGD is highly influenced by the underlying communication topology, which governs how quickly local models converge to a consensus. The convergence rate depends not only on standard optimization parameters, such as step size and variance, but also on the spectral properties of the mixing matrix that encodes the network topology. Topologies with larger spectral gaps have faster information mixing and convergence, whereas sparse topologies can suffer from slow consensus.

This report presents a comparative study of centralized and decentralized SGD under various topologies. The report provides theoretical convergence and variance bounds for both methods under convex and strongly convex settings. Furthermore, the empirical performance is evaluated by using synthetic non-IID datasets, focusing on convergence speed, consensus error, and communication cost. The results reveal the trade-offs between network connectivity and communication efficiency, and guide the selection of topologies for distributed learning systems based on bandwidth and scalability constraints.

II. Algorithms

In this section, two distributed stochastic optimization approaches are presented. The centralized method uses a parameter server to aggregate and broadcast updates, while the decentralized method only uses direct communication among nodes.

A. *Centralized Synchronous Mini-Batch SGD*

In the conventional centralized SGD approach, a single global model x is maintained. At each iteration, a subset of nodes(or possibly all nodes) send their gradient information to a central parameter server, which updates the model. For example, consider the objective function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f(x_i;\xi_i)$, where $f(x_i;\xi_i) : \mathbb{R}^d \to \mathbb{R}$ is the local objective function based on the data ξ residing at node i.

The centralized synchronous mini-batch SGD process is as follows:

1) The central server broadcasts the current model parameters x_t to all N participating worker nodes.
2) Each worker i computes a stochastic gradient at the current model. Specifically, worker i draws b i.i.d. samples $\{\xi_{i,t}^{(j)}\}_{j=1}^{b}$ (of size b, therefore total mini-batch size per round is $B = Nb$) and computes the average gradient
$$g_{i,t} := \frac{1}{b}\sum_{j=1}^{b} \nabla f_i(x_t;\xi_{i,t}^{(j)}).$$
3) All worker nodes transmit their computed stochastic gradients back to the central server.
4) The server waits to receive gradients from all worker nodes, then computes the averaged gradient:
$$g_t = \nabla\left(\frac{1}{N}\sum_{i=1}^{N} f_{i,t}\right) = \frac{1}{N}\sum_{i=1}^{N} g_{i,t}$$
5) The server updates the global model using the averaged gradient:
$$x_{t+1} = x_t - \eta_t g_t$$
where η_t is the step size at iteration t.
6) The server send back the updated x_{t+1} back to all worker nodes. Repeats step 1-5.

This procedure is synchronous, that is, all workers use the same x_t and updates use all N results. And it is equivalent to single-worker SGD with a mini-batch of size B per iteration, but implemented in parallel across N machines. Such parallel mini-batch SGD is a standard strategy to speed up training in distributed settings [4], [5].

Assumptions

Centralized mini-batch SGD behaves similarly to single-machine SGD in terms of convergence, with the benefit of processing more data per iteration. Assuming [6], [9]:

1. **Convexity.** Each $f(x_i; \xi_i)$ is convex, thus f is also convex.
2. **L-Lipschitz Smoothness.** Each $\nabla f_i(x_i; \xi_i)$ is L-Lipschitz continuous with constant $L > 0$, thus ∇f also has Lipschitz smoothness.
3. **Unbiased Stochastic Gradients Estimator.** That is, $\mathbb{E}[g(x; \xi)] = \nabla f(x)$, where ξ denotes the random mini-batch used to compute the stochastic gradient. In other words, each data point in the node's local dataset is randomly selected. This ensures that the stochastic gradient estimate points in the correct direction on average:

$$\begin{aligned}\mathbb{E}[g_t] &= \mathbb{E}[\frac{1}{N}\sum_{i=1}^{N} g_{i,t}] \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[g_{i,t}] \\ &= \frac{1}{N}\sum_{i=1}^{N}\nabla f_t \qquad \text{(Unbiased Local Gradients)} \\ &= \nabla f_t \end{aligned} \tag{1}$$

   If stochastic gradients were biased, the updates would push the parameters away from the direction of the true gradient.
4. **Bounded Variance.** $\mathrm{Var}(g_{i,t}) \leq \sigma^2$ for some $\sigma^2 \geq 0$. This limits the amount of noise introduced by using mini batch data instead of the full dataset.
5. **Strong Convexity When Specializing to Strongly Convex Objectives.** Assume f is *μ-strongly convex*: $\forall x, y$,

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}\|y - x\|^2.$$

   Strong convexity ensures a unique minimizer $x^* = \arg\min_x f(x)$ and faster convergence rates.

Variance Reduction with Mini-Batches

An important effect of mini-batches is the reduction of the gradient variance by averaging. Under the above assumptions, the averaged gradient g_t is an unbiased estimator of the true global gradient ∇f, i.e., $E[g_t] = \nabla f_t$. Crucially, the variance of this estimator is reduced by the number of workers[1]: $\mathrm{Var}[g_t] \leq \frac{\sigma^2}{N}$.

Proof: [2]

$$\begin{aligned}\mathrm{Var}[g_t] &= \mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N}(g_{i,t})\right] \\ &= \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}[g_{i,t}] \\ &\leq \frac{1}{N^2}\sum_{i=1}^{N}\sigma^2 \qquad \text{(Bounded Variance Assumption)} \\ &= \frac{1}{N^2}\cdot N\sigma^2 \\ &= \frac{\sigma^2}{N}\end{aligned}$$

□

More generally, if each of the N workers averages b samples (total batch $B = Nb$), the variance of ∇f_t is

$$\mathrm{Var}(g_t) \leq \frac{\sigma^2}{Nb} = \frac{\sigma^2}{B}. \tag{2}$$

Thus, using a mini-batch of size B yields a B-fold reduction in the stochastic gradient variance relative to single-sample SGD. Equivalently, the noise scale σ^2 in the convergence analysis can be replaced by σ^2/B when B samples are used per iteration. This is a key reason why mini-batch (parallel) SGD can converge in fewer iterations: averaging over more samples produces a gradient estimate closer to the true gradient. [4], [9].

Convergence Rates

Using the bounded variance lemma above, centralized mini-batch SGD achieves the following convergence rates:

- **Convex Case.** Under the above assumptions, mini-batch SGD almost converges linearly. With a diminishing step-size schedule η_t (e.g., $\eta_t = \eta_0/\sqrt{t}$), one can show the expected objective value converges to the optimum. In particular, it is well known [10] that vanilla SGD (with batch size $B = 1$) closes optimality gap after T iterations at

$$\mathbb{E}[f(\bar{x}_T) - f(x^*)] = \mathcal{O}\left(\frac{\log(T)}{\sqrt{T}}\right)$$

  where $\mathbb{E}[f(\bar{x}_T) - f(x^*)]$ is the expected suboptimality, and averaged iterate $\bar{x}_T = \frac{1}{\sum \eta_t}\sum \eta_t x_t$.

  Proof:[3]
  Let x^* be the minimizer of f. Starting from the update $x_{t+1} = x_t - \eta_t g_t$, where g_t is the mini-batch gradient

[1] Stich 2018, Lemma 3.2 [4].

[2] Assuming each worker node uses a single sample for simplicity. Since worker nodes have independent gradient variance,

[3] This proof is derived from Theorem 3.4, [7] and Corollary 2.2, [8]

with batch size $B = 1$. Let $D^2 = \mathbb{E}\|x_0 - x^*\|^2$, choosing $\eta_t = min\{\frac{1}{2L}, \frac{D}{\sigma\sqrt{t+1}}\}$[4]. For the distance to x^*,

$$\|x_{t+1} - x^*\|^2 = \|x_t - x^*\|^2 - 2\eta_t g_t^T(x_t - x^*) + \eta_t^2\|g_t\|^2$$

Taking expectation and using convexity assumption ($\nabla f(x_t)^T(x_t - x^*) \geq f(x_t) - f(x^*)$), as well as the unbiased gradient and bounded variance assumption:

$$\begin{aligned}\mathbb{E}\|x_{t+1} - x^*\|^2 &\leq \mathbb{E}\|x_t - x^*\|^2 - 2\eta_t\mathbb{E}[f(x_t) - f(x^*)] \\ &\quad + \eta_t^2\mathbb{E}\|g_t\|^2 \\ &\leq D^2 - 2\eta_t\mathbb{E}[f(x_t) - f(x^*)] \\ &\quad + \eta_t^2\left(\|\nabla f(x_t)\|^2 + \sigma^2\right) \qquad (*)\end{aligned}$$

Summing (*) over $t = 0$ to $T-1$, telescoping, and using L-smoothness assumption, $\|\nabla f(x_t)\|^2 \leq 2L(f(x_t) - f(x^*))$:

$$2\sum_{t=0}^{T-1}\eta_t\mathbb{E}[f(x_t) - f(x^*)] \leq D^2 + \sigma^2\sum_{t=0}^{T-1}\eta_t^2\mathbb{E}[f(x_t) - f(x^*)]$$

Since $\eta_t = min\{\frac{1}{2L}, \frac{D}{\sigma\sqrt{t+1}}\}$, this ensures $2L\eta_t^2 \leq \eta_t^2$, so it simplifies to

$$\sum_{t=0}^{T-1}\eta_t\mathbb{E}[f(x_t) - f(x^*)] \leq D^2 + \sigma^2\sum_{t=0}^{T-1}\eta_t^2 \qquad (**)$$

The average iterate $\bar{x}_T = \frac{1}{\sum\eta_t}\sum\eta_t x_t$, Jensen's inequality gives

$$f(\bar{x}_T) = f(\frac{1}{\sum_{t=0}^{T-1}\eta_t}\sum_{t=0}^{T-1}\eta_t x_t) \leq \frac{1}{\sum_{t=0}^{T-1}\eta_t}\sum_{t=0}^{T-1}\eta_t f(x_t)$$

Taking expectation and subtract x^*,

$$\mathbb{E}[f(\bar{x}_T) - f(x^*)] \leq \frac{1}{\sum_{t=0}^{T-1}\eta_t}\sum_{t=0}^{T-1}\eta_t\mathbb{E}[f(x_t) - f(x^*)]$$

Substitute (**),

$$\mathbb{E}[f(\bar{x}_T) - f(x^*)] \leq \frac{D^2 + \sigma^2\sum_{t=0}^{T-1}\eta_t^2}{\sum_{t=0}^{T-1}\eta_t}$$

For $\eta_t = \frac{D}{\sigma\sqrt{t+1}}$ (dominant for large t),

$$\sum_{t=0}^{T-1}\eta_t \leq \frac{2D\sqrt{T}}{\sigma}, \quad \sum_{t=0}^{T-1}\eta_t^2 \leq \frac{D^2}{\sigma^2}\log(T)$$

Averaging over iterates $\bar{x}_T = \frac{1}{\sum\eta_t}\sum\eta_t x_t$:

$$\begin{aligned}\mathbb{E}[f(\bar{x}_T) - f(x^*)] &\leq \frac{D^2 + \sigma^2 \cdot \frac{D^2\log(T)}{\sigma^2}}{2 \cdot \frac{2D\sqrt{T}}{\sigma}} \\ &= \mathcal{O}\left(\frac{D\sigma\log(T)}{\sqrt{T}}\right)\end{aligned}$$

Thus, $\mathbb{E}[f(\bar{x}_T) - f(x^*)] = \mathcal{O}\left(\log(T)/\sqrt{T}\right)$ with $\eta_t = \mathcal{O}(1/\sqrt{t})$

□

With a mini-batch of size B used at each iteration, the convergence improves. Each iteration is effectively B times more "sample thorough", and the error after T iterations is[5]

$$\boxed{\mathbb{E}[f(\bar{x}_T) - f(x^*)] = \mathcal{O}\left(\frac{\log(T)}{\sqrt{BT}}\right)}$$

This indicates that a batch of size B yields about a $\sqrt{B}$ times reduction in error for a given T. In other words, to reach a target accuracy ϵ such that $\mathbb{E}[f(\bar{x}_T) - f(x^*)] \leq \epsilon$, one needs $T = O(1/B\epsilon^2)$ iterations[6] if using batch size B (note $T = O(1/\epsilon^2)$ for batch size 1).

- **Strongly Convex Case.** When the objective function f is μ-strongly convex (Assumption 5), SGD converges faster than in the general convex case. Using a diminishing step-size of the form $\eta_t = c/t$ (for some constant c), the expected error in function value decreases at the rate

$$\mathbb{E}[f(\bar{x}_T) - f(x^*)] = \mathcal{O}\left(\frac{\log(T)}{T}\right),$$

as shown in [11]. This $\mathcal{O}(\log(T)/T)$ rate is significantly faster than the $\mathcal{O}(\log(T)/\sqrt{T})$ rate obtained for general convex objectives. Intuitively, strong convexity ensures a uniform curvature around the optimum, which helps the iterates contract more reliably toward the minimum as optimization progresses.

In the case of parallel mini-batch SGD, the gradient variance can be further reduced by averaging over B samples at each iteration, similar to the convex case.

$$\boxed{\mathbb{E}[f(\bar{x}_T) - f(x^*)] = \mathcal{O}\left(\frac{\log(T)}{BT}\right)}$$

Therefore, to reach an expected error of ϵ, $T = \mathcal{O}(1/B\epsilon)$ iterations are needed. When $B = 1$, this reduces to the standard result $T = \mathcal{O}(1/\epsilon)$.

In both cases, by using B samples per round, B-times fewer iterations are required. If those B samples are obtained in parallel from N workers (so $B = Nb$), this translates into roughly an N times reduction in wall-clock time to ϵ. That is of course, without considering communication costs.

However, the total number of samples processed remains the same: $B \times T = \mathcal{O}(1/\epsilon^2)$ for convex, $\mathcal{O}(1/\epsilon)$ for strong convex. So while

[4]Early iterations use the maximal stable step size $1/2L$ to exploit problem curvature, while later iterations use $\mathcal{O}(1/\sqrt{t})$ decay to suppress stochastic variance.

[5]Some sources uses $\mathcal{O}(\log(T)/\sqrt{NT})$, which is equivalent as $B = Nb$.

[6]To make the analysis easier, use the approximation $\log(T) = O(\log(1/\epsilon))$, so that

$$T = O\left(\frac{\log^2(1/\epsilon)}{N\,\epsilon^2}\right)$$

and dropping the $\log^2$ term.

increasing B reduces the number of iterations and speeds up training, it does not reduce the total data needed to reach the target accuracy. Larger mini-batches improve computational speed but offer diminishing returns in statistical accuracy [6].

Dekel et al. [5] showed that using very large batches eventually becomes inefficient — the benefit from adding more samples per round decreases, and the gain per extra sample can vanish entirely.

Communication Costs

A primary limitation of standard synchronous centralized SGD is the communication overhead inherent in each iteration. The typical protocol requires [12]:

1) Each of the N workers sending its gradient vector (size d) to the server.

2) The server broadcasting the updated parameter vector (size d) back to the N workers.

This results in a total communication volume of approximately $\mathcal{O}(Nd)$ per iteration. This cost can become a major performance bottleneck, particularly under certain conditions:

- **Network Constraints**: Limited network bandwidth, or high latency between worker nodes and the server can cause communication time to dominate computation time. This issue is especially daunting if the servers are separated afar. Moreover, a single failed server can stall the entire system, thus robustness is not guaranteed.

- **Model Size**: For models with a very large number of parameters d, the volume of data (d parameters per worker) to be transferred becomes problematic.

- **Scalability with** N: As the number of workers N increases, the central parameter server must handle incoming traffic from N sources, potentially become congested.

In practice, parameter reduction is often performed using collective communication[7]. One common approach is All-Reduce, which can distribute the communication workload more evenly than a traditional parameter server. While this can improve efficiency, the total communication volume may still scale with the number of nodes N and the parameter dimension d, depending on the specific algorithm used [12].

[7]Libraries like Nvidia's NCCL provide highly optimized versions commonly used in distributed deep learning. It is also one of the three built-in backends of `pytorch.distributed`. https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html, https://pytorch.org/docs/stable/distributed.html

B. *Decentralized Synchronous SGD (D-SGD)*

Decentralized Synchronous SGD is a distributed optimization algorithm where each worker node in the network maintains its own model parameters, and only communicates with a subset of neighbors at each synchronization step, rather than to a central server. Let each worker node i with N in total hold a local objective $f_i(x)$, where $f_i : \mathbb{R}^d \to \mathbb{R}$ based on its local data, and the global objective function is $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$. The local data $\mathcal{D}_\rangle$ (size b) is a partition of the full dataset $\mathcal{D}$ (size B), that is, $\mathcal{D} = \cup_{i=1}^{N} D_i, D_i \cap D_j = \emptyset$. Each node i maintains its own local copy of model parameters, denoted as $x_{i,t}$. at iteration t. Communication occurs only between connected neighbors, which is governed by a mixing matrix $W = [W_{ij}], W \in \mathbb{R}^{N\times N}$, and W is symmetric, doubly stochastic with spectral radius ¡ 1, that is, $W_{ij} > 0$ if and only if nodes i, j are connected (or if $i = j$).

The D-SGD procedure is synchronous, all nodes perform each step simultaneously. The process for each iteration t, performed by all nodes $i = 1...N$ in parallel is as follows:

1) Each nodes i computes a stochastic gradient $\nabla f_{i,t}$ using a mini-batch $\xi_{i,t}^{(j)}$ i.i.d. sampled from its local data $\mathcal{D}_i$:

$$g_{i,t} = \frac{1}{b}\sum_{j=1}^{b} \nabla f_i(x_{i,t}; \xi_{i,t}^{(j)})$$

2) All nodes simultaneously transmit their current parameters to their neighbors. With a predefined mixing matrix $W = [W_{i,j}]$, each node computes

$$y_{i,t} \;=\; \sum_{j\in\mathcal{N}(i)\cup\{i\}} W_{ij}\, x_{j,t},$$

where $\mathcal{N}(i)$ denotes the set of neighbors of node i.

3) Using the averaged model $y_{i,t}$, perform a gradient step:

$$x_{i,t+1} = y_{i,t} - \eta_t g_{i,t}$$

where η_t is the step size at t.

4) All nodes complete step 3 before proceeding to $t + 1$. Repeat step 1-3.

By this procedure, each node moves in a descent direction informed by its local data and a consensus with neighbors. Over many iterations, the goal is for all $x_{i,t}$ to converge to a consensus optimum x^* that minimizes the global $f(x)$. Notably, no central parameter server is needed – nodes only exchange information with their neighbors, making the method fully decentralized.

Mixing Matrix

At step 2, every worker node make consensus with its neighbors' models before taking gradient step. This mixing matrix controls the network topology of D-SGD [13]. The

mixing weights are stored in $W \in \mathbb{R}^{N \times N}$. It has the following properties:

- **Convex Combination.** Each row sums to 1, so each node keeps a convex combination of its own parameters and its neighbors', as in step 2.

- **Sparsity.** If nodes i and j do not talk to each other, $W_{i,j} = 0$.

- **Spectral Properties.** They determine the efficiency of how nodes reach consensus, which is governed by the eigenvalues of W. Particularly, the largest eigenvalue λ_1, and the second largest eigenvalue ρ (both in terms of magnitude).

  - Largest eigenvalue $\lambda_1 = 1$. This corresponds to consensus stability eigenvector $\mathbf{v}_1 = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T$ This ensures that if all nodes start with identical parameters, they remain unchanged as $W\mathbf{v}_1 = \mathbf{v}_1$ (Lemma 2, [14]).
  - Second-largest eigenvalue $\rho = max\{|\lambda_2|...|\lambda N|\}$. This governs the exponential decay rate of consensus errors (Eq.12, [13]).

$$\|W^t\mathbf{x} - \bar{\mathbf{x}}\| \leq c\rho^t\|\mathbf{x} - \bar{\mathbf{x}}\|$$

  where $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} x_i$, and c is a constant. W^t is the t iteration W is applied. Smaller ρ gives a larger spectral gap $1-\rho$, thus gives a faster convergence, and vice versa [14].

A commonly used initialization is *Metropolis-Hastings Weights* [13]. It is popular because each worker node only needs its own degree d_i and its neighbors' degree d_j.

$$W_{ij} = \begin{cases} \frac{1}{1+max\{d_i,d_j\}} & j \in \mathcal{N}(i) \\ 1 - \sum_{j \in \mathcal{N}(i)} W_{ij} & j = i \\ 0 & \text{otherwise} \end{cases}$$

W also satisfies symmetry, $W_{ij} = W_{ji}$ besides of the weight properties discussed above.

Topologies

The network topology[8] plays a crucial role in D-SGD's communication patterns and convergence behavior [15].

- **Ring Topology**. Each node has exactly two neighbors, forming a loop (Fig. 1). Communication is localized, thus parameter update information propagates around the ring gradually. The degree of each node is 2, which means at each iteration a node exchanges at most two messages (one with each neighbor).

[8]There are also exponential, star topologies.

Fig. 1: Ring topology. Each colored node represent a worker node.

For example, a N node system, degree $d_i = 2$ for all nodes. Its Metropolis-Hastings weight is

$$W_{ij} = \begin{cases} \frac{1}{3} & j \in \mathcal{N}(i) \\ 0 & \text{otherwise} \end{cases}$$

That is,

$$W_{ring} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 1 & 1 & 0 & \ddots & 0 & 0 \\ 0 & 1 & 1 & 1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & 1 & 1 & 1 & 0 \\ 0 & 0 & \ddots & 0 & 1 & 1 & 1 \\ 1 & 0 & \cdots & 0 & 0 & 1 & 1 \end{bmatrix}$$

with eigenvalues[9]

$$\begin{aligned} \lambda_k &= \frac{1}{3}(1 + e^{(2\pi ik/N)} + e^{(-2\pi ik/N)}) \\ &= \frac{1}{3}(1 + 2cos(2\pi k/N)) \end{aligned}$$

where $k = 0, 1, ...N-1$, spectral gap

$$1 - \rho = 1 - \frac{1}{3}(1 + 2cos(2\pi/N))$$

This minimal connectivity reduces per-node communication overhead to $\mathcal{O}(d)$ per iteration, where d is the dimension of x. However, reaching global consensus can be slower: the ring topology has a small spectral gap. D-SGD on a ring may require more iterations to converge compared to better-connected networks, because it takes longer for a gradient update from one node to influence distant nodes.

- **Grid Topology**. This topology (Fig. 2) places $N = m \times k$ nodes on an $m \times k$ lattice. Interior nodes has a top, down, left, right neighbor, degree $d_i = 4$. Edge nodes has $d_i = 3$, and corner nodes have $d_i = 2$. To avoid edge effect and make things simpler[10], instead one can use a toroidal

[9]"Circulant Matrix" on Wikipedia. https://en.wikipedia.org/wiki/Circulant_matrix

[10]As one can reuse the circulent matrix formula, and makes Metropolis-Hastings matrix calculation much easier.

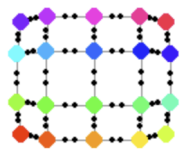

Fig. 2: Grid Topology. Each colored node represent a worker node. In toroidal grid, each edge nodes connects to the edge nodes on the other side.

grid $\sqrt{N} \times \sqrt{N}$ and let $m = \sqrt{N}$, where each node is connected to four neighbors $d_i = 4$.
In a N node system, its Metropolis-Hastings weight is

$$W_{ij} = \begin{cases} \frac{1}{5} & j \in \mathcal{N}(i) \\ 0 & \text{otherwise} \end{cases}$$

That is (for 3x3 grid), $W_{grid} \in \mathbb{R}^{9\times 9}$,

$$W_{grid} = \frac{1}{5}\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

with eigenvalues

$$\begin{aligned} \lambda_{k_1,k_2} &= \frac{1}{5}(1 + e^{(2\pi i k_1/m)} + e^{(-2\pi i k_1/m)} \\ &\quad + e^{(2\pi i k_2/m)} + e^{(-2\pi i k_2/m)}) \\ &= \frac{1}{5}(1 + 2cos(2\pi k_1/m) + 2cos(2\pi k_2/m)) \end{aligned}$$

where $k_1, k_2 = 0, 1, ...m-1$, spectral gap

$$1 - \rho = 1 - \frac{1}{5}(1 + 4cos(2\pi/m))$$

The per-node communication overhead is still $\mathcal{O}(d)$ per iteration, but due to its larger spectral gap, reaching global consensus is faster than that of ring topology.

- **Fully Connected Topology**. Each node communicates with all other nodes (Fig. 3), thus degree = $N-1$. In each sync step, every node can directly average its parameters with every other node's parameters. In a N node system, its Metropolis-Hastings weight is

$$W_{ij} = \frac{1}{N}$$

That is,

$$W_{fc} = \frac{1}{N}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \ddots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

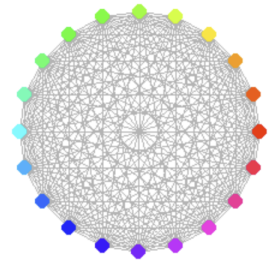

Fig. 3: Grid Topology. Each colored node represent a worker node.

where eigenvalues are $\lambda_1 = 1, \lambda_k = 0$ for $k \neq 1$. Thus spectral gap ρ is exactly 1.
Such complete mixing yields the fastest consensus. since the spectral gap is maximal. One round of communication achieves exact consensus averaging. In fact, with full connectivity, D-SGD effectively becomes equivalent to mini-batch SGD using all nodes' data at every step. The downside is communication cost: each node must send updates to $N-1$ peers, an $O(Nd)$ communication workload per node per iteration. Therefore, fully connected topology is not feasible for very large networks due to bandwidth limits.

In summary, denser topologies improve convergence per iteration (fewer iterations needed) but at higher per-iteration communication cost. Sparser topologies reduce per-iteration communication, but typically require more iterations to reach the same accuracy. Therefore, topology should be chosen considering this trade off. A ring is scalable in network size but sacrifices convergence speed, whereas a fully connected graph minimizes required iterations at the cost of $\mathcal{O}(Nd)$ messages per node. Intermediate topologies like grid balances these factors.

Assumptions

For D-SGD, all assumptions required for centralized SGD must also hold. In addition, two further assumptions specific to the decentralized setting are necessary, as outlined below [13]:

1) **Symmetry, Double Stochasticity, and Spectral Gap.** The mixing weight matrix W must be symmetric, double stochastic, and possess a good spectral gap, $\lambda_1(W) = 1 > \lambda_2(W) \geq \cdots \geq \lambda_n(W) > -1$

2) **Connectivity.** The communication graph must be connected, meaning there exists a communication path between any pair of nodes. Connectivity ensures that information can eventually propagate throughout the entire network.

These assumptions are automatically satisfied when using Metropolis-Hastings weights.

Variance Reduction with Mini-Batches and Mixing

In D-SGD, each node computes a stochastic gradient using a local mini-batch and averages its model with neighbors based on a mixing matrix W [3].

The variance of the DSGD can be defined as [11]

$$\text{Var}[\bar{g}_t] \leq \frac{\sigma^2}{N} + \mathcal{O}\left(\frac{\eta_t^2 \sigma^2}{1-\rho}\right),$$

where the first term represents for the variance due to local mini-batch sampling. This is the same as the centralized SGD. The second term represents for the consensus error, or the disagreement in gradients due to dis-synchronization across the network. This is due to the mixing step, and topologies with better connectivity improves this error.

Convergence Rates

Since D-SGD implementing the similar bounded variance assumptions, the convergence rates mirror the centralized SGD, with additional error due to communication delay and network topology.

- **Convex Case.** The error after T iterations with the batch size B for convex case is [12]

$$\boxed{\mathbb{E}[f(\bar{x}_T) - f(x^*)] = \mathcal{O}\left(\frac{1}{\sqrt{BT}} + \frac{1}{T(1-\rho)}\right)}$$

  The first term is the same as the centralized case. For the second term, the spectral gap $1-\rho$ captures how well the topology mixes information. One needs $T = \mathcal{O}(1/B\epsilon^2 + 1/\epsilon(1-\rho))$ iterations to reach target optimality ϵ, and the network connectivity has a direct impact on the iterations needed. This directly shows a larger spectral gap reduces the consensus error and improving convergence, and vice versa.

- **Strongly Convex Case.** The error after T iterations with the batch size B for strongly convex case is

$$\boxed{\mathbb{E}[f(\bar{x}_T) - f(x^*)] = \mathcal{O}\left(\frac{1}{BT} + \frac{1}{T^2(1-\rho)^2}\right)}$$

  The first term again matches the centralized rate. The second term however now scales with $1/T^2(1-\rho)^2$. One needs $T = \mathcal{O}(1/B\epsilon + 1/\sqrt{\epsilon}(1-\rho))$ iterations to reach target optimality ϵ, where $1-\rho$ is multiplied by a steeper value $\sqrt{\epsilon}$. The stronger dependency on $1-\rho$ again indicates that small spectral gaps has an amplified effect on convergence in the strongly convex setting.

[11] The second term is adapted from Theorem 4 in [3].

[12] Simplified from Theorem 1 in [3]. The original result includes explicit constants involving step size, Lipschitz constant, gradient variance, gradient dissimilarity across nodes. Here, a high-level $\mathcal{O}(\cdot)$ form in terms of suboptimality is used.

Communication Costs

As discussed in topology section, this cost structure varies significantly with the chosen topology [13], [15]:

- **Ring Topology**: Each node sends data to 2 neighbors, yielding a per-node communication cost of $\mathcal{O}(d)$ and an overall network communication volume of $\mathcal{O}(Nd)$.

- **Grid Topology**: Each node sends data to typically 4 neighbors, also resulting in a per-node cost of $\mathcal{O}(d)$ and an overall cost of $\mathcal{O}(Nd)$.

- **Fully Connected Topology**: Each node communicates with all other $N-1$ nodes. This leads to a per-node cost of $\mathcal{O}(Nd)$, which mirrors centralized mini-batch SGD interacting with a server, but results in a significantly higher overall network communication volume of $\mathcal{O}(N^2 d)$.

The implications of this decentralized approach regarding common performance factors are:

- **Network Constraints**: D-SGD spreads communication across the network rather than concentrating it at a server. This can ease the bottlenecks caused by limited bandwidth at a central point. While communication still occurs and total volume can be significant (e.g., $\mathcal{O}(Nd)$ for sparse topologies), the system might be less sensitive to high latency between geographically distant nodes and a central server, as communication is more localized.

- **Model Size**: The volume of data exchanged in each node to node communication still scales linearly with the model dimension d. However, in D-SGD with sparse topologies, each node only needs to send and receive a small, constant number of these potentially large messages per iteration.

- **Scalability with N**: This is arguably the primary advantage of D-SGD. By employing network topologies where the node degree is constant with N (e.g., ring, grid), the per-node communication cost $\mathcal{O}(d)$ can remain independent of the total number of workers N[13]. This directly addresses the server bottleneck issue that limits the scalability of centralized SGD, where the server load increases with N. Consequently, D-SGD systems can scale to larger numbers of workers more effectively.

The central takeaway is, D-SGD's neighbor-to-neighbor communication strategy allows for the use of sparse network

[13] Note that fully connected does not share this benefit, having $\mathcal{O}(Nd)$ per-node cost.

topologies. In such topologies, the per-node communication cost depends primarily on model size d and node degree, but not on the total number of workers N. This enhances scalability with N. However, the trade-off between communication cost per iteration and the number of iterations required for convergence must be considered.

III. Experiments and Results

A. Setup

To evalueate the convergence and communication behavior of centralized SGD and D-SGD under different topologies, a series of experiments[14] on synthetic non i.i.d dataset are performed, so that there are dissimilarities across nodes, thus makes topology more observable. The experiments are performed on both a logistic regression problem (follows [16]), which is convex, and a quadratic regression problem, which is strongly convex.

- **Objective functions.** Data matrix $X \in \mathbb{R}^{B\times d}$, label $y = \{-1, 1\}^B$, weight matrix $W \in \mathbb{R}^d$, and L2 regularization parameter λ. μ for strong convexity parameter.
  - Logistic.
$$f(W) = \frac{1}{B}\sum_{i=0}^{B} \log(1 + \exp(-y_i \cdot x_i^T W)) + \frac{\lambda}{2}\|W\|^2$$
  - Quadratic.
$$f(W) = \frac{1}{2B}\sum_{i=0}^{B} (x_i^T W - y_i)^2 + \frac{\mu}{2}\|W\|^2$$
- **Algorithms.**
  - Centralized Mini-Batch SGD. A central server with 36 nodes.
  - D-SGD with ring topology.
  - D-SGD with a toroidal 5x5 grid topology.
  - D-SGD with a fully connected topology.
- **Network.** A network with $N = 25$ worker nodes is simulated. In this configuration, the spectral gap $1 - \rho$ is 0.0209 for ring, 0.2764 for grid, and 1.0 for fully connected.
- **Data.** The synthetic dataset is generated using `sklearn.datasets.make_regression` The total dataset size is $B = 12500$, with $d = 80$ features. The data is distributed among the workers in Non-IID by first sorting the entire dataset based on the target variable y, then partitioning it equally among the 25 workers. This resulted in significant heterogeneity, in quadratic problem, worker 0 holds data with mean y = -976.78 and worker 24 holds data where mean y = 961.87.
- **Hyperparameters.** All algorithms run for $T = 10000$ iterations. The initial learning rate is 0.05 at $\mathcal{O}(1/\sqrt(t)$ decay. The local mini-batch size per worker is $b = 16$. The L2 regularization parameter is 0.0001. The reference optimum was computed using `sklearn`'s ridge solver, and the suboptimality threshold for reporting iteration counts was set to $\epsilon \leq 0.08$.
- **Evaluation Metrics.** The following metrics are measured:
  - Suboptimality Gap: $f(\bar{x}_T) - f(x^*)$, this is evaluated on the full dataset.
  - Consensus Error: $\frac{1}{N}\sum_{i=0}^{N} \|x_{i,t} - \bar{x}_T\|$ measuring the disagreement between worker models in D-SGD.
  - Iterations to Threshold: The number of iterations required to reach a suboptimality gap.
  - Communication Cost: Total number of floating-point values transmitted across the network over T iterations.

B. Results

Convergence: As shown in the left plots of Figure 4 and Figure 5, all methods converge toward the optimal objective value, but the rates differ according to both the communication topology and the problem's curvature. For the convex logistic loss, the suboptimality gap decreases approximately at a rate of $\mathcal{O}(1/\sqrt(T))$ as indicated by its down-bending curvature in log-scale, while for the strongly convex quadratic loss, the convergence is faster—around $\mathcal{O}(1/T)$ as indicated by its less bending, almost linear curvature. This behavior aligns with the theoretical convergence bounds derived in previous sections.

Among the methods, Centralized SGD achieves the fastest convergence for the quadratic loss, reaching the target suboptimality in just 5425 iterations. In contrast, for logistic loss, the decentralized SGD with a fully-connected topology performs best, reaching a suboptimality gap below 0.08 within 9596 iterations. The faster convergence in the quadratic case reflects its stronger curvature properties. Across topologies, fully-connected networks consistently converge the fastest due to efficient information mixing, followed by the grid. The ring topology shows the slowest convergence, attributed to its limited connectivity and smaller spectral gap. Moreover, under strongly convex objectives, the ring topology exhibits substantially slower convergence compared to grid and fully connected topologies. This is primarily due to its minimal connectivity and small spectral gap, which hinder efficient information propagation across nodes. As a result, the consensus error persists longer, and the suboptimality gap decays more slowly.

Consensus Error: As shown in the right figures in Figure 4 and Figure 5, centralized SGD has zero consensus error by definition, fully-connected topology achieves the lowest consensus error as expected, showing tight synchronization between local models, with grid closely behind. In contrast,

[14] Source code is available at https://github.com/scavenx/distributed-optimization

ring topology exhibits significantly higher consensus error especially in quadratic regression which can slow convergence.

Communication: In I and II, centralized SGD has the lowest information transmission which is 4.05×10^7 floats, while fully-connected topology has the highest total data transmission with 4.86×10^8 floats. Here, ring topology offers the lowest communication overhead among D-SGD topologies.

Summary: In the simulated Non-IID convex problem, communication topology plays an important role in D-SGD performance. While the fully-connected topology achieves convergence rates comparable to centralized SGD, it has significantly higher communication cost. The ring topology is the most communication efficient per iteration but suffers from slow convergence due to limited information mixing, as reflected in its consensus error. The grid topology offers a reasonable balance between convergence rate and communication transmission in this heterogeneous setting.

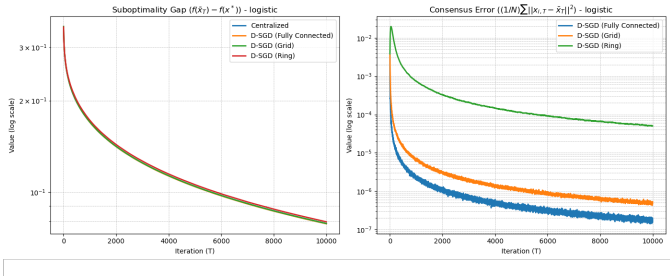

Fig. 4: Suboptimality Gap and Consensus Error of Convex Problem

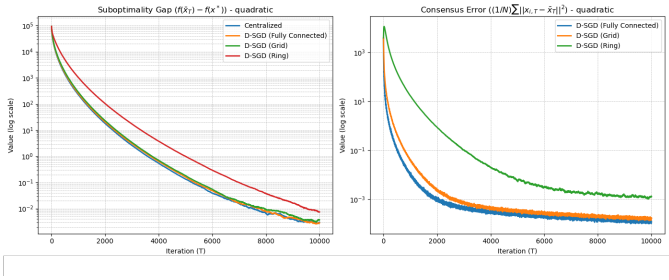

Fig. 5: Suboptimality Gap and Consensus Error of Strongly Convex Problem

TABLE I: Comparison of D-SGD and Centralized SGD on Convex Problem (Logistic Loss)

Topology	Iters to Converge	Total Transmitted
Centralized	9641	4.050×10^7
D-SGD (Ring)	9927	4.050×10^7
D-SGD (Grid)	9636	8.100×10^7
D-SGD (Fully Connected)	9596	4.860×10^8

TABLE II: Comparison of D-SGD and Centralized SGD on Strongly Convex Problem (Quadratic Loss)

Topology	Iters to Converge	Total Transmitted
Centralized	5425	4.050×10^7
D-SGD (Ring)	7214	4.050×10^7
D-SGD (Grid)	5666	8.100×10^7
D-SGD (Fully Connected)	5549	4.860×10^8

IV. Discussion

Based on the experiments, Decentralized SGD (D-SGD) clearly outperforms centralized SGD in convergence speed. Centralized SGD suffers from a single-point aggregation bottleneck—every node must wait for the parameter server to collect and redistribute gradients—which slows its convergence despite its relatively modest communication volume. In contrast, D-SGD's peer-to-peer exchanges enable much faster consensus: going from a ring to a fully connected topology, convergence accelerates further. Although the fully connected and grid topologies achieve nearly identical convergence rates, due to the large spectral gaps, which rapidly propagate information across the network. The fully connected graph incurs substantially much more data transfer than the grid. The ring topology, with the smallest spectral gap, converges more slowly, reflecting the trade-off between communication cost and convergence speed inherent to different network structures. In addition, the experiment reveals that the number of nodes significantly influences convergence rates. When the number of nodes is small, fully-connected and grid topologies do not offer substantial improvements—and in some cases, they may even perform worse due to communication overhead outweighing their consensus advantages.

These findings confirm the second term in the theoretical convergence bounds derived in Section Mixing Matrix. The impact of spectral gap is clearly visible: topologies with smaller ρ (larger spectral gap) require fewer iterations to converge. This emphasizes the importance of selecting an appropriate communication topology based on system constraints. For instance, in edge devices with limited bandwidth, ring or grid topologies may be preferred, whereas high-performance clusters may tolerate fully-connected communication for rapid convergence.

V. Conclusion

This report analyzed centralized and decentralized SGD methods under different network topologies (ring, grid, and fully-connected) across both convex problem and strongly convex problem. While fully connected and grid network promote rapid consensus and minimize iterations, the trade-off is higher communication cost. In contrast, ring topology are communication-efficient but converge slowly. The grid topology seems to be the balance between convergence rates and communication cost. The spectral gap of the mixing

matrix plays an important role in convergence rate, making it a key design consideration when scaling distributed optimization under bandwidth or topology constraints.

REFERENCES

[1] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," NeurIPS, 2012. https://papers.nips.cc/paper_files/paper/2012/file/6aca97005c68f1206823815f66102863-Paper.pdf

[2] T. Yang, Q. Ling, Z. Wen, M. Yin, and W. Yin, "A survey of distributed optimization," *Annual Reviews in Control*, vol. 47, pp. 278–305, Jan. 2019, doi: 10.1016/j.arcontrol.2019.05.006.

[3] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for Decentralized parallel Stochastic Gradient Descent," *arXiv preprint*, arXiv:1705.09056, May 2017, doi: 10.48550/arxiv.1705.09056.

[4] S. U. Stich, "Local SGD converges fast and communicates little," ICLR 2019, May 24, 2018. https://arxiv.org/abs/1805.09767

[5] O. Dekel et al., "Optimal distributed online prediction using Mini-Batches," Journal of Machine Learning Research, Jan. 2012. https://www.jmlr.org/papers/volume13/dekel12a/dekel12a.pdf

[6] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning," SIAM Review, vol. 60, no. 2, pp. 223–311, Jan. 2018, doi: 10.1137/16m1080173. https://arxiv.org/pdf/1606.04838

[7] G. Garrigos and R. M. Gower, "Handbook of Convergence Theorems for (Stochastic) Gradient Methods," arXiv, Jan. 26, 2023. https://arxiv.org/abs/2301.11235

[8] S. Ghadimi and G. Lan, "Stochastic first- and zeroth-order methods for nonconvex stochastic programming," SIAM '13, Sep. 22, 2013. https://arxiv.org/abs/1309.5549

[9] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust Stochastic Approximation Approach to Stochastic Programming," SIAM, 2009. https://www2.isye.gatech.edu/~nemirovs/SIOPT_RSA_2009.pdf

[10] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: convergence results and optimal averaging schemes," PMLR, Dec. 08, 2012. https://arxiv.org/abs/1212.1824

[11] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," ICML '12, Dec. 2, 2012. https://arxiv.org/abs/1109.5647

[12] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning," ACM Computing Surveys, vol. 52, no. 4, pp. 1–43, Aug. 2019, doi: 10.1145/3320060. https://arxiv.org/pdf/1802.09941

[13] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A Unified Theory of Decentralized SGD with Changing Topology and Local Updates." ICML '20. https://proceedings.mlr.press/v119/koloskova20a/koloskova20a.pdf

[14] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," Information Processing in Sensor Networks, p. 9, Apr. 2005, doi: 10.5555/1147685.1147698. https://lall.stanford.edu/papers/xiao_2005a_ipsn_sensors_2005_04_25_01/pubdata/entry.pdf

[15] T. Zhu, F. He, K. Chen, M. Song, and D. Tao, "Decentralized SGD and Average-direction SAM are Asymptotically Equivalent," PMLR '23, Jan. 2023. https://arxiv.org/abs/2306.02913

[16] H. Yu, R. Jin. "On the Computation and Communication Complexity of Parallel SGD with Dynamic Batch Sizes for Stochastic Non-Convex Optimization," 2019. ICML '19. https://proceedings.mlr.press/v97/yu19c/yu19c.pdf