Decentralized gradient methods: does topology matter?

Giovanni Neglia
Inria, Univ. Côte d'Azur
France

Chuan Xu
Inria, Univ. Côte d'Azur
France

Don Towsley
UMass Amherst
USA

Gianmarco Calbi
Inria, Univ. Côte d'Azur
France

Abstract

Consensus-based distributed optimization methods have recently been advocated as alternatives to parameter server and ring all-reduce paradigms for large scale training of machine learning models. In this case, each worker maintains a local estimate of the optimal parameter vector and iteratively updates it by averaging the estimates obtained from its neighbors, and applying a correction on the basis of its local dataset. While theoretical results suggest that worker communication topology should have strong impact on the number of epochs needed to converge, previous experiments have shown the opposite conclusion. This paper sheds lights on this apparent contradiction and show how sparse topologies can lead to faster convergence even in the absence of communication delays.

1 INTRODUCTION

In 2014, Google's Sybil machine learning (ML) platform was processing hundreds of terabytes through thousands of cores to train models with hundreds of billions of parameters (Canini et al., 2014). At this scale, no single machine can solve these problems in a timely manner, and, as time goes on, the need for efficient distributed solutions becomes even more urgent. For example, experiments in (Young et al., 2017) rely on more than 10^4 computing nodes to iteratively improve the (hyper)parameters of a deep neural network.

The example in (Young et al., 2017) is typical of a large class of iterative ML distributed algorithms. Such algorithms begin with a guess of an optimal vector of parameters and proceed through multiple iterations over the input data to improve the solution. The process evolves in a data-parallel manner: input data is divided among worker threads. Currently, two communication paradigms are commonly used to coordinate the different workers (Google I/O, 2018): parameter server and ring all-reduce. Both paradigms are natively supported by TensorFlow (Abadi et al., 2016).

In the first case, a stateful parameter server (PS) (Smola and Narayanamurthy, 2010) maintains the current version of the model parameters. Workers use locally available versions of the model to compute "delta" updates of the parameters (e.g., through a gradient descent step). Updates are then aggregated by the parameter server and combined with its current state to produce a new estimate of the optimal parameter vector.

As an alternative, it is possible to remove the PS, by letting each worker aggregate the inputs of all other workers through the ring all-reduce algorithm (Gibiansky, 2017). With M workers, each aggregation phase requires $2(M-1)$ communication steps with $\mathcal{O}(1)$ data transmitted per worker. There are many efficient low level implementations of ring all-reduce, e.g., in NVIDIA's library NCCL.

We observe that both the PS and the ring all-reduce paradigms 1) maintain a unique candidate parameter vector at any given time and 2) rely *logically* on an all-to-all communication scheme.[1] Recently Lian et al. (2017, 2018) have promoted an alternative approach in the ML research community, where each worker 1) keeps updating a local version of the parameters and 2) broadcasts its updates only to a subset of nodes (its neighbors). This family of algorithms became originally popular in the control community, starting from the seminal work of Tsitsiklis et al. (1986) on distributed gradient methods. They are often referred to as *consensus-based distributed optimization meth-*

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

[1]Each node needs to receive the aggregate of all other nodes' updates to move to the next iteration. Aggregation is performed by the PS or along the ring through multiple rounds.

ods. Experimental results in (Lian et al., 2017, 2018; Luo et al., 2019) show that

1. in terms of number of epochs, the convergence speed is almost the same when the communication topology is a ring or a clique, *contradicting* theoretical findings that predict convergence to be faster on a clique;
2. in terms of wall-clock time, convergence is faster for sparser topologies, an effect attributed in (Lian et al., 2017) to smaller communication load.

In particular, Luo et al. (2019) summarize their findings as follows "*in theory, the bigger the spectral gap,* [i.e., the more connected the topology] *the fewer iterations it takes to converge. However, our experiments do not show a significant difference in the convergence rate w.r.t. iterations, even when spectral gaps are very dissimilar.*"

In this paper we contribute to a better understanding of the potential advantages of consensus-based gradient methods. In particular,

1. we present a refined convergence analysis that helps to explain the apparent contradiction among theoretical results and empirical observations,
2. we show that sparse topologies can speed-up wall-clock time convergence even when communication costs are negligible, because they intrinsically mitigate the straggler problem.

The paper is organized as follows. Section 2 provides required background. Our theoretical analysis of the effect of communication topology is in Sect. 3. Experiment results in Sect. 4 confirm our findings. Section 5 concludes the paper.

2 NOTATION AND BACKGROUND

The goal of supervised learning is to learn a function that maps an input to an output using S examples from a training dataset $\mathbb{S} = \{(\boldsymbol{x}^{(l)}, y^{(l)}), l = 1, \dots S\}$. Each example $(\boldsymbol{x}^{(l)}, y^{(l)})$ is a pair consisting of an input object $\boldsymbol{x}^{(l)}$ and an associated target value $y^{(l)}$. In order to find the best statistical model, ML techniques often find the set of parameters $\boldsymbol{w} \in \mathbb{R}^n$ that solves the following optimization problem:

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \sum_{l=1}^{S} f(\boldsymbol{w}, \boldsymbol{x}^{(l)}, y^{(l)}) \tag{1}$$

where function $f(\boldsymbol{w}, \boldsymbol{x}^{(l)}, y^{(l)})$ represents the error the model commits on the l-th element of the dataset $\mathbb{S}$ when parameter vector $\boldsymbol{w}$ is used. The objective function may also include a regularization term that enforces some "simplicity" (e.g., sparseness) on $\boldsymbol{w}$; such a term is easily taken into account in our analysis.

Due to increases in available data and statistical model complexity, distributed solutions are often required to determine the parameter vector in a reasonable time. The dataset in this case is divided among M workers ($\mathbb{S} = \cup_{j=1}^{M} \mathbb{S}_j$), possibly with some overlap. For simplicity, we consider that all local datasets $\mathbb{S}_j$ have the same size. Problem (1) can be restated in an equivalent form as minimization of the sum of functions local to each node:

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad F(\boldsymbol{w}) = \sum_{j=1}^{M} F_j(\boldsymbol{w}), \tag{2}$$

where $F_j(\boldsymbol{w}) = \frac{1}{|\mathbb{S}_j|} \sum_{(\boldsymbol{x}^{(l)}, y^{(l)}) \in \mathbb{S}_j} f(\boldsymbol{w}, \boldsymbol{x}^{(l)}, y^{(l)})$.

The distributed system can be represented by a directed *dataflow graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \dots M\}$ is the set of nodes (the workers) and an edge $(i, j) \in \mathcal{E}$ indicates that, at each iteration, node j waits for updates from node i for the previous iteration. We assume the graph is strongly connected. Let $N_j = \{i | (i, j) \in \mathcal{E}\}$ denote the in-neighborhood of node j, i.e., the set of predecessors of node j in $\mathcal{G}$. Each node j maintains a local estimate of the parameter vector $\boldsymbol{w}_j(k)$ and broadcasts it to its successors. The local estimate is updated as follows:

$$\boldsymbol{w}_j(k+1) = \sum_{i \in N_j \cup \{j\}} \boldsymbol{w}_i(k) A_{i,j} - \eta(k) \boldsymbol{g}_j(\boldsymbol{w}_j(k)). \tag{3}$$

The node computes a weighted average (consensus/gossip component) of the estimates of its neighbors and itself, and then corrects it taking into account a stochastic subgradient $\boldsymbol{g}_j(\boldsymbol{w}_j(k))$ of its local function, i.e.,

$$\boldsymbol{g}_j(\boldsymbol{w}_j(k)) = \frac{1}{B} \sum_{(\boldsymbol{x}^{(l)}, y^{(l)}) \in \xi_j(k)} \partial f(\boldsymbol{w}_j(k), \boldsymbol{x}^{(l)}, y^{(l)}),$$

where $\partial f(\boldsymbol{w}, \boldsymbol{x}^{(l)}, y^{(l)})$ denotes a subgradient of f with respect to $\boldsymbol{w}$, and $\xi_j(k)$ is a random minibatch of size B drawn from $\mathbb{S}_j$. Parameter $\eta(k) > 0$ is the (potentially time-varying) learning rate. $\boldsymbol{A} = (A_{i,j})$ is an $M \times M$ matrix of non-negative weights. We call $\boldsymbol{A}$ the *consensus matrix*.[2]

The operation of a synchronous PS or ring all-reduce is captured by (3) when the underlying graph $\mathcal{G}$ is a clique, $\boldsymbol{A} = \mathbf{1}\mathbf{1}^\intercal/M$, where $\mathbf{1}$ is the $M \times 1$ vector consisting of all ones, and $\boldsymbol{w}_i(0) = \boldsymbol{w}_j(0), \forall i, j \in \mathcal{V}$.

[2]We are describing a synchronous DSM. The consistency model could be weaker, allowing node i to use older estimates from its neighbors (Li et al., 2014).

Under some standard technical conditions, (Nedić et al., 2018, Thm. 8) and (Duchi et al., 2012, Thm. 2) conclude, respectively for Distributed Subgradient Method (DSM) and for the Dual Averaging Distributed method, that the number of iterations K_ϵ needed to approximate the minimum objective function value by the desired error ϵ is

$$K_\epsilon \in \mathcal{O}\left(\frac{1}{\epsilon^2 \gamma(\boldsymbol{A})}\right), \tag{4}$$

where $0 \leq \gamma(\boldsymbol{A}) \leq 1$ is the spectral gap of the matrix $\boldsymbol{A}$, i.e., the difference between the moduli of the two largest eigenvalues of $\boldsymbol{A}$. The spectral gap quantifies how information flows in the network. In particular, the spectral gap is maximal for a clique with weights $A_{i,j} = 1/M$. Motivated by these convergence results, existing theoretically-oriented literature has concluded that a more connected network topology leads to faster convergence (Nedić et al., 2018; Duchi et al., 2012). But some recent experimental results report that consensus-based gradient methods achieve similar performance after the same number of iterations/epochs on topologies as different as rings and cliques. For example (Lian et al., 2017, Fig. 3) shows almost overlapping training losses for different ResNet architectures trained on CIFAR-10 with up to one hundred workers. (Luo et al., 2019, Fig. 20), (Koloskova et al., 2019, Fig. 11 in supplementary material), and our experimental results in Sect. 4 confirm these findings.

Lian et al. (2017) provide a partial explanation for this insensitivity in their Corollary 2, showing that the convergence rate is topology-independent 1) after a large number of iterations ($\mathcal{O}(M^5/\gamma(\boldsymbol{A})^2)$), 2) for a vanishing learning rate, and 3) when the functions F_j are differentiable with Lipschitzian gradients. Under the additional hypothesis of strong convexity, Pu et al. (2019) prove that topology insensitivity should manifest after $\mathcal{O}(M/\gamma(\boldsymbol{A})^2)$ iterations.[3] Assran et al. (2019) provide similar results in terms of the graph diameter and maximum degree. These results do not explain why insensitivity is often observed in practice (as shown in (Lian et al., 2017, 2018; Luo et al., 2019; Koloskova et al., 2019)) 1) since the beginning of the training phase, 2) with constant learning rates, and 3) for non-differentiable machine learning models (e.g., neural networks). In the following section, we present a refined convergence analysis that explains when and why the effect of topology on the number of iterations needed to converge is weaker than what previously predicted.

[3]We provide numerical estimates for the number of iterations predicted by (Lian et al., 2017) and (Pu et al., 2019) in Appendix C.

3 ANALYSIS

A less connected topology requires more iterations to achieve a given precision as indicated by (4). Our detailed analysis below shows that, when consensus-based optimization methods are used for ML training, the increase in the number of iterations is much less pronounced than previous studies predict. This is due to two different effects. First, consensus is affected only by variability in initial estimates and subgradients across nodes, and not by their absolute values. Second, certain configurations of initial estimates and subgradients are more difficult to achieve a consensus over, and would make the training highly dependent on the topology, but they are unlikely to be obtained by randomly partitioning the dataset.

Let n be the number of parameters of the model, and $\boldsymbol{W}(k)$ and $\boldsymbol{G}(k)$ be $n \times M$ matrices, whose columns are, respectively, node estimates $\boldsymbol{w}_1(k), \ldots, \boldsymbol{w}_M(k)$ and subgradients $\boldsymbol{g}_1(\boldsymbol{w}_1(k)), \ldots, \boldsymbol{g}_M(\boldsymbol{w}_M(k))$ at the completion of iteration k. Equation (3) can be rewritten in the form $\boldsymbol{W}(k+1) = \boldsymbol{W}(k)\boldsymbol{A} - \eta(k)\boldsymbol{G}(k)$, from which we obtain iteratively

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(0)\boldsymbol{A}^{k+1} - \sum_{h=0}^{k} \eta(h)\boldsymbol{G}(h)\boldsymbol{A}^{k-h}. \tag{5}$$

We make the following assumptions:[4]

A1 all functions F_i are convex,

A2 the set of (global) minimizers $\mathbb{W}^*$ is non-empty,

A3 graph $\mathcal{G}$ is strongly connected,

A4 matrix $\boldsymbol{A}$ is normal (i.e., $\boldsymbol{A}^\intercal\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^\intercal$) and doubly stochastic,

A5 the squared Frobenius norm of subgradient matrix $\boldsymbol{G}(k)$ is bounded in expectation over the vector $\boldsymbol{\xi}$ of minibatches randomly drawn at nodes, i.e., there exists E, such that $\mathbb{E}_{\boldsymbol{\xi}}\left[\|\boldsymbol{G}(k)\|_F^2\right] \leq E$.

Assumptions A1-A4 are standard ones in the related literature, see for example (Nedić and Ozdaglar, 2009; Duchi et al., 2012; Nedić et al., 2018). Assumption A5 imposes a bound on the (expected) *energy* of the subgradients, because $\|\boldsymbol{G}(k)\|_F^2 = \sum_j \|\boldsymbol{g}_j(\boldsymbol{w}_j(k))\|_2^2$. In the literature it is often replaced by the stronger requirement that the norm-2 of the subgradients $\boldsymbol{g}_j(\boldsymbol{w}_j(k))$ is bounded. Let $\Delta\boldsymbol{G}(k)$ denote the matrix $\boldsymbol{G}(k) - \boldsymbol{G}(k)\mathbf{1}\mathbf{1}^\intercal/M$, whose column j is the difference between subgradient $\boldsymbol{g}_j(\boldsymbol{w}_j(k))$ and the average

[4]Experiments in Sect. 4 show that our conclusions hold also when these assumptions are not satisfied.

of subgradients $\sum_{j=1}^{M} \boldsymbol{g}_j(\boldsymbol{w}_j(k))/M$. $\|\Delta \boldsymbol{G}(k)\|_F^2$ captures the variability in the subgradients. Assumption A5 also implies that there exist two constants $E_{\rm sp} \leq E$ and $H \leq \sqrt{E}$ such that

$$\mathbb{E}_{\boldsymbol{\xi}}\left[\|\Delta \boldsymbol{G}(k)\|_F^2\right] \leq E_{\rm sp}, \quad \|\mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{G}(k)]\|_F \leq H.$$

Similarly, let R denote the energy of the initial parameter vectors (or an upper bound for it), i.e., $R \triangleq \|\boldsymbol{W}(0)\|_F^2$. We also denote by $R_{\rm sp}$ the energy for the difference matrix $\Delta \boldsymbol{W}(0) \triangleq \boldsymbol{W}(0) - \boldsymbol{W}(0)\boldsymbol{1}\boldsymbol{1}^\intercal/M$, i.e., $R_{\rm sp} \triangleq \|\Delta \boldsymbol{W}(0)\|_F^2$. $R_{\rm sp}$ captures the variability in initial estimates. It holds $R_{\rm sp} \leq R$.

Because of Assumption A4, the consensus matrix has a spectral decomposition with orthogonal projectors $\boldsymbol{A} = \sum_{q=1}^{Q} \lambda_q \boldsymbol{P}_q$, where $\lambda_1, \ldots, \lambda_Q$ are the $Q \leq M$ distinct eigenvalues of $\boldsymbol{A}$, $\boldsymbol{P}_q$ is the orthogonal projector onto the nullspace of $\boldsymbol{A} - \lambda_q \boldsymbol{I}$ along the range of $\boldsymbol{A} - \lambda_q \boldsymbol{I}$. We assume that the eigenvalues are ordered so that $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_Q|$. Assumptions A3 and A4 imply that $\lambda_1 = 1$, and $|\lambda_2| < 1$ (Appendix B). Finally, we define

$$\alpha \triangleq \begin{cases} \sqrt{\sum_{q=2}^{Q} e_q \left|\frac{\lambda_q}{\lambda_2}\right|^2}, & \text{if } \lambda_2 \neq 0, \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

where e_q is an upper-bound for the normalized fraction of energy $\mathbb{E}_{\boldsymbol{\xi}}\left[\|\Delta \boldsymbol{G}(k)\|_F^2\right]$ in the subspace defined by projector $\boldsymbol{P}_q$ (Appendix D). The quantity α can be interpreted as an effective bound for the fraction of the energy $E_{\rm sp}$ that falls in the subspace relative to the second largest eigenvalue λ_2.

We are now ready to introduce our main convergence result. We state it for the average model over nodes and time, i.e., for $\hat{\overline{\boldsymbol{w}}}(K-1) \triangleq \frac{1}{K}\sum_{k=0}^{K-1}\frac{1}{M}\sum_{i=1}^{M}\boldsymbol{w}_i(k)$. We have also derived a similar bound for the local time-average model at each node, i.e., for $\hat{\boldsymbol{w}}_i(K-1) \triangleq \frac{1}{K}\sum_{k=0}^{K-1}\boldsymbol{w}_i(k)$ (Appendix D.3).

Proposition 3.1. *Under assumptions A1-A5 and that a constant learning rate $\eta(k) = \eta$ is used, an upper bound for the objective value at the end of the $(K-1)$th iteration is given by:*

$$\begin{aligned} \mathbb{E}\left[F\left(\hat{\overline{\boldsymbol{w}}}(K-1)\right)\right] - F^* \leq{}& \frac{M}{2\eta K}\mathrm{dist}\left(\hat{\overline{\boldsymbol{w}}}(0), \mathbb{W}^*\right)^2 + \frac{\eta E}{2} \\ &+ 2H\sqrt{R_{sp}}\frac{\sqrt{M}}{K}\frac{1-|\lambda_2|^K}{1-|\lambda_2|} \\ &+ 2\eta H\sqrt{E_{sp}}\left((1-\alpha)\frac{K-1}{K}\right. \\ &\left. + \frac{\alpha}{1-|\lambda_2|}\left(1 - \frac{1}{K}\frac{1-|\lambda_2|^K}{1-|\lambda_2|}\right)\right). \end{aligned} \tag{7}$$

Here, $\mathrm{dist}(\boldsymbol{x}, \mathbb{W}^*)$ denotes the Euclidean distance between vector $\boldsymbol{x}$ and set of global minimizers $\mathbb{W}^*$. The proof is in Appendix D.1. The first two terms on the right hand side of (7) also appear when studying the convergence of centralized subgradient methods. The last two terms appear because of the distributed consensus component of the algorithm and depend on $|\lambda_2| < 1$. We observe that $1 - |\lambda_2|$ is the spectral gap of $\boldsymbol{A}$. It measures how well connected the graph is. In particular, the larger the spectral gap (the smaller λ_2), the better the connectivity and the smaller the bound in (7).

From Proposition 3.1, we can derive a looser bound analogous to the bound for DSM in (Nedić and Ozdaglar, 2009). In fact, observing that $R_{\rm sp} \leq R$, $E_{\rm sp} \leq E$, $H \leq \sqrt{E}$, and $0 \leq \alpha \leq 1$, we can prove (Appendix D.2):

Corollary 3.2. *Under assumptions A1-A5 and that constant learning rate $\eta(k) = \eta$ is used, an upper bound for the objective value at the end of the $(K-1)$th iteration is given by:*

$$\begin{aligned} \mathbb{E}\left[F\left(\hat{\overline{\boldsymbol{w}}}(K-1)\right)\right] - F^* \leq{}& \frac{M}{2\eta K}\mathrm{dist}\left(\hat{\overline{\boldsymbol{w}}}(0), \mathbb{W}^*\right)^2 + \frac{\eta E}{2} \\ &+ 2\sqrt{E}\sqrt{R}\frac{\sqrt{M}}{K}\frac{1-|\lambda_2|^K}{1-|\lambda_2|} \\ &+ 2\eta E\frac{1}{1-|\lambda_2|}\left(1 - \frac{1}{K}\frac{1-|\lambda_2|^K}{1-|\lambda_2|}\right). \end{aligned} \tag{8}$$

In particular, if workers compute full-batch subgradients and the 2-norm of subgradients of functions F_i is bounded by a constant L, we obtain:

$$\begin{aligned} F\left(\hat{\overline{\boldsymbol{w}}}(K-1)\right) - F^* \leq{}& \frac{M}{2\eta K}\mathrm{dist}\left(\hat{\overline{\boldsymbol{w}}}(0), \mathbb{W}^*\right)^2 + \frac{\eta M L^2}{2} \\ &+ 2L\sqrt{R}\frac{M}{K}\frac{1-|\lambda_2|^K}{1-|\lambda_2|} \\ &+ 2\eta L^2\frac{M}{1-|\lambda_2|}\left(1 - \frac{1}{K}\frac{1-|\lambda_2|^K}{1-|\lambda_2|}\right). \end{aligned} \tag{9}$$

When K is large enough, the fourth term in (8) and (9) is dominant, so that the error is essentially proportional to $1/(1-|\lambda_2|)$. Note that the constant multiplying $1/(1-|\lambda_2|)$ in (8) is larger than the corresponding one in (7) by a factor

$$\beta \triangleq \frac{1}{\alpha} \times \frac{E}{\sqrt{E_{\rm sp}}H} \tag{10}$$

The value β roughly indicates how much looser bound (8) is in comparison to bound (7).

Existing theoretical works like (Nedić and Ozdaglar, 2009; Duchi et al., 2012; Nedić et al., 2018) derived bounds similar to (9) and concluded then that

one should select the learning rate proportional to $\sqrt{1-|\lambda_2|}$ to reduce the effect of topology. In particular, one obtains (4) when $\eta = \eta_0\sqrt{(1-|\lambda_2|)/K}$. Our bound (7) improves bound (8) by replacing R in the third terms of (8) by the smaller value R_{sp}, and $\sqrt{E}$ in the third and fourth terms by the smaller values H and $\sqrt{E_{\text{sp}}}$, and introducing the new coefficient α. We qualitatively describe the effect of these constants.

$\mathbf{R_{sp}}$ Bound (7) shows that the norm of the initial estimates (R) does not really matter, but rather variability among workers does. In particular, for ML computation we can make $\boldsymbol{w}_i(0) = \boldsymbol{w}_j(0)$ for each i and j, and then $R_{\text{sp}} = 0$, so that the third term in the RHS of (7) vanishes.

$\mathbf{E_{sp}}$, $\mathbf{H}$ For E_{sp}, considerations similar to those applying to R_{sp} hold. What matters is the variability of the subgradients. Assume that the dataset is replicated at each node and each node computes the subgradient over the full batch ($B = S$). In this case all subgradients would be equal, and $\|\Delta \boldsymbol{G}(k)\| = 0$, $E_{\text{sp}} = 0$, and the fourth term would also vanish. This corresponds to the fact that, when initial parameter vectors, as well as local functions, are the same, the parameter vectors are equal at any iteration k and the system evolves exactly as it would under a centralized subgradient method. In general, local subgradients can be expected to be close (and $E \gg E_{\text{sp}}$), if 1) local datasets are representative of the entire dataset (the dataset has been randomly split and $|\mathbb{S}_j| \gg M$), and 2) large batch sizes are used. On the other hand, when batch sizes are very small, one expects stochastic subgradients to be very noisy, and as a consequence the energy of the matrix $\boldsymbol{G}$ to be much larger than the energy of $\mathbb{E}_\xi[\boldsymbol{G}]$, so that $\sqrt{E} \gg H$. In both cases, $E/(\sqrt{E_{\text{sp}}}H)$ is large (in the first case because $\sqrt{E} \gg \sqrt{E_{\text{sp}}}$, and in the second because $\sqrt{E} \gg H$). We quantify these effects below.

$\boldsymbol{\alpha}$ From (5) we see that the effect of the subgradients is modulated by $\boldsymbol{A}^{k-h}$, that equals $\sum_{q=1}^{Q} \lambda_q^{k-h}\boldsymbol{P}_q$. The energy of the subgradients is spread across the different subspaces defined by the eigenvectors of $\boldsymbol{A}$. The classic bound (8) implicitly assumes that all energy falls in the subspace corresponding to λ_2 (this occurs if the row of the matrices $\boldsymbol{G}(k)$ are aligned with the second eigenvector). In reality, on average each subspace will only get $1/Q$-th of the total energy ($e_q \approx 1/Q$), and the energy in other subspaces will be dissipated faster than what happens for the subspace corresponding to λ_2. $\alpha \leq 1$ quantifies this effect.

A toy example in Appendix G illustrates qualitatively these effects. Here we provide estimates for the expected values of E, E_{sp}, and H over all possible ways to distribute the dataset $\mathbb{S}$ randomly across the nodes. We reason as follows. For a given parameter vector $\boldsymbol{w}$, consider the set of subgradients at all dataset points, i.e., $\cup_{(\boldsymbol{x}^{(l)},y^{(l)})\in\mathbb{S}}\{\partial f(\boldsymbol{w},\boldsymbol{x}^{(l)},y^{(l)})\}$. The average subgradient over all datapoints is $\partial F(\boldsymbol{w})$. Let $\sigma^2(\boldsymbol{w})$ denote the trace of the covariance matrix of all subgradients. We denote by $\mathbb{S}_C$ the expanded dataset where each datapoint is replicated C times with $1 \leq C \leq M$. The dataset $\mathbb{S}_C$ is split across the M nodes. Each node selects a random minibatch from its local dataset and we denote by $\boldsymbol{G}$ the corresponding subgradient matrix.

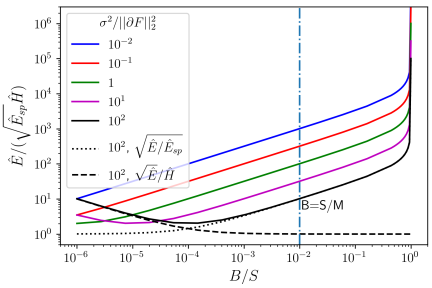


Figure 1: Estimate of $E/(\sqrt{E_{\text{sp}}}H)$ versus the relative batch size B/S for $M = 100$, $S = 10^6$, $C = M$, and different level of heterogeneity ($\sigma^2/\|\partial F\|_2^2$) of the dataset. Curves for $C = 1$ (in Appendix F) are very similar, but for the fact that the batch size can scale only up to S/M.

Proposition 3.3. *Consider a uniform random permutation π of $\mathbb{S}_C$ with the constraint that C copies of the same point are placed at C different nodes. The following holds*

$$\begin{aligned}
&\mathbb{E}_\pi\left[\mathbb{E}_{\boldsymbol{\xi}}\left[\|\boldsymbol{G}\|_F^2\right]\right] = M\left(\|\partial F\|_2^2 + \frac{S-B}{B(S-1)}\sigma^2\right),\\
&\mathbb{E}_\pi\left[\mathbb{E}_{\boldsymbol{\xi}}\left[\|\Delta\boldsymbol{G}\|_F^2\right]\right] = \sigma^2\frac{MC(S-B)-CS+MB}{CB(S-1)},\\
&\mathbb{E}_\pi\left[\|\mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{G}]\|_F\right] \qquad (11)\\
&\quad\in\left[\sqrt{M}\|\partial F\|_2, \sqrt{M}\sqrt{\|\partial F\|_2^2 + \frac{M-C}{C(S-1)}\sigma^2}\right].
\end{aligned}$$

We can use (11) to study how E, E_{sp}, and H vary with dataset size, batch size, and number of replicas, using the following approximations:

$$\begin{aligned}
&\widehat{E} = \mathbb{E}_\pi\left[\mathbb{E}_{\boldsymbol{\xi}}\left[\|\boldsymbol{G}\|_F^2\right]\right], \widehat{E}_{\text{sp}} = \mathbb{E}_\pi\left[\mathbb{E}_{\boldsymbol{\xi}}\left[\|\Delta\boldsymbol{G}\|_F^2\right]\right],\\
&\widehat{H} = \sqrt{M}\sqrt{\|\partial F\|_2^2 + \frac{M-C}{C(S-1)}\sigma^2}. \qquad (12)
\end{aligned}$$

Figure 1 illustrates the ratio $\widehat{E}/(\sqrt{\widehat{E}_{\text{sp}}}\widehat{H})(=\beta\alpha)$ for a particular setting. It also highlights the two regimes discussed above: $\beta \approx 1/\alpha \times \sqrt{E/E_{\text{sp}}}$ for large batch

sizes and $\beta \approx 1/\alpha \times \sqrt{E}/H$ for small ones. As β indicates how much looser bound (8) is in comparison to bound (7), and $\beta > E/(\sqrt{E_{\text{sp}}}H)$, the figure shows that (8) may indeed overestimate the effect of the topology by many orders of magnitudes.

4 EXPERIMENTS

With our experiments we want to 1) evaluate the effect of topology on the number of epochs to converge, and in particular quantify E, E_{sp}, H, and α in practical ML problems, 2) evaluate the effect of topology on the convergence *time*. We considered three different optimization problems:

1. Minimization of mean squared error (MSE) for linear regression on the dataset "Relative location of CT slices on axial axis" from (uci; Graf et al., 2011). Convexity holds, but gradients are potentially unbounded.

2. Minimization of cross-entropy loss through a neural network with two convolutional layers on MNIST dataset (Lecun et al., 1998). Neither convexity, nor subgradient boundness hold.

3. Minimization of cross-entropy loss through ResNet18 neural network (He et al., 2016) on CIFAR-10 dataset (Krizhevsky, 2009). Neither convexity, nor subgradient boundness hold. Moreover, we employ local subgradients with classical momentum (Sutskever et al., 2013) (with coefficient 0.9).

We have developed an ad-hoc Python simulator that allows us to test clusters with a large number of nodes, as well as a distributed application using PyTorch MPI backend to run experiments on a real GPU cluster platform. In general, datasets have been randomly split across the different workers without any replication ($C = 1$). For MNIST we have also considered a scenario with $M = 10$ workers, where each worker has been assigned all images for a specific digit. A constant learning rate has been set using the configuration rule from (Smith, 2017) described in Appendix H. Interestingly, for a given ML problem, when the dataset is split randomly, this procedure has led to choose the learning rate independently of the average node degree. The values selected are indicated in Table 1. Each node starts from the same model parameters ($R_{\text{sp}} = 0$) that have been initialized through PyTorch default functions. We report here a subset of all results, the others can be found in Appendix H.

Table 1 shows values of $\sqrt{E/E_{\text{sp}}}$, $\sqrt{E}/H$, $1/\alpha$, and their product β for different problems and different settings.[5] E, E_{sp}, and H have been evaluated through empirical averages using the random mini-batches drawn at the first iteration. α is computed for an undirected ring topology. Remember that the value β (defined in (10)) indicates how much tighter the new bound (7) is in comparison to the classic one (8). We also use (12) to provide an estimate of β as follows $\widehat{\beta} = 1/\alpha \times \widehat{E}/(\sqrt{\widehat{E}_{\text{sp}}}\widehat{H})$. The approximation is very accurate when the dataset is split randomly across the nodes. On the other hand, for MNIST, when all images for a given digit are assigned to the same node, local datasets are very different and approximations (12) are too crude (but our bound (7) still holds). Interestingly, β is dominated by different effects for the three ML problems. The similarity of local datasets prevails for CT (large $\sqrt{E/E_{\text{sp}}}$), while the noise of stochastic subgradients prevails for CIFAR (large $\sqrt{E}/H$). For MNIST the three effects, including energy spreading over different eigenspaces ($1/\alpha$), contribute almost equally. This can be explained considering that, even if local datasets have similar sizes, they are statistically more different the more complex the model to train, i.e., the larger n.

From (7) and (8), we can also compute at which iteration the two bounds predict that the effect of the topology becomes significant, by identifying when the training loss difference between the clique and the ring accounts for a given percentage of the loss decrease over the entire training period. Figure 3 qualitatively illustrates the procedure.[6] These predictions are indicated in the last columns of Table 1 and are compared with the values observed in the experiments (k').

We note that forecasts are very different, despite the fact that, in some settings, our bound is only 3 times tighter than the classic one. Bound (8) predicts that the training loss curves should differ by more than 10% since the first iteration ($k'_o = 1$). The new bound (7) correctly identifies that the topology's effect becomes evident later, sometimes beyond the total number of iterations performed in the experiment (in this case we indicate $k'_n = \infty$).

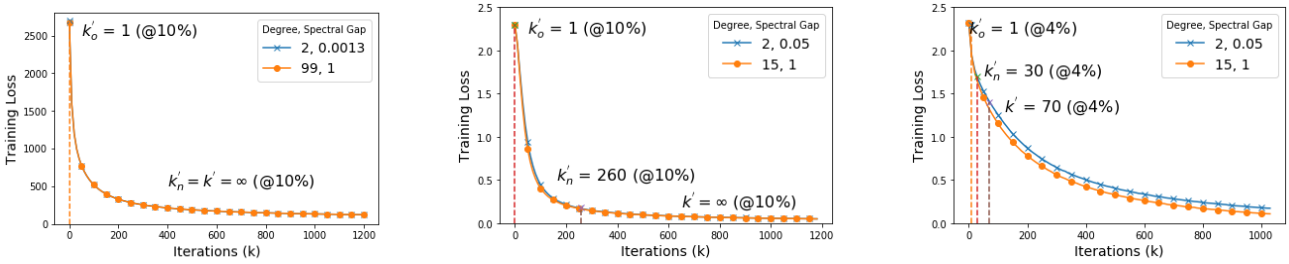
Figure 2 shows the training loss evolution $F(\hat{\boldsymbol{w}}(k))$ for specific settings (one for each ML problem) and two

[5] Some additional experiments in Appendix H show that R_{sp} and R have a smaller effect on the bounds, as the third term in (7) and in (8) converges to 0 when K diverges.

[6] In order to be able to compare the upper bounds (8) and (7) with the actual loss curves, we rescale them by a factor determined so that the upper bound curve and the experimental one are tangent for the clique topology (Fig. 3). Moreover, once determined at which iteration rings and cliques should differ, we update the upper-bounds with new estimates for E, E_{sp}, H, R, and R_{sp} computed at this iteration, and check if they now predict a larger number of iterations.

Table 1: Empirical estimation of E, E_{sp}, H, α on different ML problems and comparison of their joint effect (β) with the value $\widehat{\beta}$ predicted through (12). Number of iterations by which training losses for the ring and the clique differ by 4%, 10%, as predicted by the old bound (8), k'_o, by the new one (7), k'_n, and as measured in the experiment, k'. When a value exceeds the total number of iterations we ran (respectively 1200 for CT, 1190 for MNIST, and 1040 for CIFAR-10), we simply indicate it as ∞.

Dataset	Model	M	B	η	$\sqrt{E/E_{sp}}$	$\sqrt{E}/H$	$\frac{1}{\alpha}$	β	$\widehat{\beta}$	@4%			@10%		
										k'_o	k'_n	k'	k'_o	k'_n	k'
CT (S=52000)	Linear regr. n=384	16	128	0.0003	7.92	1.01	1.53	12.23	12.31	1	∞	∞	1	∞	∞
			3250		38.45	1.00	1.64	62.86	60.97	1	∞	∞	1	∞	∞
		100	128		7.75	1.01	1.54	12.05	11.56	1	10	∞	1	∞	∞
			520		15.58	1.00	1.51	23.60	22.96	1	17	∞	1	∞	∞
MNIST (S=60000)	2-conv layers n=431080	16	128	0.1	1.45	1.42	1.49	3.07	2.92	1	16	∞	1	72	∞
			500		2.15	1.14	1.53	3.75	3.71	1	22	40	1	260	∞
		64	128		1.41	1.42	1.51	3.02	3.03	1	10	∞	1	24	∞
split by digit		10	500	0.01	1.01	1.00	1.42	1.42	3.62	1	3	60	1	7	100
CIFAR-10 (S= 50000)	ResNet18 n=11173962	16	128	0.05	1.07	3.35	1.49	5.34	5.62	1	10	30	1	20	∞
			500		1.18	1.91	1.50	3.40	3.52	1	21	∞	1	250	∞



(a) CT: M=100, B = 128 (b) MNIST: M=16, B = 500 (c) CIFAR-10: M=16, B = 500

Figure 2: Effect of network connectivity (degree d) on the iterations to convergence.

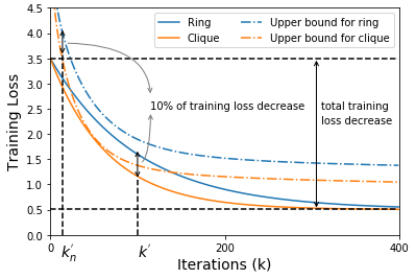


Figure 3: How to determine the number of iteration at which training loss for the clique and for the ring differs significantly.

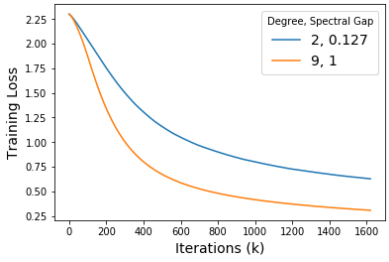
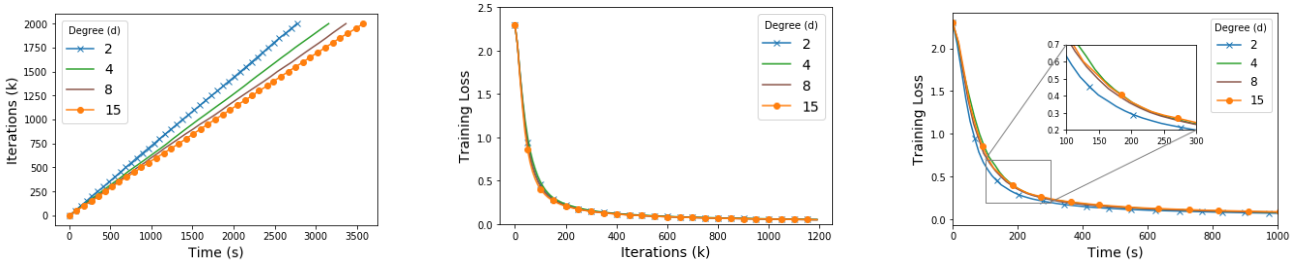


Figure 4: MNIST split by digit, M=10, B=500.

very different topologies (undirected ring and clique), when the dataset is split randomly across the nodes. The behaviour is qualitatively similar to what observed in previous works (Lian et al., 2017, 2018; Luo et al., 2019); despite the remarkable difference in the level of connectivity (quantified also by the spectral gap), the curves are very close, sometimes indistinguishable.

Figure 4 shows the same plot for the case when MNIST images for the same digit have been assigned to the same node. In this case the local datasets are very different and $\sqrt{E/E_{\text{sp}}} \approx 1$; the topology has a remarkable effect! This plot warns against extending the empirical finding in (Lian et al., 2017, 2018; Luo et al., 2019) to settings where local datasets can be highly different as it can be for example in the case of federated learning (Konecný et al., 2015).

The experiments above confirm that the communication topology has little influence on the number of



(a) Throughput (b) Error vs iterations (c) Error vs time

Figure 5: Effect of network connectivity (degree d) on the convergence for dataset MNIST with computation times from a Spark cluster. M = 16, B = 500.

epochs needed to converge (when local datasets are statistically similar). Our analysis reconciles (at least in part) theory and experiments by pushing farther the training epoch at which the effect of the topology should be evident.

The conclusion about the role of the topology is radically different if one considers the *time* to converge. For example, Karakus et al. (2017) and Luo et al. (2019) observe experimentally that sparse topologies can effectively reduce the convergence wall-clock time. A possible explanation is that each iteration is faster because less time is spent in the communication phase: the less connected the graph, the smaller the communication load at each node. Lian et al. (2017, 2018) advance this explanation to justify why DSM on ring-like topologies can converge faster than the centralized PS.

Here, we show that sparse topologies can speed-up wall-clock time convergence even when communication costs are negligible, because they intrinsically mitigate the effect of stragglers, i.e., tasks whose completion time can be occasionally much longer than its typical value. Transient slowdowns are common in computing systems (especially in shared ones) and have many causes, such as resource contention, background OS activities, garbage collection, and (for ML tasks) stopping criteria calculations. Stragglers can significantly reduce computation speed in a multi-machine setting (Ananthanarayanan et al., 2013; Karakus et al., 2017; Li et al., 2018). For consensus-based method, one can hope that, when the topology is sparse, a temporary straggler only slows down a limited number of nodes (its out-neighbors in $\mathcal{G}$), so that the system can still maintain a high throughput.

Neglia et al. (2019) have proposed approximate formulas to evaluate the throughput of distributed ML systems for some specific random distribution of the computation time (uniform, exponential, and Pareto). Here, we take a more practical approach. Our PyTorch-based distributed application allow us to simulate systems with arbitrary distributions of the computation times and communication delays. We have carried out experiments with zero communication delays (an ideal network) and two different empirical distributions for the computation time. One was obtained by running stochastic gradient descent on a production Spark cluster with sixteen servers using Zoe Analytics (Pace et al., 2017), each with two 8-core Intel E5-2630 CPUs running at 2.40GHz. The other was extracted from ASCI-Q super-computer traces (Petrini et al., 2003, Fig. 4). Figure 5 shows the effect of topology connectivity on the convergence time for a MNIST experiment with Spark computation distribution. We consider in this case undirected d-regular random graphs. The number of iterations completed per node grows faster the less connected the topology (Fig. 5 (a)). As the training loss is almost independent of the topology (Fig. 5 (b)), the ring achieves the shortest convergence time (Fig. 5 (c)), even if there is no communication delay. Qualitatively similar results for other ML problems and time distributions are in Appendix H.

5 CONCLUSIONS

We have explained, both through analysis and experiments, when and why the communication topology does not affect the number of epochs consensus-based optimization methods need to converge, an effect recently observed in many papers, but not thoroughly investigated. We have also shown that, as a consequence of this invariance, a less connected topology achieves a shorter convergence time, not necessarily because it incurs a smaller communication load, but because it mitigates the stragglers' problem. The distributed operation of consensus-based approaches appears particularly suited for federated and multi-agent learning. Our study points out that further research is required for these applications, because the benefits observed until now are dependent on the statistical similarity of the local datasets, an assumption that is not satisfied in federated learning.

6 ACKNOWLEDGEMENTS

We warmly thank Alain Jean-Marie, Pietro Michiardi, and Bruno Ribeiro for their feedback on the manuscript and their help preparing the rebuttal letter. We are also grateful to the OPAL infrastructure from Université Côte d'Azur and Inria Sophia Antipolis - Méditerranée "NEF" computation platform for providing resources and support. Finally, We would like to thank the anonymous reviewers for their insightful comments and suggestions, which helped us improve this work.

This work was supported in part by ARL under Cooperative Agreement W911NF-17-2-0196.

References

UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/.

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, 2016. ISBN 978-1-931971-33-1.

Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, and Ion Stoica. Effective straggler mitigation: Attack of the clones. In *Proc. of the 10th USENIX Conf. NSDI*, 2013.

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Michael Rabbat. Stochastic gradient push for distributed deep learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 344–353. PMLR, 2019.

Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014.

Kevin Canini, Tushar Chandra, Eugene Ie, Jim McFadden, Ken Goldman, Mike Gunter, Jeremiah Harmsen, Kristen LeFevre, Dmitry Lepikhin, Tomas Lloret Llinares, Indraneel Mukherjee, Fernando Pereira, Josh Redstone, Tal Shaked, and Yoram Singer. Sibyl: A system for large scale supervised machine learning, 2014. Technical talk.

John C. Duchi, Alekh Agarwal, and Martin J. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Trans. on Automatic Control*, 57(3):592–606, 2012.

Andrew Gibiansky. Bringing hpc techniques to deep learning. online, http://research.baidu.com/bringing-hpc-techniques-deep-learning, 2017.

Google I/O. Distributed tensorflow training. online, https://www.youtube.com/watch?v=bRMGoPqsn20, 2018.

Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2D Image Registration in CT Images Using Radial Image Descriptors. In *Proc. of MICCAI*, pages 607–614, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Can Karakus, Yifan Sun, Suhas Diggavi, and Wotao Yin. Straggler mitigation in distributed optimization through data encoding. In *Proc. of NIPS*, pages 5434–5442. 2017.

Anastasia Koloskova, Sebastian U. Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3478–3487. PMLR, 2019.

Jakub Konecný, Brendan McMahan, and Daniel Ramage. Federated Optimization: Distributed Optimization Beyond the Datacenter. In *Neural Information Processing Systems (workshop)*, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 19–27. Curran Associates, Inc., 2014.

Songze Li, Seyed Mohammadreza Mousavi Kalan, A. Salman Avestimehr, and Mahdi Soltanolkotabi. Near-Optimal Straggler Mitigation for Distributed Gradient Methods. In *Proc. of the 7th Intl. Workshop ParLearning*, May 2018.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can Decentralized Algo-

rithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *NIPS*, 2017.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *ICML*, 2018.

Qinyi Luo, Jinkun Lin, Youwei Zhuo, and Xuehai Qian. Hop: Heterogeneity-aware decentralized training. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, pages 893–907, 2019.

Brendan D. McKay. The expected eigenvalue distribution of a large regular graph. *Linear Algebra and its Applications*, 40:203 – 216, 1981.

Brendan D. McKay and Nicholas C. Wormald. Uniform generation of random regular graphs of moderate degree. *J. Algorithms*, 11(1):52–67, February 1990. ISSN 0196-6774.

Carl D. Meyer, editor. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA, USA, 2000. ISBN 0-89871-454-0.

Angelia Nedić and Asuman E. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Automat. Contr.*, 54(1):48–61, 2009.

Angelia Nedić, Alex Olshevsky, and Michael G. Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proc. of the IEEE*, 106(5):953–976, May 2018.

Giovanni Neglia, Gianmarco Calbi, Don Towsley, and Gayane Vardoyan. The Role of Network Topology for Distributed Machine Learning. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2019.

Francesco Pace, Daniele Venzano, Damiano Carra, and Pietro Michiardi. Flexible scheduling of distributed analytic applications. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid '17, pages 100–109, Piscataway, NJ, USA, 2017. IEEE Press. ISBN 978-1-5090-6610-0.

Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of asci q. In *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*, SC '03, pages 55–, 2003.

Shi Pu, Alex Olshevsky, and Ioannis Ch. Paschalidis. A non-asymptotic analysis of network independence for distributed stochastic gradient descent, 2019. arXiv preprint arXiv:1906.02702v9.

Leslie N Smith. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE, 2017.

Alexander Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *Proc. VLDB Endow.*, 3(1-2):703–710, September 2010. ISSN 2150-8097.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, September 1986.

Steven R. Young, Derek C. Rose, Travis Johnston, William T. Heller, Thomas P. Karnowski, Thomas E. Potok, Robert M. Patton, Gabriel Perdue, and Jonathan Miller. Evolving Deep Networks Using HPC. In *Proceedings of the Machine Learning on HPC Environments*, MLHPC'17, 2017.