Optimal Complexity in Decentralized Training

Yucheng Lu 1  Christopher De Sa 1

Abstract

Decentralization is a promising method of scaling up parallel machine learning systems. In this paper, we provide a tight lower bound on the iteration complexity for such methods in a stochastic non-convex setting. Our lower bound reveals a theoretical gap in known convergence rates of many existing decentralized training algorithms, such as D-PSGD. We prove by construction this lower bound is tight and achievable. Motivated by our insights, we further propose DeTAG, a practical gossip-style decentralized algorithm that achieves the lower bound with only a logarithm gap. Empirically, we compare DeTAG with other decentralized algorithms on image classification tasks, and we show DeTAG enjoys faster convergence compared to baselines, especially on unshuffled data and in sparse networks.

1. Introduction

Parallelism is a ubiquitous method to accelerate model training (Abadi et al., 2016; Alistarh, 2018; Alistarh et al., 2020; Lu et al., 2020). A parallel learning system usually consists of three layers (Table 1): an application to solve, a communication protocol deciding how parallel workers coordinate, and a network topology determining how workers are connected. Traditional design for these layers usually follows a centralized setup: in the application layer, training data is connected. Traditional design for these layers usually follows a centralized protocol AllReduce and shared among parallel workers; while in the protocol and network layers, workers either communicate via a fault-tolerant single central node (e.g. Parameter Server) (Li et al., 2014a,b; Ho et al., 2013) or a fully-connected topology (e.g. AllReduce) (Gropp et al., 1999; Patarasuk & Yuan, 2009). This centralized design limits the scalability of learning systems in two aspects. First, in many scenarios, such as Federated Learning (Koloskova et al., 2019a; McMahan et al., 2016) and Internet of Things (IOT) (Kanawaday & Sane, 2017), a shuffled dataset or a complete (bipartite) communication graph is not possible or affordable to obtain. Second, a centralized communication protocol can significantly slow down the training, especially with a low-bandwidth or high-latency network (Lian et al., 2017b; Tang et al., 2019b; Yu et al., 2018).

The rise of decentralization. To mitigate these limitations, decentralization comes to the rescue. Decentralizing the application and network allows workers to learn with unshuffled local datasets (Li et al., 2019) and arbitrary topologies (Seaman et al., 2017; Shanthamallu et al., 2017). Furthermore, the decentralized protocol, i.e. Gossip, helps to balance load, and has been shown to outperform centralized protocols in many cases (Lian et al., 2017a; Yu et al., 2019; Nazari et al., 2019; Lu & De Sa, 2020).

Understanding decentralization with layers. Many decentralized training designs have been proposed, which can lead to confusion as the term “decentralization” is used inconsistently in the literature. Some works use “decentralized” to refer to approaches that can tolerate non-iid or unshuffled datasets (Li et al., 2019), while others use it to mean gossip communication (Lian et al., 2017a), and still others use it to mean a sparse topology graph (Wan et al., 2020). To eliminate this ambiguity, we formulate Table 1, which summarizes the different “ways” a system can be decentralized. Note that the choices to decentralize different layers are independent, e.g., the centralized protocol AllReduce can still be implemented on a decentralized topology like the Ring graph (Wan et al., 2020).

<table>
<thead>
<tr>
<th>Layer</th>
<th>Centralized</th>
<th>Decentralized</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>Shuffled Data</td>
<td>Unshuffled Data (Federated Learning)</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>AllReduce/AllGather</td>
<td>Gossip</td>
</tr>
<tr>
<td><strong>Network Topology</strong></td>
<td>Complete- (Bipartite) Graph</td>
<td>Arbitrary Graph</td>
</tr>
</tbody>
</table>

Table 1. Design choice of centralization and decentralization in different layers of a parallel machine learning system. The protocol specifies how workers communicate. The topology refers to the overlay network that logically connects all the workers.

---

1Department of Computer Science, Cornell University, Ithaca, New York, United States. Correspondence to: Yucheng Lu <yl2967@cornell.edu>

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).
The theoretical limits of decentralization. Despite the empirical success, the best convergence rates achievable by decentralized training—and how they interact with different notions of decentralization—remains an open question. Previous works often show complexity of a given decentralized algorithm with respect to the number of iterations $T$ or the number of workers $n$, ignoring other factors including network topologies, function parameters or data distribution. Although a series of decentralized algorithms have been proposed showing theoretical improvements—such as using variance reduction (Tang et al., 2018b), acceleration (Seaman et al., 2017), or matching (Wang et al., 2019)—we do not know how close they are to an “optimal” rate or whether further improvement is possible.

In light of this, a natural question is: What is the optimal complexity in decentralized training? Has it been achieved by any algorithm yet? Previous works have made initial attempts on this question, by analyzing this theoretical limit in a non-stochastic or (strongly) convex setting (Seaman et al., 2017; Scaman et al., 2018; Koloskova et al., 2020; Woodworth et al., 2018; Dvinskikh & Gasnikov, 2019; Sun & Hong, 2019). These results provide great heuristics but still leave the central question open, since stochastic methods are usually used in practice and many real-world problems of interest are non-convex (e.g. deep learning). In this paper we give the first full answer to this question: our contributions are as follows.

• In Section 4, we prove the first (to our knowledge) tight lower bound for decentralized training in a stochastic non-convex setting. Our results reveal an asymptotic gap between our lower bound and known convergence rates of existing algorithms.

• In Section 5, we prove our lower bound is tight by exhibiting an algorithm called DeFacto that achieves it—albeit while only being decentralized in the sense of the application and network layers.

• In Section 6, we propose DeTAG, a practical algorithm that achieves the lower bound with only a logarithm gap and that is decentralized in all three layers.

• In Section 7, we experimentally evaluate DeTAG on the CIFAR benchmark and show it converges faster compared to decentralized learning baselines.

2. Related Work

Decentralized Training. In the application layer, decentralized training usually denotes federated learning (Zhao et al., 2018). Research on decentralization in this sense investigates convergence where each worker samples only from a local dataset which is not independent and identically distributed to other workers’ datasets (Bonawitz et al., 2019; Tran et al., 2019; Yang et al., 2019; Konečný et al., 2016). Another line of research on decentralization focuses on the protocol layer—with average gossip (Boyd et al., 2005; 2006), workers communicate by averaging their parameters with neighbors on a graph. D-PSGD (Lian et al., 2017a) is one of the most basic algorithms that scales SGD with this protocol, achieving a linear parallel speed up. Additional works extend D-PSGD to asynchronous and variance-reduced cases (Lian et al., 2017b; Tang et al., 2018b; Tian et al., 2020; Zhang & You, 2019b; Hendrikx et al., 2019; Xin et al., 2021a). After those, Zhang & You (2019c); Xin et al. (2019; 2021b) propose adding gradient trackers to D-PSGD. Other works discuss the application of decentralization on specific tasks such as linear models or deep learning (He et al., 2018; Assran et al., 2018). Zhang & You (2019a) treats the case where only directed communication can be performed. Wang et al. (2019) proposes using matching algorithms to optimize the gossip protocol. Multiple works discuss using compression to decrease communication costs in decentralized training (Koloskova et al., 2019b;a; Lu & De Sa, 2020; Tang et al., 2019a; 2018a), and other papers connect decentralized training to other parallel methods and present a unified theory (Lu et al., 2020; Koloskova et al., 2020; Wang & Joshi, 2018). In some even earlier works
like (Nedic & Ozdaglar, 2009; Duchi et al., 2010), full local gradients on a convex setting is investigated.

**Lower Bounds in Stochastic Optimization.** Lower bounds are a well-studied topic in non-stochastic optimization, especially in convex optimization (Agarwal & Bottou, 2014; Arjevani & Shamir, 2015; Lan & Zhou, 2018; Fang et al., 2018; Arjevani & Shamir, 2017). In the stochastic setting, Allen-Zhu (2018) and Foster et al. (2019) discuss the complexity lower bound to find stationary points on convex problems. Other works study the lower bound in a convex, data-parallel setting (Diakonikolas & Guzmán, 2018; Balkanski & Singer, 2018; Tran-Dinh et al., 2019), and Colin et al. (2019) extends the result to a model-parallel setting. In the domain of non-convex optimization, Carmon et al. (2017; 2019) propose a zero-chain model that obtains tight bound for a first order method to obtain stationary points. Zhou & Gu (2019) extends this lower bound to a finite sum setting, and Arjevani et al. (2019) proposes a probabilistic zero-chain model that obtains tight lower bounds for first-order methods on stochastic and non-convex problems.

### 3. Setting

In this section, we introduce the notation and assumptions we will use. Throughout the paper, we consider the standard *data-parallel* training setup with *n* parallel workers. Each worker *i* stores a copy of the model *x* ∈ ℝ^d and a local dataset *D* _i_. The model copy and local dataset define a local loss function (or empirical risk) *f* _i_. The ultimate goal of the parallel workers is to output a target model *x̂* that minimizes the average over all the local loss functions, that is,

\[ x̂ = \arg \min_{x \in \mathbb{R}^d} \left( \frac{1}{n} \sum_{i=1}^{n} E_{\xi_i \sim D_i} f_i(x; \xi_i) \right). \]  

(1)

Here, *ξ*_i is a data sample from *D*_i and is used to compute a stochastic gradient via some oracle, e.g. back-propagation on a mini-batch of samples. The loss functions can (potentially) be non-convex so finding a global minimum is NP-Hard; instead, we expect the workers to output a point *x̂* at which *f*(*x̂*) has a small gradient magnitude in expectation: \( E \|\nabla f(x̂)\| \leq \epsilon \), for some small \( \epsilon \). The assumptions our theoretical analysis requires can be categorized by the layers from Table 1: in each layer, “being decentralized” corresponds to certain assumptions (or lack of assumptions). We now describe these assumptions for each layer separately.

---

\[ 1 \leq \frac{1}{n} \sum_{i=1}^{n} \|\nabla f_i(x) - \nabla f(x)\| \leq \epsilon^2, \quad \forall x \in \mathbb{R}^d, \]  

(5)

for some constant \( \epsilon \), which is said to bound the “outer variance” among workers. This is often unreasonable, as it suggests the local datasets on workers must have close distribution: in practice, ensuring this often requires some sort of...
Table 2. Complexity comparison among different algorithms in the stochastic non-convex setting on arbitrary graphs. The blue text are the results from this paper. Definitions to all the parameters can be found in Section 3. Other algorithms like EXTRA (Shi et al., 2015) or MSDA (Scaman et al., 2017) are not comparable since they are designed for (strongly) convex problems. Additionally, Liu & Zhang (2021) provides alternative complexity bound for algorithms like D-PSGD which improves upon the spectral gap. However, the new bound would compromise the dependency on $\epsilon$, which does not conflict with our comparison here.

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Sample Complexity</th>
<th>Comm. Complexity</th>
<th>Gap to Lower Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>Theorem 1</td>
<td>Central $\Omega \left( \frac{\Delta L n^2}{n B} \right)$</td>
<td>$\Omega \left( \frac{\Delta L D}{\epsilon^2} \right)$</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Corollary 1</td>
<td>Decentral $\Omega \left( \frac{\Delta L n^2}{n B} \right)$</td>
<td>$\Omega \left( \frac{\Delta L D}{\epsilon^2} \right)$</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>DeFacto (Theorem 2)</td>
<td>Central $O \left( \frac{\Delta L n^2}{n B} \right)$</td>
<td>$O \left( \frac{\Delta L D}{\epsilon^2} \right)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>DeTAG (Theorem 3)</td>
<td>Decentral $O \left( \frac{\Delta L n^2}{n B} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
</tr>
<tr>
<td></td>
<td>D-PSGD (Lian et al., 2017a)</td>
<td>Decentral $O \left( \frac{\Delta L n^2}{n B} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
</tr>
<tr>
<td></td>
<td>SGP (Assran et al., 2019)</td>
<td>Decentral $O \left( \frac{\Delta L n^2}{n B} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
</tr>
<tr>
<td></td>
<td>D$^2$ (Tang et al., 2018b)</td>
<td>Decentral $O \left( \frac{\Delta L n^2}{n B} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
</tr>
<tr>
<td></td>
<td>DSGT (Zhang &amp; You, 2019c)</td>
<td>Decentral $O \left( \frac{\Delta L n^2}{n B} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
</tr>
<tr>
<td></td>
<td>GT-DSGD (Xin et al., 2021b)</td>
<td>Decentral $O \left( \frac{\Delta L n^2}{n B} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
<td>$O \left( \frac{\Delta L n_{nc}}{\epsilon^2 (1-\lambda)^2} \right)$</td>
</tr>
</tbody>
</table>

of shuffling or common centralized data source. We do not assume (5) but instead adopt the much weaker assumption

\[
\frac{1}{n} \sum_{i=1}^{n} \| \nabla f_i(0) - \nabla f(0) \|^2 \leq \varsigma_0^2, \tag{6}
\]

for constant $\varsigma_0 > 0$.\footnote{As we only use $\varsigma_0$ for upper bounds, not lower bounds, we do not define a “class” that depends on this parameter.} This assumption only requires a bound at point 0, which is, to the best of our knowledge, the weakest assumption of this type used in the literature (Tang et al., 2018b; Zhang & You, 2019c). Requiring such a weak assumption allows workers to (potentially) sample from different distributions or vary largely in their loss functions (e.g. in a federated learning environment).

### 3.2. Protocol Layer

Protocol-layer assumptions comprise constraints on the parallel learning algorithm itself, and especially on the way that the several workers communicate to approach consensus.

**Algorithm class ($B$).** We consider algorithms $A$ that divide training into multiple iterations, and between two adjacent iterations, there must be a synchronization process among workers (e.g. a barrier) such that they start each iteration simultaneously.\footnote{We consider synchronous algorithms only here for simplicity of presentation; further discussion of extension to asynchronous algorithms is included in the supplementary material.} Each worker running $A$ has a local copy of the model, and we let $x_{t,i} \in \mathbb{R}^d$ denote this model on worker $i$ at iteration $t$. We assume without loss of generality that $A$ initializes each local model at zero: $x_{0,i} = 0$ for all $i$. At each iteration, each worker makes at most $B$ queries to its gradient oracle $g_i$, for some constant $B \in \mathbb{N}^+$, and then uses the resulting gradients to update its model. We do not make any explicit rules for output and allow the output of the algorithm $\hat{x}_t$ at the end of iteration $t$ (the model that $A$ would output if it were stopped at iteration $t$) to be any linear combination of all the local models, i.e.

\[
\hat{x}_t \in \text{span}(\{x_{t,i}\}_{i \in [n]}) = \{\sum_{j=1}^{n} c_j x_{t,i} | c_j \in \mathbb{R}\}. \tag{7}
\]

Beyond these basic properties, we further require $A$ to satisfy the following “zero-respecting” property from Carmon et al. (2017). Specifically, if $z$ is any vector worker $i$ queries its gradient oracle with at iteration $t$, then for any $k \in [d]$, if $e_i^T z \neq 0$, then there exists a $s \leq t$ and a $j \in [n]$ such that either $j = i$ or $j$ is a neighbor of $i$ in the network connectivity graph $G$ (i.e. $(i, j) \in \{(i, i) \cup G\}$) and $(e_i^T x_{s,j}) \neq 0$. More informally, the worker will not query its gradient oracle with a nonzero value for some weight unless that weight was already nonzero in the model state of the worker or one of its neighbors at some point in the past. Similarly, for any $k \in [d]$, if $(e_k^T x_{t+1,i}) \neq 0$, then either there exists an $s \leq t$ and $j$ such that $(i, j) \in \{(i, i) \cup G\}$ and $(e_j^T x_{s,j}) \neq 0$, or one of the gradient oracle’s outputs $v$ on worker $i$ at iteration $t$ has $e_k^T v \neq 0$. Informally, a worker’s model will not have a nonzero weight unless either (1) that weight was nonzero on that worker or one of its neighbors at a previous iteration,
or (2) the corresponding entry in one of the gradients the worker sampled at that iteration was nonzero.

Intuitively, we are requiring that algorithm $A$ will not modify those coordinates that remain zero in all previous oracle outputs and neighboring models.\footnote{On the other hand, it is possible to even drop the zero-respecting requirement and extend $A$ to all the deterministic (not in the sense of sampling but the actual executions) algorithms. At a cost, we would need the function class to follow an “orthogonal invariant” property, and the model dimension needs to be large enough. We leave this discussion to the appendix.} This lets $A$ use a wide space of accessible information in communication and allows our class to cover first-order methods including SGD (Ghadimi & Lan, 2013), Momentum SGD (Nesterov, 1983), Adam (Kingma & Ba, 2014), RMSProp (Tieleman & Hinton, 2012), Adagrad (Ward et al., 2018), and AdaDelta (Zeiler, 2012). We let algorithm class $A_B$ denote the set of all algorithms $A$ that satisfy these assumptions.

So far our assumptions in this layer cover both centralized and decentralized protocols. Decentralized protocols, however, must satisfy the additional assumption that they communicate via gossip (see Section 2) (Boyd et al., 2005; 2006). A single step of gossip protocol can be expressed as

$$z_{t,i} \leftarrow \sum_{j \in N_i} y_{t,j} W_{ji}, \forall i \in [n] \tag{8}$$

for some constant doubly stochastic matrix $W \in \mathbb{R}^{n \times n}$ called the communication matrix and $y$ and $z$ are the input and output of the gossip communication step, respectively. The essence of a single Gossip step is to take weighted average over the neighborhood specified by a fixed matrix. To simplify later discussion, we further define the gossip matrix class $\mathcal{W}_n$ as the set of all matrices $W \in \mathbb{R}^{n \times n}$, where $W$ is doubly stochastic and $W_{ij} \neq 0$ only if $(i, j) \in G$. We call every $W \in \mathcal{W}_n$ a gossip matrix and we use $\lambda = \max\{|\lambda_2|, |\lambda_n|\} \in [0, 1)$ to denote its general second-largest eigenvalue, where $\lambda_i$ denotes the $i$-th largest eigenvalue of $W$. We let gossip algorithm class $A_B, W$ denote the set of all algorithms $A \in A_B$ that only communicate via gossip using a single matrix $W \in \mathcal{W}_n$. It trivially holds that $A_B, W \subset A_B$.

### 3.3. Topology Layer

Topology-layer assumptions comprise constraints on how workers are connected topologically. We let the graph class $\mathcal{G}_{n,D}$ denote the class of graphs $G$ connecting $n$ workers (vertices) with diameter $D$, where diameter of a graph measures the maximum distance between any two vertices (so $1 \leq D \leq n - 1$). A centralized analysis here typically will also require that $G$ be either complete or complete-bipartite (with parameter servers and workers as the two parts): lacking this requirement and allowing arbitrary graphs is what makes an analysis decentralized in the topology layer.

### 3.4. Complexity Measures

Now that we have defined the classes we are interested in, we can use them to define the complexity measures we will bound in our theoretical results. Given a loss function $f \in \mathcal{F}_{\Delta,L}$, a set of underlying oracles $O \in \mathcal{O}_{\sigma_d}$, a graph $G \in \mathcal{G}_{n,D}$, and an algorithm $A \in A_B$, let $\hat{\mathbf{x}}^A_{t,f,O,G}$ denote the output of algorithm $A$ at the end of iteration $t$ under this setting. Then the iteration complexity of $A$ solving $f$ under $O$ and $G$ is defined as

$$T_c(A, f, O, G) = \min \left\{ t \in \mathbb{N} \middle| \mathbb{E} \left[ \| \nabla f(\hat{\mathbf{x}}^A_{t,f,O,G}) \| \right] \leq \epsilon \right\},$$

that is, the least number of iterations required by $A$ to find a $\epsilon$-stationary-in-expectation point of $f$.

### 4. Lower Bound

Given the setup in Section 3, we can now present and discuss our lower bound on the iteration complexity. Note that in the formulation of protocol layer, the algorithm class $A_B$ only specifies the information available for each worker, and thus $A_B$ covers both centralization and decentralization in the protocol layer. Here, we show our lower bound in two parts: first a general bound where an arbitrary protocol that follows $A_B$ is allowed, and then a corollary bound for the case where only decentralized protocol is allowed.

#### 4.1. Lower Bound for Arbitrary Protocol

We start from the general bound. We expect this lower bound to show given arbitrary setting (functions, oracles and graph), the smallest iteration complexity we could obtain from $A_B$, i.e.

$$\inf_{A \in A_B} \sup_{f \in \mathcal{F}_{\Delta,L}} \sup_{O \in \mathcal{O}_{\sigma_d}} \sup_{G \in \mathcal{G}_{n,D}} T_c(A, f, O, G), \tag{9}$$

it suffices to construct a hard instance containing a loss function $\hat{f} \in \mathcal{F}_{\Delta,L}$, a graph $\hat{G} \in \mathcal{G}_{n,D}$ and a set of oracles $\hat{O} \in \mathcal{O}_{\sigma_d}$ and obtain a valid lower bound on $\inf_{A \in A_B} T_c(A, f, \hat{O}, \hat{G})$ since Equation (9) is always lower bounded by $\inf_{A \in A_B} T_c(A, f, \hat{O}, \hat{G})$.

For the construction, we follow the idea of the probabilistic zero-chain model (Carmon et al., 2017; 2019; Arjevani et al., 2019; Zhou & Gu, 2019), which is a special loss function where adjacent coordinates are closely dependent on each other like a “chain.” Our main idea is to use this function as $\hat{f}$ and split this chain onto different workers. Then the workers must conduct a sufficient number of optimization steps and rounds of communication to make progress.\footnote{For brevity, we leave details in the supplementary material.} From this, we obtain the following lower bound.
Theorem 1. For function class \( F_{\Delta,L} \), oracle class \( O_{\sigma} \) and graph class \( G_{n,D} \) defined with any \( \Delta > 0, L > 0, n \in \mathbb{N}^+ \), \( D \in \{1, 2, \ldots, n-1\}, \sigma > 0 \), and \( B \in \mathbb{N}^+ \), there exists \( f \in F_{\Delta,L}, O \in O_{\sigma}, \text{ and } G \in G_{n,D} \), such that no matter what \( A \in A_B \) is used, \( T_c(A,f,O,G) \) will always be lower bounded by

\[
\Omega \left( \frac{\Delta L \sigma^2}{nBc^4} + \frac{\Delta L}{\varepsilon^2} \right). \tag{10}
\]

**Dependency on the parameters.** The bound in Theorem 1 consists of a sample complexity term, which is the dominant one for small \( \varepsilon \), and a communication complexity term. We can see the increase of query budget \( B \) will only reduce the sample complexity. On the other hand, as the diameter \( D \) of a graph will generally increase as the number of vertices \( n \) increases, we can observe a trade-off between two terms when the system scales up: when more workers join the system, the communication complexity will gradually become the dominant term.

**Consistency with the literature.** Theorem 1 is tightly aligned with the state-of-the-art bounds in many settings. With \( n = B = D = 1 \), we recover the tight bound for sequential stochastic non-convex optimization \( \Theta(\Delta L \sigma^2 \varepsilon^{-4}) \) as shown in Arjevani et al. (2019). With \( \sigma = 0, D = 1 \), we recover the tight bound for sequential non-stochastic non-convex optimization \( \Theta(DL^{-2}) \) as shown in Carmon et al. (2019). With \( B = 1, D = 1 \), we recover the tight bound for centralized training \( \Theta(\Delta L \sigma^2(\eta e)^{-1}) \) given in Li et al. (2014b).

**Improvement upon previous results.** Previous works like Seaman et al. (2017); Scaman et al. (2018) provide similar lower bounds in a convex setting which relates to the diameter. However, these results treat \( D \) as a fixed value, i.e., \( D = n-1 \), and thus makes the bound to be only tight on linear graph. By comparison, Theorem 1 allows \( D \) to be chosen independently to \( n \).

### 4.2. Lower Bound for Decentralized Protocol

The bound in Theorem 1 holds for both centralized and decentralized protocols. A natural question is: **How would the lower bound adapt if the protocol is restricted to be decentralized?** i.e., the quantity of

\[
\inf_{A \in A_B, W} \sup_{f \in F_{\Delta,L}} \sup_{O \in O_{\sigma}} \sup_{G \in G_{n,D}} T_c(A,f,O,G),
\]

we can extend the lower bound to Gossip in the following corollary.

**Corollary 1.** For every \( \Delta > 0, L > 0, n \in \{2, 3, 4, \cdots\}, \lambda \in [0, \cos(\pi/n)], \sigma > 0 \), and \( B \in \mathbb{N}^+ \), there exists a loss function \( f \in F_{\Delta,L} \), a set of underlying oracles \( O \in O_{\sigma} \), a gossip matrix \( W \in W_n \) with second largest eigenvalue being \( \lambda \), and a graph \( G \in G_{n,D} \), such that no matter what

\[
\Omega \left( \frac{\Delta L \sigma^2}{nBc^4} + \frac{\Delta L}{\varepsilon^2 T - \lambda} \right). \tag{11}
\]

**Gap in the existing algorithms.** Comparing this lower bound with many state-of-the-art decentralized algorithms (Table 2), we can see they match on the sample complexity but leave a gap on the communication complexity. In many cases, the spectral gap significantly depends on the number of workers \( n \) and thus can be arbitrarily large. For example, when the graph \( G \) is a cycle graph or a linear graph, the gap of those baselines can increase by up to \( O(n^4) \) (Brooks et al., 2011; Gerencsér, 2011)!

### 5. DeFacto: Optimal Complexity in Theory

In the previous section we show the existing algorithms have a gap compared to the lower bound. This gap could indicate the algorithms are suboptimal, but it could also be explained by our lower bound being loose. In this section we address this issue by proposing DeFacto, an example algorithm showing the lower bound is achievable, which verifies the tightness of our lower bound—showing that (10) would hold with equality and \( \Theta(\cdot) \), not just \( \Omega(\cdot) \).

We start with the following insight on the theoretical gap: the goal of communication is to let all the workers obtain information from neighbors. Ideally, the workers would, at each iteration, perform (8) with \( W^T = I_n 1_n^T / n \), where \( 1_n \)
Algorithm 2 Decentralized Stochastic Gradient Tracking with By-Phase Accelerated Gossip (DeTAG) on worker $i$

Require: initialized model $x_{0,i}$, a copy of model $x_{0,i} \leftarrow x_{0,i}$, gradient tracker $g_{0,i}$, gradient buffer $g(0) = g_{(i-1)} = 0$, step size $\alpha$, a gossip matrix $W$, number of iterations $T$, neighbor list $N_i$

1: for $t = 0, 1, \ldots, T - 1$ do
2: $k \leftarrow \lfloor t/R \rfloor$
3: $r \leftarrow t \mod R$
4: Perform the $r$-th step in Accelerate Gossip:
   \[ \mathbf{x}_{t+1,i} \leftarrow AG(\mathbf{x}_{t,i}, W, \mathcal{N}_i, i) \]  
   \[ y_{t+1,i} \leftarrow AG(y_{t,i}, W, \mathcal{N}_i, i) \]
5: Spend all $B$ oracle budgets to compute stochastic gradient $g$ at point $x_{t,i}$ and accumulate it to gradient buffer: $g(r) \leftarrow g(r) + g$
6: if $r = T - 1$ then
7: Update gradient tracker and model:
   \[ \mathbf{x}_{t+1,i} \leftarrow \mathbf{x}_{t+1,i} - \alpha y_i \]  
   \[ y_{t+1,i} \leftarrow y_{t+1,i} + g(r) - g(r-1) \]
8: Reinitialize gradient buffer: $g(r-1) \leftarrow g(r)$ and then $g(r) = 0$
9: Copy the current model: $\mathbf{x}_{t+1,i} \leftarrow \mathbf{x}_{t+1,i}$
10: end if
11: end for
12: return $\hat{x} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_{t,i}$

Algorithm 3 Accelerated Gossip (AG) with R steps

Require: $x_{0,i}$, $W$, $\mathcal{N}_i$, $i$

1: $z_{-1,i} \leftarrow x_{0,i}$
2: $\eta \leftarrow \frac{1 - \sqrt{1 - \lambda^2}}{1 + \sqrt{1 - \lambda^2}}$
3: for $r = 0, 1, 2, \ldots, R - 1$ do
4: $z_{r+1,i} \leftarrow (1 + \eta) \sum_{j \in \mathcal{N}_i \cup \{i\}} z_{r,j} W_{ji} - \eta z_{r-1,i}$
5: end for
6: return $z_{R,i}$

is the $n$-dimensional all-one vector. We call this matrix the Average Consensus matrix. The Average Consensus is statistically equivalent to centralized communication (All-Reduce operation). However, due to the graph constraints, we can not use this $W^*$ unless workers are fully connected; instead, a general method is to repeatedly apply a sequence communication matrices in consecutive iterations and let workers achieve or approach the Average Consensus. Previous work uses Gossip matrix $W$ and expect $\prod_{r=1}^{R} W \approx 1_n 1_n^\top / n$ for some $R$. This $R$ is known to be proportional to the mixing time of the Markov Chain $W$ defines (Lu et al., 2020; Lu & De Sa, 2020), which is related to the inverse of its spectral gap (Levin & Peres, 2017). This limits convergence depending on the spectrum of the $W$ chosen. The natural question to ask here is: can we do better? What are the limits of how fast we can reach average consensus on a connectivity graph $G$? This question is answered by the following lemma.

**Lemma 1.** For any $G \in \mathcal{G}_{n,D}$, let $\mathcal{W}_G$ denote the set of $n \times n$ matrices such that for all $W \in \mathcal{W}_G$, $W_{ij} = 0$ if edge $(i, j)$ does not appear in $G$. There exists a sequence of $R$ matrices $\{W_r\}_{r \in [R]}$ that belongs to $\mathcal{W}_G$ such that $R \in [D, 2D]$ and

$$W_{R-1} W_{R-2} \cdots W_0 = \frac{1_n 1_n^\top}{n} = W^*.$$  

Lemma 1 is a classic result in the literature of graph theory. The formal proof and detailed methods to identify these matrices can be found in many previous works (Georgopoulos, 2011; Ko, 2010; Hendrickx et al., 2014). Here we treat this as a black box procedure.\(^6\)

Lemma 1 shows that we can achieve the exact average consensus by factorizing the matrix $1_n 1_n^\top / n$, and we can obtain the factors from a preprocessing step. From here, the path to obtain an optimal rate becomes clear: starting from $t = 0$, workers first spend $R$ iterations only computing stochastic gradients and then another $R$ iterations to reach consensus communicating via factors from Lemma 1; they then repeat this process until a stationary point is found. We call this algorithm DeFacto (Algorithm 1).

DeFacto is statistically equivalent to centralized SGD operating $T/2R$ iterations with a mini-batch size of $BR$. It can be easily verified that DeFacto holds membership in $A_B$. A straightforward analysis gives the convergence rate of DeFacto shown in the following Theorem.

**Theorem 2.** Let $A_1$ denote Algorithm 1. For $\mathcal{F}_{\Delta,L}$, $\mathcal{O}_{\sigma^2}$ and $\mathcal{G}_{n,D}$ defined with any $\Delta > 0$, $L > 0$, $n \in \mathbb{N}^+$, $D \in \{1, 2, \ldots, n - 1\}$, $\sigma > 0$, and $B \in \mathbb{N}^+$, the convergence rate of $A_1$ running on any loss function $f \in \mathcal{F}_{\Delta,L}$ any graph $G \in \mathcal{G}_{n,D}$, and any oracles $O \in \mathcal{O}_{\sigma^2}$ is bounded by

$$T_e(A_1, f, O, G) \leq O \left( \frac{\Delta \sigma^2}{nBc^2} + \frac{\Delta LD}{c^2} \right).$$  

(17)

Comparing Theorem 1 and Theorem 2, DeFacto achieves the optimal rate asymptotically. This shows that our lower bound in Theorem 1 is tight.

Despite its optimality, the design of DeFacto is unsatisfying in three aspects: (1) It compromises the throughput\(^7\) by a factor of two because in each iteration, a worker either communicates with neighbors or computes gradients but not both. This fails to overlap communication and computation and creates extra idle time for the workers. (2) It needs to iterate over all the factor matrices before it can query the gradient oracle at subsequent parameters. When diameter $D$ increases, the total time to finish such round will increase proportionally. (3) DeFacto works with decentralized data and arbitrary graph, achieving decentralization in both application and topology layers. However, the matrices used

\(^6\)We cover specific algorithms and details in the supplementary.

\(^7\)The number of stochastic gradients computed per iteration.
in Lemma 1 are not Gossip matrices as defined in $W_n$, and thus it fails to be decentralized in the protocol-layer sense.

6. DeTAG: Optimal Complexity in Practice

To address the limitations of DeFacto, a natural idea is to replace all the factor matrices in Lemma 1 with a gossip matrix $W$. The new algorithm after this mild modification is statistically equivalent to a D-PSGD variant: every $R$ iterations, it updates the model the same as one iteration in D-PSGD with a mini-batch size of $BR$ and communicate with a matrix $W^R$ whose second largest eigenvalue $\lambda = \lambda^R$, with $T/R$ iterations in total. However, even with arbitrarily large $R$, the communication complexity in this “updated D-PSGD” is still $O(\Delta L n \epsilon^2)$ (Table 2), leaving an $O(n \epsilon)$ gap compared to our lower bound.

To close this gap, we adopt two additional techniques: one is a gradient tracker $\hat{y}$ that is used as reference capturing gradient difference in the neighborhood; the other is using acceleration in gossip as specified in Algorithm 3. Modifying DeFacto results in Algorithm 2, which we call DeTAG. DeTAG works as follows: it divides the total number of iterations $T$ into several phases where each phase contains $R$ iterations. In each iteration, the communication process calls Accelerated Gossip to update a model replica $\tilde{x}$ and the gradient tracker (line 4) while the computation process constantly computes gradients at the same point (line 5). At the end of each phase, model $x$, its replica $\tilde{x}$ and gradient tracker $y$ are updated in line 7-10 and then DeTAG steps into the next phase. Aside from the two additional techniques, the main difference between DeTAG and DeFacto is that the communication matrix in DeTAG is a fixed gossip matrix $W$, which allows DeTAG to benefit from decentralization in the protocol layer as well as to adopt arbitrary $R \geq 1$ in practice (allowing $R$ to be tuned independently of $G$).

Improvement on design compared to baselines. Comparing with other baselines in Table 2, the design of DeTAG improves in the sense that (1) It removes the dependency on the outer variance $\varsigma$. (2) It drops the requirement on the gossip matrix assumed in Tang et al. (2018b). (3) The baseline DSGT (Zhang & You, 2019c) and GT-DSGD (Xin et al., 2021b) can be seen as special cases of taking $R = 1$ and $\eta = 0$ in DeTAG. That implies in practice, a well-tuned DeTAG can never perform worse than the baseline DSGT or GT-DSGD.

The convergence rate of DeTAG is given in the following theorem.

**Theorem 3.** Let $A_2$ denote Algorithm 2. For $F_{\Delta, L}$, $O_{\sigma^2}$ and $G_{n, D}$ defined with any $\Delta > 0$, $L > 0$, $n \in N^+$, $\lambda \in [0, 1)$, $\sigma > 0$, and $B \in N^+$, under the assumption of Equation (6), if we set the phase length $R$ to be

$$R = \max \left( \frac{1}{2} \log(n), \frac{1}{2} \log \left( \frac{s^2 \tau}{2L} \right) \right) \sqrt{1 - \lambda},$$

the convergence rate of $A_2$ running on any loss function $f \in F_{\Delta, L}$, any graph $G \in G_{n, D}$, and any oracles $O \in O_{\sigma^2}$ is bounded by

$$T_{o}(A_2, f, O, G) \leq O \left( \frac{\Delta L \sigma^2}{n B c^4} + \frac{\Delta L \log \left( n + \frac{\sqrt{n} q_0}{c^4 \sqrt{\Delta L}} \right)}{\epsilon^2 \sqrt{1 - \lambda}} \right).$$

Comparing Theorem 1 and Theorem 3, DeTAG achieves the optimal complexity with only a logarithm gap.

Improvement on complexity. Revisiting Table 2, we can see the main improvement of DeTAG’s complexity is in the two terms on communication complexity: (1) DeTAG only depends on the outer variance term $q_0$ inside a log, and (2) It reduces the dependency on the spectral gap $1 - \lambda$ to the lower bound of square root, as shown in Corollary 1.

Understanding the phase length $R$. In DeTAG, the phase length $R$ is a tunable parameter. Theorem 3 provides a suggested value for $R$. Intuitively, the value of $R$ captures the level of consensus of workers should reach before they step into the next phase. Theoretically, we observe $R$ is closely correlated to the mixing time of $W$: if we do not use acceleration in Gossip, then $R$ will become $O \left( \frac{1}{\sqrt{1 - \lambda}} \right)$, which is exactly the upper bound on the mixing time of the Markov Chain $W$ defines (Levin & Peres, 2017).

7. Experiments

In this section we empirically compare the performance of different algorithms. All the models and training scripts in this section are implemented in PyTorch and run on an Ubuntu 16.04 LTS cluster using a SLURM workload manager running CUDA 9.2, configured with 8 NVIDIA GTX 2080Ti GPUs. We launch one process from the host as one worker and let them use gloo as the communication backend. In each experiment, we compare the following algorithms: D-PSGD (Lian et al., 2017a), D$^2$ (Tang et al., 2018b), DSGT (Zhang & You, 2019c) and DeTAG. Note that GT-DSGD (Xin et al., 2021b) and DSGT (Zhang & You, 2019c) are essentially the same algorithm so we omit the comparison to GT-DSGD. Also note that SGP (Assran et al.,...
We can see in Figure 2 that when we shuffle less portion of the dataset, i.e., the dataset becomes more decentralized, DSGT fails to converge even with fine-tuned hyperparameter. Meanwhile, among D², DSGT and DeTAG, we can see DeTAG converges the fastest. When dataset becomes more decentralized, DSGT seems to receive more stable performance than D².

Convergence over different spectral gaps. In the second experiments, we proceed to explore the relation between convergence speed and spectral gap $1 - \lambda$ of the gossip matrix $\hat{W}$. We use 16 workers connected with a Ring graph to train Resnet20 on CIFAR10, and we generate a $W_0$ on such graph using Metropolis method. Then we adopt the slack matrix method to modify the spectral gap (Lu et al., 2020): $W_\kappa = \kappa W_0 + (1 - \kappa)I$, where $\kappa$ is a control parameter. We test $\kappa = 1, 0.1, 0.05, 0.01$ and plot the results in Figure 3. We can see with different $\kappa$, DeTAG is able to achieve faster convergence compared to baselines. When the network becomes sparse, i.e., $\kappa$ decreases, DeTAG enjoys more robust convergence.

8. Conclusion

In this paper, we investigate the tight lower bound on the iteration complexity of decentralized training. We propose two algorithms, DeFacto and DeTAG, that achieve the lower bound in terms of different decentralization in a learning system. DeTAG uses Gossip protocol, and is shown to be empirically competitive to many baseline algorithms, such as D-PSGD. In the future, we plan to investigate the variants of the complexity bound with respect to communication that are compressed, asynchronous, etc.

Acknowledgement

This work is supported by NSF IIS-2046760. The authors would like to thank A. Feder Cooper, Jerry Chee, Zheng Li, Ran Xin, Jiagu Zhang and anonymous reviewers from ICML 2021 for providing valuable feedbacks on earlier versions of this paper.
References


Optimal Complexity in Decentralized Training


