Cooperative SGD: A Unified Framework for the Design and Analysis of Communication-Efficient SGD Algorithms

Jianyu Wang [1] **Gauri Joshi** [1]

Abstract

Communication-efficient SGD algorithms, which allow nodes to perform local updates and periodically synchronize local models, are highly effective in improving the speed and scalability of distributed SGD. However, a rigorous convergence analysis and comparative study of different communication-reduction strategies remains a largely open problem. This paper presents a unified framework called Cooperative SGD that subsumes existing communication-efficient SGD algorithms such as periodic-averaging, elastic-averaging and decentralized SGD. By analyzing Cooperative SGD, we provide novel convergence guarantees for existing algorithms. Moreover, this framework enables us to design new communication-efficient SGD algorithms that strike the best balance between reducing communication overhead and achieving fast error convergence with low error floor.

1. Introduction

Stochastic gradient descent (SGD) is the backbone of most state-of-the-art machine learning algorithms. Due to its widespread applicability, speeding-up SGD is arguably the single most impactful and transformative problem in machine learning. Classical SGD was designed to be run on a single computing node, and its error-convergence has been extensively analyzed and improved in optimization and learning theory (Bottou et al., 2018). However, with the massive training datasets and deep neural network architectures used today, running SGD at a single node can be prohibitively slow. This calls for distributed implementations of SGD, where gradient computations and aggregation are parallelized across multiple worker nodes. Although parallelism boosts the amount of data processed per iteration, it exposes SGD to unpredictable synchronization and communication delays stemming from variability in the computing infrastructure.

Limitations of Parameter Server Framework. A commonly used method to parallelize gradient computation and process more training data per iteration is the parameter server framework (Li et al., 2014). Each of the m worker nodes computes the gradients of one mini-batch of data, and a parameter server aggregates these gradients and updates the model parameters. Synchronization delays caused by waiting for slow workers can be alleviated via asynchronous gradient aggregation. However, it is difficult to eliminate communication delays since by design, parameter server framework requires gradients and model updates to be communicated between the parameter server and workers after every iteration.

Communication-Efficient Distributed SGD Variants. To address the limitations of the parameter server framework, recent works have proposed several strategies to reduce the communication overhead in distributed SGD algorithm. A natural idea is to allow workers to perform τ local updates to the model instead of just computing gradients, and then periodically average the local models. Although extensive empirical results have validated the effectiveness of this periodic averaging strategy, rigorous theoretical understanding of how its convergence depends on the number of local updates τ is limited to convex setting (Stich, 2018). Instead of simple averaging, elastic-averaging SGD (EASGD) (Zhang et al., 2015) adds a proximal term to the objective function in order to allow some slack between local models. The efficiency of EASGD and its asynchronous and periodic averaging variants has been empirically validated. However, its convergence analysis under general convex or non-convex objectives is an open problem.

A different approach to reducing communication overhead is to perform decentralized training with a sparse-connected network of worker nodes. Each node only synchronizes with its one or very few neighbors. Decentralized (or gossip-type) training has a long history in the distributed and consensus optimization community and was successfully applied to deep learning in recent works (Lian et al., 2017; Jiang et al.,

[1]Department of Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Jianyu Wang <jianyuw1@andrew.cmu.edu>, Gauri Joshi <gaurij@andrew.cmu.edu>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2017). While the authors provide the convergence analysis for 1 local update per worker case, it is still unclear how decentralized training compares or combines with periodic averaging strategy.

Main Contributions. We note that a common thread in all the communication-efficient strategies described above is that *they allow worker nodes to perform local model-updates and limit the synchronization/consensus between the local models*. Based on this observation, we propose a powerful framework called *Cooperative SGD* that enables us to obtain an integrated analysis and comparison of communication-efficient algorithms. Existing algorithms including periodic averaging SGD (PASGD), EASGD, decentralized parallel SGD (D-PSGD) are all special cases of cooperative SGD, and thus can be analyzed under one single umbrella. By analyzing cooperative SGD, we provide the first convergence guarantee for EASGD with non-convex objective function and better optimization error bound for PASGD. The novel theoretical results can serve as guidelines to choose the best hyperparameters in practice. Moreover, the general framework greatly enlarges the design space of distributed SGD algorithms. One can easily design and analyze new efficient SGD variants by combining different strategies.

Notation. All vectors considered in this paper are column vectors. For convenience, we use $\mathbf{1}$ to denote $[1, 1, \ldots, 1]^\top$ and define matrix $\mathbf{J} = \mathbf{1}\mathbf{1}^\top/(\mathbf{1}^\top\mathbf{1})$. Unless otherwise stated, $\mathbf{1}$ is a size m column vector, and the matrix $\mathbf{J}$ and identity matrix $\mathbf{I}$ are of size $m \times m$, where m is the number of workers.

2. The Cooperative SGD Framework

The Cooperative SGD algorithm is denoted by $\mathcal{A}(\tau, \mathbf{W}, v)$, where τ is the number of local updates, $\mathbf{W}$ is the mixing matrix used for model averaging, and v is the number of auxiliary variables.

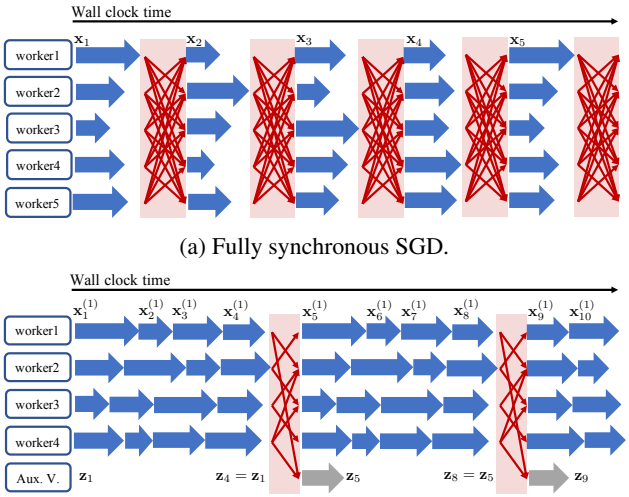
At iteration k, the m workers have different versions $\mathbf{x}_k^{(1)}, \ldots, \mathbf{x}_k^{(m)} \in \mathbb{R}^d$ of the model. In addition, there are v auxiliary variables $\mathbf{z}_k^{(1)}, \ldots, \mathbf{z}_k^{(v)}$ that are either stored at v additional nodes or at one or more of the workers, depending upon implementation. These model versions will update at each iteration as follows.

1. **Local Compute Phase.** The workers evaluate the gradient $g(\mathbf{x}_k^{(i)})$ for one mini-batch of data and update $\mathbf{x}_k^{(i)}$. The auxiliary variables are only updated by averaging a subset of the local models as described in point 2 below. Thus, their gradients are zero.

2. **Model Averaging Phase.** The local models and auxiliary variables are averaged with neighbors according to matrix $\mathbf{S}_k \in \mathbb{R}^{(m+v)\times(m+v)}$. To capture periodic-averaging pattern, we use a time-varying $\mathbf{S}_k$ that varies as:

$$\mathbf{S}_k = \begin{cases} \mathbf{W}, & k \bmod \tau = 0 \\ \mathbf{I}_{(m+v)\times(m+v)}, & \text{otherwise}, \end{cases} \tag{1}$$

where the identity matrix $\mathbf{I}$ means that there is no inter-node communication during the τ local updates.



(a) Fully synchronous SGD.

(b) Cooperative SGD.

Figure 1: Illustration of the execution pipeline of cooperative SGD. Blue, red, grey arrows represent gradient computation, worker communication, and update of auxiliary variables respectively.

We now present a general update rule that combines the above elements. Define matrices $\mathbf{X}_k, \mathbf{G}_k \in \mathbb{R}^{d\times(m+v)}$ that concatenate all local models and gradients:

$$\mathbf{X}_k = [\mathbf{x}_k^{(1)}, \ldots, \mathbf{x}_k^{(m)}, \mathbf{z}_k^{(1)}, \ldots, \mathbf{z}_k^{(v)}], \tag{2}$$

$$\mathbf{G}_k = [g(\mathbf{x}_k^{(1)}), \ldots, g(\mathbf{x}_k^{(m)}), \mathbf{0}, \ldots, \mathbf{0}]. \tag{3}$$

The update rule in terms of these matrices is

$$\mathbf{X}_{k+1} = (\mathbf{X}_k - \eta\mathbf{G}_k)\,\mathbf{S}_k. \tag{4}$$

Remark 1. *Instead of using update* (4)*, one can use an alternative rule:* $\mathbf{X}_{k+1} = \mathbf{X}_k\mathbf{W}_k - \eta\mathbf{G}_k$*. The convergence analyses and insights in this paper can be extended to this update rule.*

The framework improves the communication-efficiency of distributed SGD in three different ways. We illustrate these in Figure 1, which compares the execution timeline of cooperative SGD with fully synchronous SGD.

3. Unified Convergence Analysis

In this section, we present the unified convergence analysis of algorithms in cooperative SGD framework. Due to the space limitations, please refer to (Wang & Joshi, 2018) to check the assumptions [1] and proof details.

Theorem 1 (Convergence of Cooperative SGD). *For algorithm $\mathcal{A}(\tau, \mathbf{W}, v)$, suppose the total number of iterations K can be divided by the communication period τ. Under certain assumptions (stated in (Wang & Joshi, 2018)), if the learning rate satisfies $\eta_{\mathit{eff}}L + 5\eta_{\mathit{eff}}^2L^2[(1+\frac{v}{m})\frac{\tau}{1-\zeta}]^2 \leq 1$ where $\zeta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_{m+v}(\mathbf{W})|\}$, and all local models are initialized at a same point $\mathbf{u}_1$, then after K iterations,*

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\|\nabla F(\mathbf{u}_k)\|^2\right] \leq \underbrace{\frac{2\left[F(\mathbf{u}_1)-F_{\mathit{inf}}\right]}{\eta_{\mathit{eff}}K} + \frac{\eta_{\mathit{eff}}L\sigma^2}{m}}_{\textit{fully sync SGD}} + \underbrace{\eta_{\mathit{eff}}^2L^2\sigma^2\left(\frac{1+\zeta^2}{1-\zeta^2}\tau - 1\right)\left(1+\frac{v}{m}\right)^2}_{\textit{network error}} \tag{5}$$

where $\mathbf{u}_k = \mathbf{X}_k\mathbf{J}, \eta_{\mathit{eff}} = \frac{m}{m+v}\eta$ are averaged model and effective learning rate, respectively.

Error decomposition. It is worth noting that the upper bound is decomposed into two parts. The first two terms are same as the optimization error bound in fully synchronous SGD (Bottou et al., 2018). The last term is *network error*, resulted from performing local updates and reducing inter-worker communication. It directly increases the error floor at convergence. Note that the network error term is in a higher order of learning rate. Thus, by setting a small or decaying learning rate, the negative effect of communication reduction can be omitted.

Dependence on $\tau, \mathbf{W}, v$. Theorem 1 states that the error floor at convergence will monotonically increase along with communication period τ and the second largest eigenvalue magnitude ζ. The value of ζ reflects the mixing rates of different variables. When there is no communication among local workers, then $\mathbf{W} = \mathbf{I}_{m+v}$ and $\zeta = 1$; When local models are fully synchronized, then $\mathbf{W} = \mathbf{J}_{m+v}$ and $\zeta = 0$. Typically, a sparser matrix means a larger value of ζ. Moreover, note that the effective learning rate is determined by the number of auxiliary variables.

4. Novel Analyses for Existing Algorithms.

By setting different hyperparameters in Theorem 1, one can easily model existing algorithms as special cases and obtain the convergence guarantee.

[1] We do not assume convex objective function.

Fully synchronous SGD $\Leftrightarrow \mathcal{A}(1, \mathbf{J}, 0)$. The local models are synchronized with all other workers after every iteration.
PASGD $\Leftrightarrow \mathcal{A}(\tau, \mathbf{J}, 0)$. The local models are synchronized with all other workers after every τ iterations.

D-PSGD $\Leftrightarrow \mathcal{A}(1, \mathbf{W}, 0)$. The mixing matrix $\mathbf{W}$ in D-PSGD is fixed as a sparse matrix. Only one local update before averaging is considered and there are no auxiliary variables.

EASGD $\Leftrightarrow \mathcal{A}(1, \mathbf{W}_\alpha, 1)$. In EASGD, there is one auxiliary variable. Besides, the mixing matrix is controlled by a hyper-parameter α as follows

$$\mathbf{W}_\alpha = \begin{bmatrix} (1-\alpha)\mathbf{I} & \alpha\mathbf{1} \\ \alpha\mathbf{1}^\top & 1-m\alpha \end{bmatrix} \in \mathbb{R}^{(m+1)\times(m+1)}. \tag{6}$$

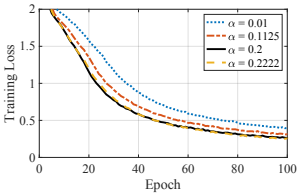
One can validate that the updates defined in (1), (4) and (6) are equivalent to the updates in (Zhang et al., 2015) when using the alternative update rule $\mathbf{X}_{k+1} = \mathbf{X}_k\mathbf{S}_k - \eta\mathbf{G}_k$.

The unified analysis brings new insights such as the best choice of elasticity parameter α.

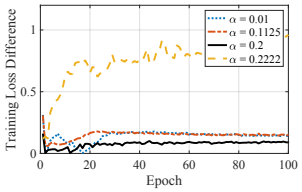
Lemma 1 (Best Choice of α). *If $\alpha = 2/(m+2)$, then the second largest absolute eigenvalue of $\mathbf{W}_\alpha$ (6), achieves the minimal value $m/(m+2)$.*

Accordingly, by choosing the best α, the upper bound of error floor at convergence given by (5) can also be minimized.

To the best of our knowledge, Theorem 1 together with Lemma 1 provide the first convergence result for EASGD with general objectives and also the first theoretical justification for the best choice of α. In Figure 2, we validate our theoretical findings. Please refer to (Wang & Joshi, 2018) to check more empirical results.



(a) Training loss of workers.



(b) The difference of training loss between workers and the auxiliary variable.

Figure 2: EASGD training on CIFAR-10 with VGG-16. Since there are 8 worker nodes and 1 auxiliary variable, the best value of α given by Lemma 1 is $2/(m+2) = 0.2$, which performs better than the empirical choice $\alpha = 0.9/m = 0.1125$ suggested in (Zhang et al., 2015). The best choice of α yields the lowest training loss and the least discrepancies between workers and auxiliary variable.

References

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Jiang, Z., Balu, A., Hegde, C., and Sarkar, S. Collaborative deep learning in fixed topology networks. In *Advances in Neural Information Processing Systems*, pp. 5906–5916, 2017.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B.-Y. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pp. 583–598, 2014.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 5336–5346, 2017.

Stich, S. U. Local SGD converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Wang, J. and Joshi, G. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

Zhang, S., Choromanska, A. E., and LeCun, Y. Deep learning with elastic averaging SGD. In *Advances in Neural Information Processing Systems*, pp. 685–693, 2015.