Communication-Efficient Distributed Stochastic AUC Maximization with Deep Neural Networks

Zhishuai Guo [1] **Mingrui Liu** [1] **Zhuoning Yuan** [1] **Li Shen** [2] **Wei Liu** [2] **Tianbao Yang** [1]

Abstract

In this paper, we study distributed algorithms for large-scale AUC maximization with a deep neural network as a predictive model. Although distributed learning techniques have been investigated extensively in deep learning, they are not directly applicable to stochastic AUC maximization with deep neural networks due to its striking differences from standard loss minimization problems (e.g., cross-entropy).Towards addressing this challenge, we propose and analyze a communication-efficient distributed optimization algorithm based on a *non-convex concave* reformulation of the AUC maximization, in which the communication of both the primal variable and the dual variable between each worker and the parameter server only occurs after multiple steps of gradient-based updates in each worker. Compared with the naive parallel version of an existing algorithm that computes stochastic gradients at individual machines and averages them for updating the model parameters, our algorithm requires a much less number of communication rounds and still achieves a linear speedup in theory. To the best of our knowledge, this is the **first** work that solves the *non-convex concave min-max* problem for AUC maximization with deep neural networks in a communication-efficient distributed manner while still maintaining the linear speedup property in theory. Our experiments on several benchmark datasets show the effectiveness of our algorithm and also confirm our theory.

[1]Department of Computer Science, The University of Iowa, Iowa City, IA 52242, USA [2]Tencent AI Lab, Shenzhen, China. Correspondence to: Tianbao Yang <tianbao-yang@uiowa.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

1. Introduction

Large-scale distributed deep learning (Dean et al., 2012; Li et al., 2014) has achieved tremendous successes in various domains, including computer vision (Goyal et al., 2017), natural language processing (Devlin et al., 2018; Yang et al., 2019), generative modeling (Brock et al., 2018), reinforcement learning (Silver et al., 2016; 2017), etc. From the perspective of learning theory and optimization, most of them are trying to minimize a surrogate loss of a specific error measure using parallel minibatch stochastic gradient descent (SGD). For example, on the image classification task, the surrogate loss is usually the cross entropy between the estimated probability distribution according to the output of a certain neural network and the vector encoding ground-truth labels (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2016), which is a surrogate loss of the misclassification rate. Based on the surrogate loss, parallel minibatch SGD (Goyal et al., 2017) is employed to update the model parameters.

However, when the data for classification is imbalanced, AUC (short for Area Under the ROC Curve) is a more suitable measure (Elkan, 2001). AUC is defined as the probability that a positive sample has a higher score than a negative sample (Hanley & McNeil, 1982; 1983). Despite the tremendous applications of distributed deep learning in different fields, the study about optimizing AUC with distributed deep learning technologies is rare. The commonly used parallel mini-batch SGD for minimizing a surrogate loss of AUC will suffer from high communication costs in a distributed setting due to the non-decomposability nature of AUC measure. The reason is that positive and negative data pairs that define a surrogate loss for AUC may sit on different machines. To the best of our knowledge, Liu et al. (2020b) is the only work trying to optimize a surrogate loss of *AUC with a deep neural network* that explicitly tackles the non-decomposability of AUC measure. Nevertheless, their algorithms are designed only for the single-machine setting and hence are far from sufficient when encountering a huge amount of data. Although a naive parallel version of the stochastic algorithms proposed in (Liu et al., 2020b) can be used for distributed AUC maximization with a deep neural network, it would still suffer from high communication

overhead due to a large number of communication rounds.

In this paper, we bridge the gap between stochastic AUC maximization and distributed deep learning by proposing a communication-efficient distributed algorithm for stochastic AUC maximization with a deep neural network. **The focus is to make the total number of communication rounds much less than the total number of iterative updates.** We build our algorithm upon the nonconvex-concave min-max reformulation of the original problem. The key ingredient is to design a communication-efficient distributed algorithm for solving the regularized min-max subproblems using multiple machines. Specifically, we follow the proximal primal-dual algorithmic framework proposed by (Rafique et al., 2018; Liu et al., 2020b), i.e., by solving a sequence of quadratically regularized min-max saddle-point problems with periodically updated regularizers successively. **The key difference** is that the inner min-max problem solver is built on a distributed periodic model averaging technique, which consists of a fixed number of stochastic primal-dual updates over individual machines and a small number of averagings of model parameters from multiple machines. This mechanism can greatly reduce the communication cost, which is similar to (Zhou & Cong, 2017; Stich, 2018; Yu et al., 2019b). However, their analysis cannot be applied to our case since their analysis only works for convex or non-convex *minimization* problems. In contrast, our algorithm is designed for a particular *non-convex concave min-max* problem induced by the original AUC maximization problem. Our contributions are summarized as follows:

- We propose a communication-efficient distributed stochastic algorithm named CoDA for solving a nonconvex-concave min-max reformulation of AUC maximization with deep neural networks by local primal-dual updating and periodically global variable averaging. To our knowledge, this is the first communication-efficient distributed stochastic algorithm for learning a deep neural network by AUC maximization.

- We analyze the iteration complexity and communication complexity of the proposed algorithm under the commonly used Polyak- Łojasiewicz (PL) condition as in (Liu et al., 2020b). Comparing with (Liu et al., 2020b), our theoretical result shows that the iteration complexity can be reduced by a factor of K (the number of machines) in a certain region, while the communication complexity (the rounds of communication) is much lower than that of a naive distributed version of the stochastic algorithm proposed in (Liu et al., 2020b). The summary of iteration and communication complexities is given in Table 1.
- We verify our theoretical claims by conducting experiments on several large-scale benchmark datasets. The experimental results show that our algorithm indeed exhibits good acceleration performance in practice.

2. Related Work

Stochastic AUC Maximization. It is challenging to directly solve the stochastic AUC maximization in the online learning setting since the objective function of AUC maximization depends on a sum of pairwise losses between samples from positive and negative classes. Zhao et al. (2011) addresses this problem by maintaining a buffer to store representative data samples, employing the reservoir sampling technique to update the buffer, calculating gradient information based on the data in the buffer, and then performing a gradient-based update rule to update the classifier. Gao et al. (2013) does not maintain a buffer, while they instead maintained first-order and second-order statistics of the received data to update the classifier by gradient-based update. Both of them are infeasible in big data scenarios since Zhao et al. (2011) suffers from a large amount of training data and Gao et al. (2013) is not suitable for high dimensional data. Ying et al. (2016) addresses these issues by introducing a min-max reformulation of the original problem and solving it by a primal-dual stochastic gradient method (Nemirovski et al., 2009), in which no buffer is needed and the per-iteration complexity is the same magnitude as the dimension of the feature vector. Natole et al. (2018) improves the convergence rate by adding a strongly convex regularizer upon the original formulation. Based on the same saddle point formulation as in (Ying et al., 2016), Liu et al. (2018) gets an improved convergence rate by developing a multi-stage algorithm without adding the strongly convex regularizer. However, all of these studies focus on learning a linear model. Recently, Liu et al. (2020b) considers stochastic AUC maximization for learning a deep non-linear model, in which they designed a proximal primal-dual gradient-based algorithm under the PL condition and established non-asymptotic convergence results.

Communication Efficient Algorithms. There are multiple approaches for reducing the communication cost in distributed optimization, including skipping communication and compression techniques. Due to limit of space, we mainly review the literature on skipping communication. For compression techniques, we refer the readers to (Jiang & Agrawal, 2018; Stich et al., 2018; Basu et al., 2019; Wangni et al., 2018; Bernstein et al., 2018) and references therein. Skipping communication is realized by doing multiple local gradient-based updates in each worker before aggregating the local model parameters. One special case is so-called one-shot averaging (Zinkevich et al., 2010; McDonald et al., 2010; Zhang et al., 2013), where each machine solves a local problem and averages these solutions only at the last iterate. Some works (Zhang et al., 2013; Shamir & Srebro, 2014; Godichon-Baggioni & Saadane, 2017; Jain et al.,

Table 1. The summary of Iteration and Communication Complexities, where K is the number of machines and $\mu \leq 1$. NP-PPD-SG denotes the naive parallel version of PPD-SG, which is also a special case of our algorithm, whose complexities can be derived by following our analysis.

Algorithm	Setting	Iteration Complexity	Communication Complexity
PPD-SG (Liu et al., 2020b)	Single	$O(1/(\mu^2\epsilon))$	-
NP-PPD-SG	Distributed	$O(1/(K\mu^2\epsilon))$	$O(1/(K\mu^2\epsilon))$
CoDA	Distributed	$O(1/(K\mu^2\epsilon))$	$O(1/(\mu^{3/2}\epsilon^{1/2}))$

2017; Koloskova et al., 2019; Koloskova* et al., 2020) consider one-shot averaging with one-pass of the entire data and establish statistical convergence, which is usually not able to guarantee the convergence of the training error. The scheme of local SGD update at each worker with skipping communication is analyzed for convex (Stich, 2018; Jaggi et al., 2014) and nonconvex problems (Zhou & Cong, 2017; Jiang & Agrawal, 2018; Wang & Joshi, 2018b; Lin et al., 2018b; Wang & Joshi, 2018a; Yu et al., 2019b;a; Basu et al., 2019; Haddadpour et al., 2019). There are also several empirical studies (Povey et al., 2014; Su & Chen, 2015; McMahan et al., 2016; Chen & Huo, 2016; McMahan et al., 2016; Lin et al., 2018b; Kamp et al., 2018) showing that this scheme exhibits good empirical performance for distributed deep learning. However, all of these works only consider minimization problems and do not apply to the nonconvex-concave min-max formulation as considered in this paper.

Nonconvex Min-max Optimization. Stochastic nonconvex min-max optimization has garnered increasing attention recently (Rafique et al., 2018; Lin et al., 2018a; Sanjabi et al., 2018; Lu et al., 2019; Lin et al., 2019; Jin et al., 2019; Liu et al., 2020a). Rafique et al. (2018) considered the case where the objective function is weakly-convex and concave and proposed an algorithm based on the spirit of proximal point method (Rockafellar, 1976), in which a proximal subproblem with periodically updated reference points is approximately solved by an appropriate stochastic algorithm. They established the convergence to a nearly stationary point for the equivalent minimization problem. Under the same setting, Lu et al. (2019) designed a block-based algorithm and showed that it can converge to a solution with a small stationary gap, and Lin et al. (2019) considered solving the problem using vanilla stochastic gradient descent ascent and established its convergence to a stationary point under the smoothness assumption. There are also several papers (Lin et al., 2018a; Sanjabi et al., 2018; Liu et al., 2020a) trying to solve non-convex non-concave min-max problems. Lin et al. (2018a) proposed an inexact proximal point method for solving a class of weakly-convex weakly-concave problems, which was proven to converge to a nearly stationary point. Sanjabi et al. (2018) exploited the PL condition for the inner maximization problem and designed a multi-step alternating optimization algorithm which was able to converge to a stationary point. Liu et al. (2020a) considered solving a class of nonconvex-nonconcave min-max problems by designing an adaptive gradient method and established an adaptive complexity for finding a stationary point. However, none of them is particularly designed for the distributed stochastic AUC maximization problem with a deep neural network.

3. Preliminaries and Notations

The area under the ROC curve (AUC) on a population level for a scoring function $h : \mathcal{X} \to \mathbb{R}$ is defined as

$$AUC(h) = \Pr(h(\mathbf{x}) \geq h(\mathbf{x}')|y = 1, y' = -1), \quad (1)$$

where $\mathbf{z} = (\mathbf{x}, y)$ and $\mathbf{z}' = (\mathbf{x}', y')$ are drawn independently from $\mathbb{P}$. By employing the squared loss as the surrogate for the indicator function which is commonly used by previous studies (Gao et al., 2013; Ying et al., 2016; Liu et al., 2018; 2020b), the deep AUC maximization problem can be formulated as

$$\min_{\mathbf{w}\in\mathbb{R}^d} \mathbb{E}_{\mathbf{z},\mathbf{z}'}\left[(1 - h(\mathbf{w};\mathbf{x}) + h(\mathbf{w};\mathbf{x}'))^2 | y = 1, y' = -1\right],$$

where $h(\mathbf{w};\mathbf{x})$ denotes the prediction score for a data sample $\mathbf{x}$ made by a deep neural network parameterized by $\mathbf{w}$. It was shown in (Ying et al., 2016) that the above problem is equivalent to the following min-max problem:

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^d \\ (a,b)\in\mathbb{R}^2}} \max_{\alpha\in\mathbb{R}} f(\mathbf{w}, a, b, \alpha) = \mathbb{E}_{\mathbf{z}}[F(\mathbf{w}, a, b, \alpha, \mathbf{z})], \quad (2)$$

where

$$\begin{aligned} F(\mathbf{w}, a, b, \alpha; \mathbf{z}) &= (1-p)(h(\mathbf{w};\mathbf{x}) - a)^2\mathbb{I}_{[y=1]} \\ &+ p(h(\mathbf{w};\mathbf{x}) - b)^2\mathbb{I}_{[y=-1]} + 2(1+\alpha)(ph(\mathbf{w};\mathbf{x})\mathbb{I}_{[y=-1]} \\ &- (1-p)h(\mathbf{w},\mathbf{x})\mathbb{I}_{[y=1]}) - p(1-p)\alpha^2, \end{aligned}$$

where $p = \Pr(y = 1)$ denotes the prior probability that an example belongs to the positive class, and $\mathbb{I}$ denotes an indicator function. The above min-max reformulation allows us to decompose the expectation over all data into the expectation over data on individual machines.

In this paper, we consider the following distributed AUC maximization problem:

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^d \\ (a,b)\in\mathbb{R}^2}} \max_{\alpha\in\mathbb{R}} f(\mathbf{w}, a, b, \alpha) = \frac{1}{K}\sum_{k=1}^{K} f_k(\mathbf{w}, a, b, \alpha), \quad (3)$$

where K is the total number of machines, $f_k(\mathbf{w}, a, b, \alpha) = \mathbb{E}_{\mathbf{z}^k}[F_k(\mathbf{w}, a, b, \alpha; \mathbf{z}^k)]$, $\mathbf{z}^k = (\mathbf{x}^k, y^k) \sim \mathbb{P}_k$, $\mathbb{P}_k$ is the

data distribution on machine k, and $F_k(\mathbf{w}, a, b, \alpha; \mathbf{z}^k) = F(\mathbf{w}, a, b, \alpha; \mathbf{z}^k)$. Our goal is to utilize K machines to jointly solve the optimization problem (3). We emphasize that the k-th machine can only access data $\mathbf{z}^k \sim \mathbb{P}_k$ of its own. It is notable that our formulation includes both the batch-learning setting and the online learning setting. For the batch-learning setting, $\mathbb{P}_k$ represents the empirical distribution of data on the k-th machine and p denotes the empirical positive ratio for all data. For the online learning setting, $\mathbb{P}_k = \mathbb{P}, \forall k$ represents the same population distribution of data and p denotes the positive ratio in the population level.

Notations. We define the following notations:

$$\mathbf{v} = (\mathbf{w}^T, a, b)^T, \quad \phi(\mathbf{v}) = \max_{\alpha} f(\mathbf{v}, \alpha),$$
$$\phi_s(\mathbf{v}) = \phi(\mathbf{v}) + \frac{1}{2\gamma}\|\mathbf{v} - \mathbf{v}_{s-1}\|^2,$$
$$\mathbf{v}_\phi^* = \arg\min_{\mathbf{v}} \phi(\mathbf{v}), \quad \mathbf{v}_{\phi_s}^* = \arg\min_{\mathbf{v}} \phi_s(\mathbf{v}).$$

We make the following assumption throughout this paper.

Assumption 1
(i) There exist $\mathbf{v}_0, \Delta_0 > 0$ *such that* $\phi(\mathbf{v}_0) - \phi(\mathbf{v}_\phi^*) \leq \Delta_0$.
(ii) For any $\mathbf{x}$, $\|\nabla h(\mathbf{w}; \mathbf{x})\| \leq G_h$.
(iii) $\phi(\mathbf{v})$ *satisfies the* μ*-PL condition, i.e.,* $\mu(\phi(\mathbf{v}) - \phi(\mathbf{v}_*)) \leq \frac{1}{2}\|\nabla\phi(\mathbf{v})\|^2$*;* $\phi(\mathbf{v})$ *is* L_1*-smooth, i.e.,* $\|\phi(\mathbf{v}_1) - \phi(\mathbf{v}_2)\| \leq L_1\|\mathbf{v}_1 - \mathbf{v}_2\|$.
(iv) For any $\mathbf{x}$, $h(\mathbf{w}; \mathbf{x})$ *is* L_h*-smooth, and* $h(\mathbf{w}; \mathbf{x}) \in [0, 1]$.

Remark: Assumptions (i), (ii), (iii) and $h(\mathbf{w}; \mathbf{x}) \in [0, 1]$ of (iv) are also assumed in (Liu et al., 2020b), which have been justified as well. L-smoothness of function h is a standard assumption in the optimization literature. Finally, it should be noted that μ is usually much smaller than 1 (Yuan et al., 2019). This is important for us to understand our theoretical result later.

4. Main Result and Theoretical Analysis

In this section, we first describe our algorithm, and then present its convergence result followed by its analysis. For simplicity, we assume that the ratio p of data with the positive label is known. For the batch learning setting, p is indeed the empirical ratio of positive examples. For the online learning setting with an unknown distribution, we can follow the online estimation technique in (Liu et al., 2020b) to do the parameter update.

Algorithm 1 describes the proposed algorithm CoDA for optimizing AUC in a communication-efficient distributed manner. CoDA shares the same algorithmic framework as proposed in (Liu et al., 2020b). In particular, we employ a proximal-point algorithmic scheme that successively solves

Algorithm 1 CoDA

1: Initialization: $(\mathbf{v}_0 = \mathbf{0} \in \mathbb{R}^{\alpha+2}, \alpha_0 = 0, \gamma)$.
2: **for** $s = 1, ..., S$ **do**
3: $\quad \mathbf{v}_s = \text{DSG}(\mathbf{v}_{s-1}, \alpha_{s-1}, \eta_s, T_s, m_s, I_s, \gamma)$;
4: $\quad$ Each machine draws a minibatch $\{\mathbf{z}_1^k, ..., \mathbf{z}_{m_s}^k\}$ of size m_s and does:
5: $\quad\quad h_-^k = \sum_{i=1}^{m_s} h(\mathbf{v}_s; \mathbf{x}_i^k)\mathbb{I}_{y_i^k=-1}, N_-^k = \sum_{i=1}^{m_s} \mathbb{I}_{y_i^k=-1}$,
6: $\quad\quad h_+^k = \sum_{i=1}^{m_s} h(\mathbf{v}_s; \mathbf{x}_i^k)\mathbb{I}_{y_i^k=1}, N_+^k = \sum_{i=1}^{m_s} \mathbb{I}_{y_i^k=1}$;
7: $\quad \alpha_s = \frac{1}{K}\sum_{k=1}^{K}\left[\frac{h_-^k}{N_-^k} - \frac{h_+^k}{N_+^k}\right]$; $\quad\quad \diamond$ communicate
8: **end for**
9: Return $\mathbf{v}_S$.

the following convex-concave problem approximately:

$$\min_{\mathbf{v}} \max_{\alpha} f(\mathbf{v}, \alpha) + \frac{1}{2\gamma}\|\mathbf{v} - \mathbf{v}_0\|^2, \tag{4}$$

where γ is an appropriate regularization parameter to make sure that the regularized function is strongly-convex and strongly-concave. The reference point $\mathbf{v}_0$ is periodically updated after a number of iterations. At the s-th stage our algorithm invokes a communication-efficient algorithm for solving the above strongly-convex and strongly-concave subproblems. After obtaining a primal solution $\mathbf{v}_s$ at the s-th stage, we sample some data from individual machines to obtain an estimate of the corresponding dual variable α_s.

Our new contribution is the communication-efficient distributed algorithm for solving the above strongly-convex and strongly-concave subproblems. The algorithm referred to as DSG is presented in Algorithm 2. Each machine makes a stochastic proximal-gradient update on the primal variable and a stochastic gradient update on the dual variable at each iteration. After every I iterations, all the K machines communicate to compute an average of local primal solutions $\mathbf{v}_t^k$ and local dual solutions α_t^k. It is not difficult to show that when $I = 1$, our algorithm reduces to the naive parallel version of the PPD-SG algorithm proposed in (Liu et al., 2020b), i.e., by averaging individual primal and dual gradients and then updating the primal-dual variables according to the averaged gradient [1]. Our novel analysis allows us to use $I > 1$ to skip communications, leading to a much less number of communications. The intuition behind this is that, as long as the step size η_s is sufficiently small we can control the distance between individual solutions $(\mathbf{v}_t^k, \alpha_t^k)$ to their global averages, which allows us to control the error

[1] A tiny difference is that we use a proximal gradient update to handle the regularizer $\frac{1}{2\gamma}\|\mathbf{v} - \mathbf{v}_0\|^2$, while Liu et al. (2020b) used the gradient update. Using the proximal gradient update allows us to remove the assumption that $\|\mathbf{v}_t^k - \mathbf{v}_0\|$ is upper bounded.

Algorithm 2 DSG($\mathbf{v}_0, \alpha_0, \eta, T, I, \gamma$)

Each machine does initialization: $\mathbf{v}_0^k = \mathbf{v}_0, \alpha_0^k = \alpha_0$,
for $t = 0, 1, ..., T-1$ **do**
  Each machine k updates its local solution in parallel:

$$\mathbf{v}_{t+1}^k = \arg\min_{\mathbf{v}} \Big[\nabla_{\mathbf{v}} F_k(\mathbf{v}_t^k, \alpha_t^k; \mathbf{z}_t^k)^T \mathbf{v} + \tfrac{1}{2\eta}\|\mathbf{v} - \mathbf{v}_t^k\|^2 + \tfrac{1}{2\gamma}\|\mathbf{v} - \mathbf{v}_0\|^2\Big],$$

$$\alpha_{t+1}^k = \alpha_t^k + \eta \nabla_\alpha F_k(\mathbf{v}_t^k, \alpha_t^k; \mathbf{z}_t^k),$$

  if $t+1 \bmod I = 0$ **then**

$$\mathbf{v}_{t+1}^k = \frac{1}{K}\sum_{k=1}^K \mathbf{v}_{t+1}^k, \quad \diamond \text{ communicate}$$

$$\alpha_{t+1}^k = \frac{1}{K}\sum_{k=1}^K \alpha_{t+1}^k, \quad \diamond \text{ communicate}$$

  end if
end for
Return $\tilde{\mathbf{v}} = \frac{1}{K}\sum_{k=1}^K \frac{1}{T}\sum_{t=1}^T \mathbf{v}_t^k$.

term that is caused by the discrepancy between individual machines. We will provide more explanations as we present the analysis.

Below, we present the main theoretical result of CoDA. Note that in the following presentation, $L_{\mathbf{v}}, H, B, \sigma_{\mathbf{v}}, \sigma_\alpha$ are appropriate constants, whose values are given in the proofs of Lemma 1 and Lemma 2 in the supplement.

Theorem 1 *Set* $\gamma = \frac{1}{2L_{\mathbf{v}}}$, $c = \frac{\mu/L_{\mathbf{v}}}{5+\mu/L_{\mathbf{v}}}$, $\eta_s = \eta_0 K \exp(-(s-1)c) \leq O(1)$, $T_s = \frac{\max(8, 8G_h^2)}{L_{\mathbf{v}}\eta_0 K}\exp((s-1)c)$, $I_s = \max(1, 1/\sqrt{K\eta_s})$ *and* $m_s = \max\left(\frac{1+C}{\eta_{s+1}^2 T_{s+1} p^2(1-p)^2}, \frac{\log(K)}{\log(1/\tilde{p})}\right)$, *where* $C = \frac{3\tilde{p}^{\frac{1}{\ln(1/\tilde{p})}}}{2\ln(1/\tilde{p})}$ *and* $\tilde{p} = \max(p, 1-p)$. *To return* $\mathbf{v}_S$ *such that* $\mathbb{E}[\phi(\mathbf{v}_S) - \phi(\mathbf{v}_\phi^*)] \leq \epsilon$, *it suffices to choose* $S \geq \frac{5L_{\mathbf{v}}+\mu}{\mu}\max\left\{\log\left(\frac{2\Delta_0}{\epsilon}\right), \log S + \log\left[\frac{2\eta_0}{\epsilon}\frac{6HB^2+12(\sigma_{\mathbf{v}}^2+\sigma_\alpha^2)}{5}\right]\right\}$. *As a result, the number of iterations is at most* $T = \widetilde{O}\left(\max\left(\frac{\Delta_0}{\mu\epsilon\eta_0 K}, \frac{L_{\mathbf{v}}}{\mu^2 K\epsilon}\right)\right)$ *and the number of communications is at most* $\widetilde{O}\left(\max\left(\frac{K}{\mu} + \frac{\Delta_0^{1/2}}{\mu(\eta_0\epsilon)^{1/2}}, \frac{K}{\mu} + \frac{L_{\mathbf{v}}^{1/2}}{\mu^{3/2}\epsilon^{1/2}}\right)\right)$, *where* $\widetilde{O}$ *suppresses logarithmic factors, and* $H, B, \sigma_{\mathbf{v}}, \sigma_\alpha$ *are appropriate constants.*

We have the following remarks about Theorem 1.

- First, we can see that the step size η_s is reduced geometrically in a stagewise manner. This is due to the PL condition. We note that a stagewise geometrically decreasing step size is usually used in practice in deep learning (Yuan et al., 2019). Second, by setting $\eta_0 = O(1/K)$ we have $I_s = \Theta(\frac{1}{\sqrt{K}}\exp((s-1)c/2)$. It means two things: (i) the larger the number of machines the smaller value of I_s, i.e., more frequently the machines need to communicate. This is reasonable since more machines will create a larger discrepancy between data among different machines; (ii) the value I_s can be increased geometrically across stages. This is because that the step size η_s is reduced geometrically, which causes one step of primal and dual updates on individual machines to diverge less from their averaged solutions. As a result, more communications can be skipped.
- Second, we can see that when $K \leq \Theta(1/\mu)$, we have the total iteration complexity given by $\widetilde{O}(\frac{1}{\mu^2 K\epsilon})$. Compared with the iteration complexity of the PPD-SG algorithm proposed in (Liu et al., 2020b) that is $\widetilde{O}(\frac{1}{\mu^2\epsilon})$, the proposed algorithm CoDA enjoys an iteration complexity that is reduced by a factor of K. This means that up to a certain large threshold $\Theta(1/\mu)$ for the number K of machines, CoDA enjoys a linear speedup.
- Finally, let us compare CoDA with the naive parallel version of PPD-SG, which is CoDA by setting $I = 1$. In fact, our analysis of the iteration complexity for this case is still applicable, and it is not difficult to show that the iteration complexity of the naive parallel version of PPD-SG is given by $\widetilde{O}(\frac{1}{\mu^2 K\epsilon})$ when $K \leq 1/\mu$. As a result, its communication complexity is also $\widetilde{O}(\frac{1}{\mu^2 K\epsilon})$. In contrast, CoDA's communication complexity is $\widetilde{O}(\frac{1}{\mu^{3/2}\epsilon^{1/2}})$ when $K \leq \frac{1}{\mu} \leq \frac{1}{\mu^{1/2}\epsilon^{1/2}} \leq \frac{1}{\epsilon}$ according to Theorem 1 [2]. Hence, our algorithm is more communication efficient, i.e., $\widetilde{O}(\frac{1}{\mu^{3/2}\epsilon^{1/2}}) \leq \widetilde{O}(\frac{1}{\mu^2 K\epsilon})$ when $K \leq \frac{1}{\mu}$ [3]. This means that up to a certain large threshold $\Theta(1/\mu)$ for the number K of machines, CoDA has a smaller communication complexity than the naive parallel version of PPD-SG.

4.1. Analysis

Below, we present a sketch of the proof of Theorem 1 by providing some key lemmas. We first derive some useful properties regarding the random function $F_k(\mathbf{v}, \alpha, \mathbf{z})$.

Lemma 1 *Suppose that Assumption 1 holds and* $\eta \leq \min(\frac{1}{2p(1-p)}, \frac{1}{2(1-p)}, \frac{1}{2p})$. *Then there exist some constants* $L_2, B_\alpha, B_{\mathbf{v}}, \sigma_{\mathbf{v}}, \sigma_\alpha$ *such that*

$$\|\nabla_{\mathbf{v}} F_k(\mathbf{v}_1, \alpha; \mathbf{z}) - \nabla_{\mathbf{v}} F_k(\mathbf{v}_2, \alpha; \mathbf{z})\| \leq L_2\|\mathbf{v}_1 - \mathbf{v}_2\|,$$
$$\|\nabla_{\mathbf{v}} F_k(\mathbf{v}, \alpha; \mathbf{z})\|^2 \leq B_{\mathbf{v}}^2, |\nabla_\alpha F_k(\mathbf{v}, \alpha; \mathbf{z})|^2 \leq B_\alpha^2,$$
$$\mathbb{E}[\|\nabla_{\mathbf{v}} f_k(\mathbf{v}, \alpha) - \nabla_{\mathbf{v}} F_k(\mathbf{v}, \alpha; \mathbf{z})\|^2] \leq \sigma_{\mathbf{v}}^2,$$
$$\mathbb{E}[|\nabla_\alpha f_k(\mathbf{v}, \alpha) - \nabla_\alpha F_k(\mathbf{v}, \alpha; \mathbf{z})|] \leq \sigma_\alpha^2.$$

[2] Assume that ϵ is set to be smaller than μ.

[3] Indeed, K can be as large as $\frac{1}{\mu^{1/2}\epsilon^{1/2}}$ for CoDA to be more communication-efficient.

Remark: We include the proofs of these properties in the Appendix. In the following, we will denote $B^2 = \max(B_{\mathbf{v}}^2, B_\alpha^2)$ and $L_{\mathbf{v}} = \max(L_1, L_2)$.

Next, we introduce a key lemma, which is of vital importance to establish the upper bound of the objective gap of the regularized subproblem.

Lemma 2 *(One call of Algorithm 2) Let* $\psi(\mathbf{v}) = \max_\alpha f(\mathbf{v}, \alpha) + \frac{1}{2\gamma}\|\mathbf{v} - \mathbf{v}_0\|^2$, $\tilde{\mathbf{v}}$ *be the output of Algorithm 2, and* $\mathbf{v}_\psi^* = \arg\min \psi(\mathbf{v})$, $\alpha^*(\tilde{\mathbf{v}}) = \arg\max_\alpha f(\tilde{\mathbf{v}}, \alpha) + \frac{1}{2\gamma}\|\tilde{\mathbf{v}} - \mathbf{v}_0\|^2$. *By running Algorithm 2 with given input* $\mathbf{v}_0, \alpha_0$ *for* T *iterations,* $\gamma = \frac{1}{2L_{\mathbf{v}}}$, *and* $\eta \leq \min\{\frac{1}{L_{\mathbf{v}}+3G_\alpha^2/\mu_\alpha}, \frac{1}{L_\alpha+3G_{\mathbf{v}}^2/L_{\mathbf{v}}}, \frac{3}{2\mu_\alpha}, \frac{1}{2p(1-p)}, \frac{1}{2(1-p)}, \frac{1}{2p}\}$, *we have*

$$\mathbb{E}[\psi(\tilde{\mathbf{v}}) - \min_{\mathbf{v}} \psi(\mathbf{v})] \leq \frac{2\|\mathbf{v}_0 - \mathbf{v}_\psi^*\|^2 + \mathbb{E}[(\alpha_0 - \alpha^*(\tilde{\mathbf{v}}))^2]}{\eta T} + H\eta^2 I^2 B^2 \mathbb{I}_{I>1} + \frac{\eta(2\sigma_{\mathbf{v}}^2 + 3\sigma_\alpha^2)}{2K},$$

where $\mu_\alpha = 2p(1-p)$, $L_\alpha = 2p(1-p)$, $G_\alpha = 2\max\{p, 1-p\}$, $G_{\mathbf{v}} = 2\max\{p, 1-p\}G_h$, *and* $H = \left(\frac{6G_{\mathbf{v}}^2}{\mu_\alpha} + 6L_{\mathbf{v}} + \frac{6G_\alpha^2}{L_{\mathbf{v}}} + \frac{6L_\alpha^2}{\mu_\alpha}\right)$.

Remark: The above result is similar to Lemma 2 in (Liu et al., 2020b). The key difference lies in the second and third terms in the upper bound. The second term arises because of the discrepancy of updates between individual machines. The third term is due to the variance reduction by using multiple machines, which is the key to establish the linear speed-up. It is easy to see that by setting $I = \frac{1}{\sqrt{\eta K}}$, the second term and the third term have the same order. With the above lemma, the proof of Theorem 1 follows similar analysis to in (Liu et al., 2020b).

Sketch of the Proof of Lemma 2. Below, we present a roadmap for the proof of the key Lemma 2. The main idea is to first bound the objective gap of the subproblem in Lemma 3. Then we further bound every term in the RHS in Lemma 3 appropriately, which is realized by Lemma 4, Lemma 5 and Lemma 6. All the detailed proofs of Lemmas can be found in Appendix.

Lemma 3 *Define* $\bar{\mathbf{v}}_t = \frac{1}{K}\sum_{k=1}^K \mathbf{v}_t^k, \bar{\alpha}_t = \frac{1}{K}\sum_{k=1}^K \alpha_t^k$. *Suppose Assumption 1 holds and by running Algorithm 2, we have*

$$\begin{aligned}
&\psi(\tilde{\mathbf{v}}) - \min_{\mathbf{v}} \psi(\mathbf{v}) \\
&\leq \frac{1}{T}\sum_{t=1}^T \Bigg[\underbrace{\langle \nabla_{\mathbf{v}} f(\bar{\mathbf{v}}_{t-1}, \bar{\alpha}_{t-1}), \bar{\mathbf{v}}_t - \mathbf{v}_\psi^*\rangle + 2L_{\mathbf{v}}\langle \bar{\mathbf{v}}_t - \mathbf{v}_0, \bar{\mathbf{v}}_t - \mathbf{v}_\psi^*\rangle}_{A_1} \\
&+ \underbrace{\langle \nabla_\alpha f(\bar{\mathbf{v}}_{t-1}, \bar{\alpha}_{t-1}), \alpha^* - \bar{\alpha}_t\rangle}_{A_2} \\
&+ \underbrace{\frac{L_{\mathbf{v}} + 3G_\alpha^2/\mu_\alpha}{2}\|\bar{\mathbf{v}}_t - \bar{\mathbf{v}}_{t-1}\|^2 + \frac{L_\alpha + 3G_{\mathbf{v}}^2/L_{\mathbf{v}}}{2}(\bar{\alpha}_t - \bar{\alpha}_{t-1})^2}_{A_3} \\
&+ \frac{2L_{\mathbf{v}}}{3}\|\bar{\mathbf{v}}_{t-1} - \mathbf{v}_\psi^*\|^2 - L_{\mathbf{v}}\|\bar{\mathbf{v}}_t - \mathbf{v}_\psi^*\|^2 - \frac{\mu_\alpha}{3}(\bar{\alpha}_{t-1} - \alpha^*)^2\Bigg].
\end{aligned}$$

Next, we will bound A_1, A_2 in Lemma 4 and Lemma 5. A_3 can be cancelled with similar terms in the following two lemmas. The remaining terms will be left to form a telescoping sum with other similar terms in the following two lemmas.

Lemma 4 *Define* $\hat{\mathbf{v}}_t = \arg\min_{\mathbf{v}} \left(\frac{1}{K}\sum_{k=1}^K \nabla_{\mathbf{v}} f(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k)\right)^T \mathbf{v} + \frac{1}{2\eta}\|\mathbf{v} - \bar{\mathbf{v}}_{t-1}\|^2 + \frac{1}{2\gamma}\|\mathbf{v} - \mathbf{v}_0\|^2$. *We have*

$$\begin{aligned}
A_1 \leq{}& \frac{3G_\alpha^2}{2L_{\mathbf{v}}}\frac{1}{K}\sum_{k=1}^K(\bar{\alpha}_{t-1} - \alpha_{t-1}^k)^2 + \frac{3L_{\mathbf{v}}}{2}\frac{1}{K}\sum_{k=1}^K\|\bar{\mathbf{v}}_{t-1} - \mathbf{v}_{t-1}^k\|^2 \\
&+ \eta\left\|\frac{1}{K}\sum_{k=1}^K[\nabla_{\mathbf{v}} f_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k) - \nabla_{\mathbf{v}} F_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k; \mathbf{z}_{t-1}^k)]\right\|^2 \\
&+ \frac{1}{K}\sum_{k=1}^K\langle \nabla_{\mathbf{v}} f_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k) - \nabla_{\mathbf{v}} F_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k; \mathbf{z}_{t-1}^k), \hat{\mathbf{v}}_t - \mathbf{v}_\psi^*\rangle \\
&+ \frac{1}{2\eta}(\|\bar{\mathbf{v}}_{t-1} - \mathbf{v}_\psi^*\|^2 - \|\bar{\mathbf{v}}_{t-1} - \bar{\mathbf{v}}_t\|^2 - \|\bar{\mathbf{v}}_t - \mathbf{v}_\psi^*\|^2) + \frac{L_{\mathbf{v}}}{3}\|\bar{\mathbf{v}}_t - \mathbf{v}_\psi^*\|^2.
\end{aligned}$$

Lemma 5 *Define* $\hat{\alpha}_t = \bar{\alpha}_{t-1} + \frac{\eta}{K}\sum_{k=1}^K \nabla_\alpha f_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k)$, *and*

$$\tilde{\alpha}_t = \tilde{\alpha}_{t-1} + \frac{\eta}{K}\sum_{k=1}^K(\nabla_\alpha F_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k; \mathbf{z}_{t-1}^k) - \nabla_\alpha f_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k)).$$

We have,

$$\begin{aligned}
A_2 \leq{}& \frac{3G_{\mathbf{v}}^2}{2\mu_\alpha}\frac{1}{K}\sum_{k=1}^K\|\bar{\mathbf{v}}_{t-1} - \mathbf{v}_{t-1}^k\|^2 + \frac{3L_\alpha^2}{2\mu_\alpha}\frac{1}{K}\sum_{k=1}^K(\bar{\alpha}_{t-1} - \alpha_{t-1}^k)^2 \\
&+ \frac{3\eta}{2}(\frac{1}{K}\sum_{k=1}^K[\nabla_\alpha f_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k) - \nabla_\alpha F_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k; \mathbf{z}_{t-1})])^2 \\
&+ \frac{1}{K}\sum_{k=1}^K\langle \nabla_\alpha f_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k) - \nabla_\alpha F_k(\mathbf{v}_{t-1}^k, \alpha_{t-1}^k; \mathbf{z}_{t-1}^k), \tilde{\alpha}_{t-1} - \hat{\alpha}_t\rangle \\
&+ \frac{1}{2\eta}((\bar{\alpha}_{t-1} - \alpha^*(\tilde{\mathbf{v}}))^2 - (\bar{\alpha}_{t-1} - \bar{\alpha}_t)^2 - (\bar{\alpha}_t - \alpha^*(\tilde{\mathbf{v}}))^2) \\
&+ \frac{\mu_\alpha}{3}(\bar{\alpha}_t - \alpha^*(\tilde{\mathbf{v}}))^2 + \frac{1}{2\eta}((\alpha^*(\tilde{\mathbf{v}}) - \tilde{\alpha}_t)^2 - (\alpha^*(\tilde{\mathbf{v}}) - \tilde{\alpha}_{t+1})^2).
\end{aligned}$$

The first two terms in the upper bounds of A_1, A_2 are the differences between individual solutions and their averages, the third term is the variance of stochastic gradient, and the expectation of the fourth term will diminish. The lemma below will bound the difference between the averaged solution and the individual solutions.

Lemma 6 *If* K *machines communicate every* I *iterations, and update with step size* η, *then*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|\bar{\mathbf{v}}_t - \mathbf{v}_t^k\|^2] \leq 4\eta^2 I^2 B_{\mathbf{v}}^2 \mathbb{I}_{I>1}$$
$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|\bar{\alpha}_t - \alpha_t^k\|^2] \leq 4\eta^2 I^2 B_{\alpha}^2 \mathbb{I}_{I>1}.$$

Combining the results in Lemma 3, Lemma 4, Lemma 5 and Lemma 6, we can prove the key Lemma 2.

5. Experiments

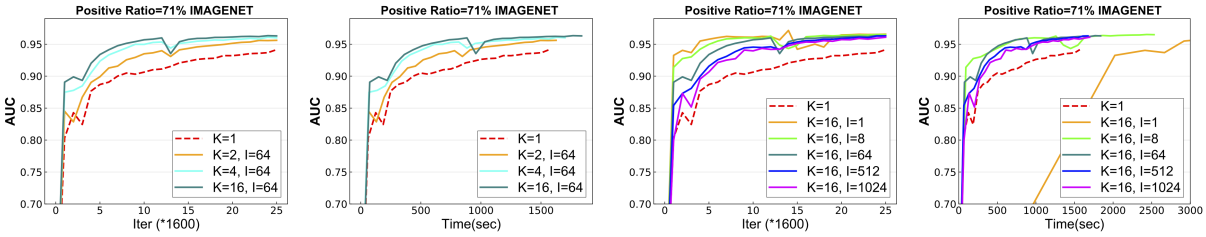
In this section, we conduct some experiments to verify our theory. In our experiments, one "machine" corresponds to one GPU. We use a cluster of 4 computing nodes with each computer node having 4 GPUs, which gives a total of 16 "machines". We would like to emphasize that even though 4 GPUs sit on one computing node, they only access to different parts of the data. For the experiment with $K = 1$ GPU, We run one computing node by using one GPU. For experiments with $K = 4$ GPUs, we run one computing node by using all four GPUs, and for those experiments with $K = 16$ GPUs, we use four computing nodes by using all GPUs. We notice that the communication costs among GPUs on one computing node may be less than that among GPUs on different computing nodes. Hence, it should be kept in mind that when comparing with $K = 4$ GPUs on different computing nodes, the margin of using $K = 16$ GPUs over using $K = 4$ GPUs should be larger than what we will see in our experimental results. All algorithms are implemented by PyTorch (Paszke et al., 2019).

Data. We do experiments on 3 datasets: Cifar10, Cifar100 and ImageNet. For Cifar10, we split the original training data into two classes, i.e., the positive class contains 5 original classes and the negative class is composed of the other 5 classes. The Cifar100 dataset is split in a similar way, i.e., the positive class contains 50 original classes and the negative class is composed of the other 50 classes. Testing data for Cifar10 and Cifar100 is the same as the original dataset. For the ImageNet dataset, we sample 1% of the original training data as testing data and use the remaining data as the training data. The training data is split in a similar way to Cifar10 and Cifar100, i.e., the positive class contains 500 original classes and the negative class is composed of the other 500 classes. For each dataset, we create two versions of training data with different positive ratios. By keeping all data in the positive and negative classes, we have $p = 50\%$ for all three datasetes. To create imbalanced data, we drop some proportion of the negative data for each dataset and keep all the positive examples. In particular, by keeping all the positive data and 40% of the negative data we construct three datasets with positive ratio $p = 71\%$. Training data is shuffled and evenly divided to each GPU, i.e., each GPU has access to $1/K$ of the training data, where K is the number of GPUs. For all data, we use ResNet50 as our neural network (He et al., 2016) and initialize the model as the pretrained model from PyTorch. Due to the limit of space, we only report the results on datasets with $p = 71\%$ positive ratio, and other results are included in the supplement.

Baselines and Parameter Setting. For baselines, we compare with the single-machine algorithm PPD-SG as proposed in (Liu et al., 2020b), which is represented by $K = 1$ in our results, and the naive parallel version of PPD-SG, which is denoted by $K = \text{X}$, $I = 1$ in our results. For all algorithms, we set $T_s = T_0 3^k$, $\eta_s = \eta_0/3^k$. T_0 and η_0 are tuned for PPD-SG and set to the same for all other algorithms for fair comparison. T_0 is tuned in $[2000, 5000, 10000]$, and η_0 is tuned in $[0.1, 0.01, 0.001]$. We fix the batch size for each GPU as 32. For simplicity, in our experiments we use a fixed value of I in order to see its tradeoff with the number of machines K.

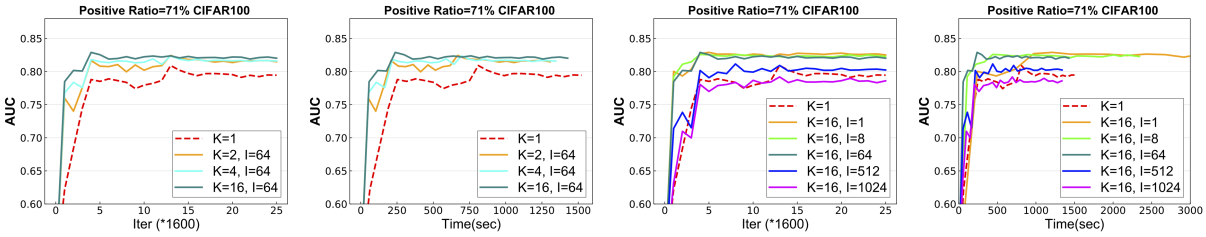
Results. We plot the curve of testing AUC versus the number of iterations and versus running time. We notice that evaluating the training objective function value on all examples is very expensive, so we use the testing AUC as our evaluation metric. It may cause some gap between our results and the theory; however, the trend should be enough for our purpose to verify that our distributed algorithms can enjoy faster convergence in both the number of iterations and running time. We have the following observations.

- **Varying K.** By varying K and fixing the value of I, we aim to verify the parallel speedup. The results are shown in Figure 1(a), Figure 2(a) and Figure 3(a). They show that when K becomes larger, then our algorithm requires a less number of iterations to converge to the target AUC, which is consistent with the parallel speedup result as indicated by Theorem 1. In addition, CoDA with $K = 16$ machines is also the most time-efficient algorithm among all settings.
- **Varying I.** By varying I and fixing the value of K, we aim to verify that skipping communications up to a certain number of iterations of CoDA does not hurt the iteration complexity but can dramatically reduce the total communication costs. In particular, we fix $K = 16$ and vary I in the range $\{1, 8, 64, 512, 1024\}$. The results are shown in Figures 1(b), Figures 2(b) and Figures 3(b). They exhibit that even when I becomes moderately large, our algorithm is still able to deliver comparable performance in terms of the number of iterations compared with the case when $I = 1$. The largest value of I that does not cause a dramatic performance drop compared with $I = 1$ is $I = 1024$, $I = 64$, $I = 64$ on ImageNet, CIFAR100 and CIFAR10, respectively. However, up to these thresholds the running time of CoDA can be dramatically reduced than the naive parallel version with $I = 1$.
- **Trade-off between I and K.** Finally, we verify the trade-off between I and K as indicated in Theorem 1.



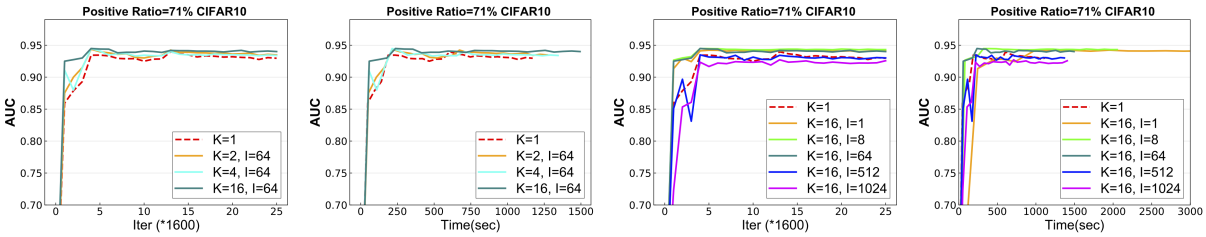
(a) Fix I, vary K (b) Fix K, vary I

Figure 1. ImageNet, positive ratio = 71%.



(a) Fix I, vary K (b) Fix K, vary I

Figure 2. Cifar100, positive ratio = 71%.



(a) Fix I, vary K (b) Fix K, vary I

Figure 3. Cifar10, positive ratio = 71%.

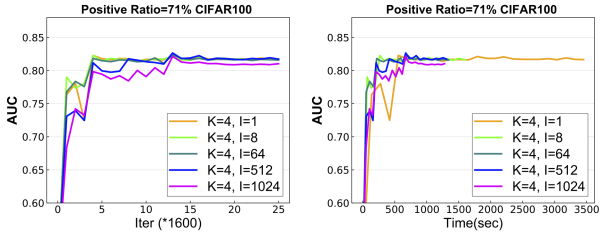


Figure 4. Cifar100, positive ratio = 71%, K=4.

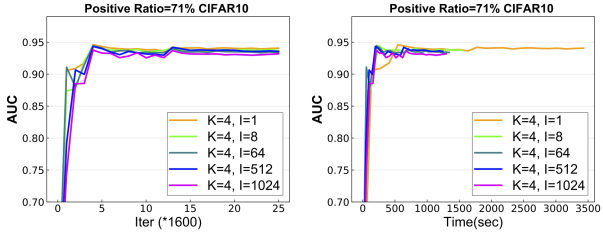


Figure 5. Cifar10, positive ratio = 71%, K=4.

To this end, we conduct experiments by fixing $K = 4$ GPUs and varying the value I, and comparing the limits of I for $K = 4$ and $K = 16$. The results of using $K = 4$ on CIFAR100 and CIFAR10 are reported in Figure 4 and Figure 5. We can observe that when $K = 4$ the upper limit of I that does not cause a dramatic performance drop compared with $I = 1$ is $I = 512$ for the two datasets, which is larger than the upper limit of $I = 64$ for $K = 16$. This is consistent with our Theorem 1.

6. Conclusion

In this paper, we have designed a communication-efficient distributed stochastic deep AUC maximization algorithm, in which each machine is able to do multiple iterations of local updates before communicating with the central node. We have proven the linear speedup property and shown that the communication complexity can be dramatically reduced for multiple machines up to a large threshold number. Our empirical studies verify the theory and also demonstrate the effectiveness of the proposed distributed algorithm on benchmark datasets.

Acknowledgements

This work is partially supported by National Science Foundation CAREER Award 1844403 and National Science Foundation Award 1933212.

References

Basu, D., Data, D., Karakus, C., and Diggavi, S. Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, pp. 14668–14679, 2019.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Chen, K. and Huo, Q. Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering. In *2016 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 5880–5884. IEEE, 2016.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pp. 1223–1231, 2012.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Elkan, C. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pp. 973–978. Lawrence Erlbaum Associates Ltd, 2001.

Gao, W., Jin, R., Zhu, S., and Zhou, Z.-H. One-pass auc optimization. In *ICML (3)*, pp. 906–914, 2013.

Godichon-Baggioni, A. and Saadane, S. On the rates of convergence of parallelized averaged stochastic gradient algorithms. *arXiv preprint arXiv:1710.07926*, 2017.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Haddadpour, F., Kamani, M. M., Mahdavi, M., and Cadambe, V. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. In *Advances in Neural Information Processing Systems*, pp. 11080–11092, 2019.

Hanley, J. A. and McNeil, B. J. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

Hanley, J. A. and McNeil, B. J. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, 1983.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jaggi, M., Smith, V., Takác, M., Terhorst, J., Krishnan, S., Hofmann, T., and Jordan, M. I. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3068–3076, 2014.

Jain, P., Netrapalli, P., Kakade, S. M., Kidambi, R., and Sidford, A. Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification. *The Journal of Machine Learning Research*, 18(1):8258–8299, 2017.

Jiang, P. and Agrawal, G. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Advances in Neural Information Processing Systems*, pp. 2525–2536, 2018.

Jin, C., Netrapalli, P., and Jordan, M. I. Minmax optimization: Stable limit points of gradient descent ascent are locally optimal. *arXiv preprint arXiv:1902.00618*, 2019.

Kamp, M., Adilova, L., Sicking, J., Hüger, F., Schlicht, P., Wirtz, T., and Wrobel, S. Efficient decentralized deep learning by dynamic model averaging. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 393–409. Springer, 2018.

Koloskova, A., Stich, S. U., and Jaggi, M. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 3478–3487, 2019.

Koloskova*, A., Lin*, T., Stich, S. U., and Jaggi, M. Decentralized deep learning with arbitrary communication compression. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgGCkrKvH.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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 *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pp. 583–598, 2014.

Lin, Q., Liu, M., Rafique, H., and Yang, T. Solving weakly-convex-weakly-concave saddle-point problems as weakly-monotone variational inequality. *arXiv preprint arXiv:1810.10207*, 2018a.

Lin, T., Stich, S. U., Patel, K. K., and Jaggi, M. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217*, 2018b.

Lin, T., Jin, C., and Jordan, M. I. On gradient descent ascent for nonconvex-concave minimax problems. *arXiv preprint arXiv:1906.00331*, 2019.

Liu, M., Zhang, X., Chen, Z., Wang, X., and Yang, T. Fast stochastic auc maximization with o (1/n)-convergence rate. In *International Conference on Machine Learning*, pp. 3195–3203, 2018.

Liu, M., Mroueh, Y., Ross, J., Zhang, W., Cui, X., Das, P., and Yang, T. Towards better understanding of adaptive gradient algorithms in generative adversarial nets. In *International Conference on Learning Representations*, 2020a. URL https://openreview.net/forum?id=SJxIm0VtwH.

Liu, M., Yuan, Z., Ying, Y., and Yang, T. Stochastic auc maximization with deep neural networks. *ICLR*, 2020b.

Lu, S., Tsaknakis, I., Hong, M., and Chen, Y. Hybrid block successive approximation for one-sided non-convex min-max problems: Algorithms and applications. *arXiv preprint arXiv:1902.08294*, 2019.

McDonald, R., Hall, K., and Mann, G. Distributed training strategies for the structured perceptron. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*, pp. 456–464. Association for Computational Linguistics, 2010.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Natole, M., Ying, Y., and Lyu, S. Stochastic proximal algorithms for auc maximization. In *International Conference on Machine Learning*, pp. 3707–3716, 2018.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Nesterov, Y. E. *Introductory Lectures on Convex Optimization - A Basic Course*, volume 87 of *Applied Optimization*. Springer, 2004.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Povey, D., Zhang, X., and Khudanpur, S. Parallel training of dnns with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*, 2014.

Rafique, H., Liu, M., Lin, Q., and Yang, T. Non-convex min-max optimization: Provable algorithms and applications in machine learning. *arXiv preprint arXiv:1810.02060*, 2018.

Rockafellar, R. T. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

Sanjabi, M., Razaviyayn, M., and Lee, J. D. Solving non-convex non-concave min-max games under polyak-l ojasiewicz condition. *arXiv preprint arXiv:1812.02878*, 2018.

Shamir, O. and Srebro, N. Distributed stochastic optimization and learning. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 850–857. IEEE, 2014.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pp. 4447–4458, 2018.

Su, H. and Chen, H. Experiments on parallel training of deep neural network using model averaging. *arXiv preprint arXiv:1507.01239*, 2015.

Wang, J. and Joshi, G. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd. *arXiv preprint arXiv:1810.08313*, 2018a.

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*, 2018b.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1299–1309, 2018.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5754–5764, 2019.

Ying, Y., Wen, L., and Lyu, S. Stochastic online auc maximization. In *Advances in Neural Information Processing Systems*, pp. 451–459, 2016.

Yu, H., Jin, R., and Yang, S. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. *arXiv preprint arXiv:1905.03817*, 2019a.

Yu, H., Yang, S., and Zhu, S. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5693–5700, 2019b.

Yuan, Z., Yan, Y., Jin, R., and Yang, T. Stagewise training accelerates convergence of testing error over SGD. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 2604–2614, 2019.

Zhang, Y., Duchi, J. C., and Wainwright, M. J. Communication-efficient algorithms for statistical optimization. *The Journal of Machine Learning Research*, 14 (1):3321–3363, 2013.

Zhao, P., Jin, R., Yang, T., and Hoi, S. C. Online auc maximization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 233–240, 2011.

Zhou, F. and Cong, G. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. *arXiv preprint arXiv:1708.01012*, 2017.

Zinkevich, M., Weimer, M., Li, L., and Smola, A. J. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pp. 2595–2603, 2010.