Asynchronous Training Schemes in Distributed Learning with Time Delay

Anonymous Author(s)
Affiliation
Address
email

Abstract

In the context of distributed deep learning, the issue of stale weights or gradients could result in poor algorithmic performance. This issue is usually tackled by delay tolerant algorithms with some mild assumptions on the objective functions and step sizes. In this paper, we propose a different approach to develop a new algorithm, called **P**redicting **C**lipping **A**synchronous **S**tochastic **G**radient **D**escent (aka, PC-ASGD). Specifically, PC-ASGD has two steps - the *predicting step* leverages the gradient prediction using Taylor expansion to reduce the staleness of the outdated weights while the *clipping step* selectively drops the outdated weights to alleviate their negative effects. A tradeoff parameter is introduced to balance the effects between these two steps. Theoretically, we present the convergence rate considering the effects of delay of the proposed algorithm with constant step size when the smooth objective functions are weakly strongly convex and nonconvex. One practical variant of PC-ASGD is also proposed by adopting a condition to help with the determination of the tradeoff parameter. For empirical validation, we demonstrate the performance of the algorithm with two deep neural network architectures on two benchmark datasets.

1 Introduction

The availability of large data sets and powerful computing led to the emergence of deep learning that is revolutionizing many application sectors from the internet industry and healthcare to transportation and energy [1]. As the applications are scaling up, the learning process of large deep learning models is looking to leverage emerging resources such as edge computing and distributed data centers privacy preserving. In this regard, distributed deep learning algorithms are being explored by the community that leverage synchronous and asynchronous computations with multiple computing agents that exchange information over communication networks [2]. We consider an example setting involving an industrial IoT framework where the data is geographically distributed as well as the computing resources. While the computing resources within a local cluster can operate in a (loosely) synchronous manner, multiple (geographically distributed) clusters may need to operate in an asynchronous manner. Furthermore, communications among the computing resources may not be reliable and prone to delay and loss of information.

Among various distributed deep learning algorithms, Federated Averaging and its variants are considered to be the state-of-the-art for training deep learning models with data distributed among the edge computing resources such as smart phones and idle computers [3]. The master-slave and peer-to-peer are two categories of distributed learning architectures. Along with Federated Averaging, variants such as PySyft [4] and its robust version [5] and the scalable distributed DNN training algorithms [6] and more recent distributed SVRG [7] are examples of the master-slave architecture. On the other hand, examples of the peer-to-peer architecture include the gossip algorithms [8, 9], and the collaborative learning frameworks [10].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

However, as mentioned earlier, communication delay remains a critical challenge for achieving convergence in an asynchronous learning setting [11, 12] and influences the performances of the frameworks above. Furthermore, the amount of delay could be varying widely due to artifacts of wireless communication and different devices. To tackle the influence of varying delays on the convergence characteristics of distributed learning algorithms, this work proposes a novel algorithm, called **P**redicting **C**lipping **A**synchronous **S**tochastic **G**radient **D**escent (aka, PC-ASGD). The goal is to solve the distributed learning problems involving multiple computing or edge devices such as GPUs and CPUs with varying communication delays among them. Different from traditional distributed learning scenarios where synchronous and asynchronous algorithms are considered separately, we take both into account together in a networked setting.

Related work. In the early works on distributed learning with master-slave architecture, Asynchronous Stochastic Gradient Descent (ASGD) algorithm has been adopted [13], where each local worker continues its training process right after its gradient is added to the global model. The algorithm could tolerate the delay in communication. Later works [14, 15, 16] extend ASGD to more realistic scenarios and implement the algorithms with a central server and other parallel workers. Typically, since asynchronous algorithms suffer from stale gradients, researchers have proposed algorithms such as DC-ASGD [17], adopting the concept of delay compensation to reduce the influence of staleness and improved the performance of ASGD. For the distributed learning with peer-to-peer architecture, [2] proposes the algorithm AD-PSGD (decentralized ASGD algorithm) that deals with the problem of the stale parameter exchange, as well as some theoretical analysis for the algorithm performance under bounded delay. [18] also proposes a similar algorithm with some variations in the assumptions. However, these algorithms do not provide empirical or theoretical analysis on the impacts of delay in detail. Additional works such as using a central agent for control [19], requiring prolonged communication [20], utilizing stochastic primal-dual method [21], and adopting importance sampling [22], have also been done to address the communication delay in the decentralized setting. More recently, [23] proposes the DC-s3gd algorithm to enable large-scale decentralized neural network training with the consideration of delay. [24], [25] and [26] also develop algorithms for asynchronous decentralized training for neural networks, while theoretical guarantee is not provided.

Table 1: Comparisons between asynchronous algorithms

Methods	f	∇f	Delay Ass.	Con.Rate	D.C.	G.C.	A.S.
ASGD [13]	Non-convex	Lip.	Bou.	$\mathcal{O}(\frac{1}{\sqrt{T}})$	No	No	No
DC-ASGD [17]	Str-con	Lip.	Bou.	$\mathcal{O}(\frac{1}{T})$	No	Yes	No
	Non-convex	Lip.	Bou.	$\mathcal{O}(\frac{1}{\sqrt{T}})$	No	Yes	No
AD-PSGD [2]	Non-convex	Lip.&Bou.	Bou.	$\mathcal{O}(\frac{1}{\sqrt{T}})$	Yes	No	No
DC-s3dg [23]	Non-convex	Lip.	Unbou.	N/A	Yes	Yes	No
PC-ASGD (This paper)	Weakly Str-con	Lip.	Bou.	$\mathcal{O}(\epsilon^T)$	Yes	Yes	Yes
	Non-convex	Lip.	Bou.	$\mathcal{O}(\frac{1}{T})$	Yes	Yes	Yes

Con.Rate: convergence rate, Str-con: strongly convex. Lip.& Bou.: Lipschitz continuous and bounded. Delay Ass.: Delay Assumption. Unbou.: Unbounded. T: Total iterations. D.C.: decentralized computation. G.C.: Gradient Compensation. A.S.: Alternant Step, $\epsilon \in (0, 1)$ is a positive constant. Note that the convergence rate of PC-ASGD is obtained by using the constant step size.

Contributions. The contributions of this work are specifically as follows. (i) A novel algorithm, called PC-ASGD for distributed learning is proposed to tackle the convergence issues due to the varying communication delays. Built upon ASGD, the PC-ASGD algorithm consists of two steps. While the predicting step leverages the gradient prediction using Taylor expansion to reduce the staleness of the outdated weights, the clipping step selectively drops the outdated weights to alleviate their negative effects. To balance the effects, a tradeoff parameter is introduced to combine these two steps. (ii) We show that with a proper constant step size, PC-ASGD can converge to the neighborhood of the optimal solution at a linear rate for weakly strongly convex functions while at a sublinear rate for nonconvex functions (specific comparisons with other related existing approaches are listed in Table 1). We also model the delay and take it into consideration in the convergence analysis. (iii) PC-ASGD is deployed on distributed GPUs with two datasets CIFAR-10 and CIFAR-100 by using PreResNet110 and DenseNet architectures. The proposed algorithm outperforms the existing delay tolerant algorithm as well as the variants of the proposed algorithm using only the predicting step or the clipping step.

2 Formulation and Preliminaries

Consider N agents in a networked system such that their interactions are driven by a graph $\mathcal{G}$, where $\mathcal{G} = \{V, E\}$, where $V = \{1, 2, .., N\}$ indicates the node or agent set, $E \subseteq V \times V$ is the edge set. Throughout the paper, we assume that the graph is undirected and connected. The connection between any two agents i and j can be determined by their physical connections, leading to the communication between them. Traditionally, if agent j is in the neighborhood of agent i, they can communicate with each other. Thus, we define the neighborhood for any agent i as $Nb(i) := \{j \in V|(i, j) \in E \text{ } or \text{ } j = i\}$. Rather than considering synchronization and asynchronization separately, this paper considers both scenarios together by defining the following terminologies.

Definition 1. At a time step t, an agent j is called a ***reliable neighbor*** of the agent i if agent i has the state information of agent j up to $t - 1$.

Definition 2. At a time step t, an agent j is called an ***unreliable neighbor*** of the agent i if agent i has the state information of agent j only up to $t - \tau$, where τ is the so-called *delay* and $1 < \tau < \infty$.

Definitions 1 and 2 allow us to perceive the delay problem in the decentralized learning with a new perspective that depends on the amount of delay. One agent can selectively make use of the outdated information from unreliable neighbors or completely drop such information. The first scenario is related to most previous works on asynchronous delay tolerant approaches as it involves a gradient prediction technique to reduce the negative effects of stale parameters. The second scenario corresponds to most synchronous schemes since the agent only collects information from the reliable neighbors. Thus, inside the neighborhood of an agent, there are reliable and unreliable neighbors respectively and this work aims at studying how to effectively tackle issues such as negative impacts that delays may bring on the performance. For analysis, we define a set for reliable neighbors of agent i as: $\mathcal{R} := \{j \in Nb(i) \mid p(x^j = x^j_{t-1}|t) = 1\}$, where $p(x^j = x^j_{t-1}|t) = 1$ is the probability, implying that agent j has the state information x up to the time $t - 1$, i.e., x^j_{t-1}. Then we can have the set for unreliable neighbors such that $\mathcal{R}^c = Nb \setminus \mathcal{R}$.

Note that the delay varies in the asynchronous learning scheme, and there are two types of asynchronization, (i) fixed value of delays [17, 23] and (ii) time-varying delays [13, 2] along the learning process. We follow the first setting in this work to implement the experiments. The definition and analysis can also be applicable for the later case when the delay τ changes to a time-varying vector.

Consider the decentralized empirical risk minimization problems, which can be expressed as the summation of all local losses incurred by each agent:

$$\min \;\; F(\mathbf{x}) := \sum_{i=1}^{N} \sum_{s \in \mathcal{D}_i} f_i^s(x) \tag{1}$$

where $\mathbf{x} = [x^1; x^2; ...; x^N]$, x_i is the local copy of $x \in \mathbb{R}^d$, $\mathcal{D}_i$ is a local data set uniquely known by agent i, $f_i^s : \mathbb{R}^d \to \mathbb{R}$ is the incurred local loss of agent i given a sample s. Based on the above formulation, we then assume everywhere that our objective function is bounded from below and denote the minimum by $F^* := F(\mathbf{x}^*)$ where $\mathbf{x}^* :=$ argmin $F(\mathbf{x})$. Hence $F^* > -\infty$. Moreover, all vector norms refer to the Euclidean norm while matrix norms refer to the Frobenius norm. Some necessary definitions and assumptions are given below for characterizing the main results.

Assumption 1. Each objective function f_i is assumed to satisfy the following conditions: a) f_i is $\gamma_i - smooth$; b) f_i is proper (not everywhere infinite) and coercive.

Assumption 2. A mixing matrix $\underline{W} \in \mathbb{R}^{N \times N}$ satisfies a) $\mathbf{1}^\top \underline{W} = \mathbf{1}^\top, \underline{W}\mathbf{1}^\top = \mathbf{1}^\top$; b) Null$\{I - \underline{W}\} =$ Span$\{\mathbf{1}\}$; c) $I \succeq \underline{W} \succ 0$.

Assumption 3. The stochastic gradient of F at any $\mathbf{x}$ is denoted by $\mathbf{g}(\mathbf{x})$, such that a) $\mathbf{g}(\mathbf{x})$ is the unbiased estimate of gradient $\nabla F(\mathbf{x})$; b) The variance is uniformly bounded by σ^2, i.e.,$\mathbb{E}[\|\mathbf{g}(\mathbf{x}) - \nabla F(\mathbf{x})\|^2] \leq \sigma^2$; c) The second moment of $\mathbf{g}(\mathbf{x})$ is bounded, i.e., $\mathbb{E}[\|\mathbf{g}(\mathbf{x})\|^2] \leq G^2$.

Given Assumption 1, one immediate consequence is that F is $\gamma_m :=$ max$\{\gamma_1, \gamma_2, ..., \gamma_N\}$-smooth at all $\mathbf{x} \in \mathbb{R}^{dN}$. The main outcome of Assumption 2 is that the mixing matrix $\underline{W}$ is doubly stochastic matrix and that we have $e_1(\underline{W}) = 1 > e_2(\underline{W}) \geq .. \geq e_N(\underline{W}) > 0$, where $e_z(\underline{W})$ denotes the z-th largest eigenvalue of $\underline{W}$. In Assumption 3, the first two are quite generic. While the third part is much weaker than the bounded gradient that is not necessarily applicable to quadratic-like objectives.

3 PC-ASGD

3.1 Algorithm Design

We present the specific update law for our proposed method, PC-ASGD in Algorithm 1. In Algorithm 1, for the predicting step (line 6), any agent k that is unreliable has delay when communicating its weights with agent i. To compensate for the delay, we adopt the Taylor expansion to approximate the gradient for each time step. The predicting gradient (or delay compensated gradient) is denoted by $g_k^{dc}(x_{t-\tau}^k)$, which is expressed as follows

$$g_k^{dc,r}(x_{t-\tau}^k) = \sum_{r=0}^{\tau-1} g_k(x_{t-\tau}^k) + \lambda g_k(x_{t-\tau}^k) \odot g_k(x_{t-\tau}^k) \odot (x_{t-\tau+r}^i - x_{t-\tau}^i), \tag{2}$$

where λ is a positive constant in $(0, 1]$ and the term $\lambda g_k(x_{t-\tau}^k) \odot g_k(x_{t-\tau}^k)$ is an estimate of the Hessian matrix, $\nabla g_k(x_{t-\tau}^k)$. Due to the limit of space, we omit the details of deriving Eq. 2, referring interested readers to the supplementary materials. We define another doubly stochastic matrix $\underline{\tilde{W}} \in \mathbb{R}^{N\times N}$ that follows Assumption 2 for the clipping step.

Algorithm 1 PC-ASGD

Input: number of agents N, learning rate $\eta > 0$, agent interaction matrices $\underline{W}$, $\underline{\tilde{W}}$, number of epochs T, the tradeoff parameter $0 \leq \theta_t \leq 1, t \in \{0, 1, ..., T-1\}$
Output: the models' parameters in agents x_T^i,$i = 1, 2, ...N$

1: **Initialize** all the agents' parameters x_0^i, $i = 1, 2, ...N$
2: Do broadcast to identify the clusters of reliable agents and the delay τ
3: $t = 0$
4: **while** $epoch\ t < T$ **do**
5: **for** each agent i **do**
6: Predicting Step: $x_{t+1,pre}^i = \sum_{j\in\mathcal{R}} w_{ij} x_t^j - \eta g_i(x_t^i) + \sum_{k\in\mathcal{R}^c} w_{ik}(x_{t-\tau}^k - \eta g_k^{dc}(x_{t-\tau}^k))$
7: Clipping Step: $x_{t+1,cli}^i = \sum_{j\in\mathcal{R}} \tilde{w}_{ij} x_t^j - \eta g_i(x_t^i)$
8: $x_{t+1}^i = \theta_t x_{t+1,pre}^i + (1-\theta_t) x_{t+1,cli}^i$
9: **end for**
10: $t = t + 1$
11: **end while**

Different from the DC-ASGD, which significantly relies on a central server to receive information from each agent, our work removes the dependence on the central server, and instead constructs a graph for all of agents. The clipping step (line 7) essentially rejects information from all the unreliable neighbor in the neighborhood of one agent. One observation can be made from the predicting and clipping steps is that the weights for consensus terms are different. In this context, for the clipping step, the weight values of some edges associated with the underlying static graph has been changed accordingly, but the connections of the graph still keep fixed. Subsequently, the equality in line 8 balances the tradeoff between the predicting and clipping steps. In practice, the determination of θ_t results in some practical variants. In the empirical study presented in Section 5, one can see that θ_t is either 0 or 1 by leveraging one condition, which implies that in each epoch, only one step is adopted. Additionally, θ_t can be fixed as 0 or 1, yielding two other variants shown in the experiments, C-ASGD or P-ASGD. However, for the sake of generalization, we provide the analysis for the combined steps (line 8). Before concluding this section, we give the compact form of the combination of PC steps and defer the detailed analysis to the supplementary materials.

Since the term $\sum_{k\in\mathcal{R}^c} w_{ik} \sum_{r=0}^{\tau-1} g_k^{dc,r}(x_t^k)$ applies to unreliable neighbors only, for the convenience of analysis, we expand it to the whole graph. It means that we establish an expanded graph to cover all of agents by setting some elements in the mixing matrix $\underline{W'} \in \mathbb{R}^{N\times N}$ equal to 0, but keeping the same connections as in $\underline{W}$. By setting the current time as $t + \tau$, the compact form in line 8 can be rewritten as:

$$\mathbf{x}_{t+\tau+1} = \mathcal{W}_{t+\tau}\mathbf{x}_{t+\tau} - \eta(\mathbf{g}(\mathbf{x}_{t+\tau}) + \theta_{t+\tau} \sum_{r=0}^{\tau-1} W' \mathbf{g}^{dc,r}(\mathbf{x}_t)) \tag{3}$$

$\mathcal{W}_{t+\tau}$ is denoted by $\theta_{t+\tau}W + (1-\theta_{t+\tau})\tilde{W}$, where $W = \underline{W} \otimes I_{d\times d}$, $\tilde{W} = \underline{\tilde{W}} \otimes I_{d\times d}$, and $W' = \underline{W}' \otimes I_{d\times d}$. Though the original graphs corresponding to the predicting and clipping steps are static, the equivalent graph $\mathcal{W}_{t+\tau}$ has become time-varying due to the time-varying θ value.

4 Convergence Analysis

This section presents convergence results for the PC-ASGD. We show the consensus estimate and the optimality for both weakly strongly convex and nonconvex smooth objectives. The consensus among agents (aka, disagreement estimate) can be thought of as the norms $\|x_t^i - x_t^j\|$, the differences between the iterates x_t^i and x_t^j. Alternatively, the consensus can be measured with respect to a reference sequence, i.e., $y_t = \frac{1}{N}\sum_{i=1}^{N} x_t^i$. In particular, we discuss $\|x_t^i - y_t\|$ for any time t as the metrics with respect to the delay τ.

Lemma 1. (***Consensus***) Let Assumptions 2 and 3 hold. Assume that the delay compensated gradients are uniformly bounded, i.e., there exists a scalar $B > 0$, such that

$$\|\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \leq B, \ \ \forall t \geq 0 \ and \ 0 \leq r \leq \tau - 1,$$

Then for all $i \in V$ and $t \geq 0$, $\exists \eta > 0$, we have

$$\mathbb{E}[\|x_t^i - y_t\|] \leq \eta \frac{G + (\tau-1)B\theta_m}{1-\delta_2}, \tag{4}$$

where $\theta_m = \max\{\theta_{s+1}\}_{s=t}^{t+\tau-1}$, $\delta_2 = \max\{\theta_s e_2 + (1-\theta_s)\tilde{e}_2\}_{s=0}^{t+\tau-1} < 1$, where $e_2 := e_2(W) < 1$ and $\tilde{e}_2 := e_2(\tilde{W}) < 1$.

The detailed proof is shown in the supplement materials. Lemma 1 states the consensus bound among agents, which is proportional to the step size η and inversely proportional to the gap between the largest and the second-largest magnitude eigenvalues of the equivalent graph $\mathcal{W}$.

Remark 1. One implication that can be made from Lemma 1 is when $\tau = 1$, the consensus bound becomes the smallest, which can be obtained as $\frac{\eta G}{1-\delta_2}$. This bound is the same as obtained already by most decentralized learning (or optimization) algorithms. This accordingly implies that the delay compensated gradient or predicting gradient does not necessarily require many time steps ahead prediction as more compounding error could be included. Alternatively, $\theta_m = 0$ can also result in such a bound, suggesting that the clipping step dominates in the update. On the other hand, once $\tau \gg 1$ and $\theta_m \neq 0$, the consensus bound becomes worse, which will be validated by the empirical results. Additionally, if the network is sparse, which suggests $e_2 \to 1$ and $\tilde{e}_2 \to 1$, the consensus among agents may not be achieved well and correspondingly the optimality would be negatively affected.

Most previous works have typically explored the convergence rate on the strongly convex objectives. However, the assumption of strong convexity can be a quite strong condition in most models such that the results obtained may be theoretically instructive and useful. Hence, we introduce a condition that is able to relax the strong convexity but still maintain the similar theoretical property, i.e., Polyak-Łojasiewicz (PL) condition [27]. The condition is expressed as follows: A differentiable function F satisfies the PL condition such that there exists a constant $\mu > 0$

$$\frac{1}{2}\|\nabla F(\mathbf{x})\|^2 \geq \mu(F(\mathbf{x}) - F^*). \tag{5}$$

When $F(\mathbf{x})$ is strongly convex, it also implies the PL condition. However, it is not vice versa. Hence we can arrive at the following results.

Theorem 1. Let Assumptions 1,2 and 3 hold. Assume that the delay compensated gradients are uniformly bounded, i.e., there exits a scalar $B > 0$ such that

$$\|\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \leq B, \ \ \forall t \geq 0 \ and \ 0 \leq r \leq \tau - 1, \tag{6}$$

and that $\nabla F(\mathbf{x}_t)$ is ξ_m-smooth for all $t \geq 0$. Then for the iterates generated by PC-ASGD, when $0 < \eta \leq \frac{1}{2\mu\tau}$ and the objective satisfies the PL condition, they satisfy

$$\mathbb{E}[F(\mathbf{x}_t) - F^*] \leq (1-2\mu\eta\tau)^{t-1}(F(\mathbf{x}_1) - F^* - \frac{Q}{2\mu\eta\tau}) + \frac{Q}{2\mu\eta\tau}, \tag{7}$$

$$Q = 2(1 - 2\mu\eta\tau)G\eta C_1 + \frac{\eta^3 \xi_m G}{2} \sum_{r=1}^{\tau-1} C_r + 2\eta^2 G\gamma_m C_1 + G\eta\tau\sigma$$
$$+ \eta^2 G(\gamma_m + \epsilon_D + \epsilon + (1-\lambda)G^2) \sum_{r=1}^{\tau-1} C_r + \eta G^2 + \eta^2 \gamma_m G\tau C_2 \tag{8}$$

and $C_1 = \frac{G+(\tau-1)B\theta_m}{1-\delta_2}, C_r = \frac{2G+(r-1)B\theta_m}{1-\delta_2}, C_2 = \frac{2G+(\tau-1)B\theta_m}{1-\delta_2}$, $\epsilon_D > 0$ and $\epsilon > 0$ are upper bounds for the approximation errors of the Hessian matrix that are obtained in the supplementary materials.

The proof for this theorem is fairly non-trivial and technical. We refer readers to the supplementary materials for more detail. To simplify the proof, this main result will be divided into several lemmas. One implication from Theorem 1 is that PC-ASGD enables the iterates $\{\mathbf{x}_t\}$ to converge to the neighborhood of $\mathbf{x}^*$, which is $\frac{Q}{2\eta\mu\tau}$. In addition, Theorem 1 shows that the error bound is significantly attributed to network errors caused by the disagreement among agents with respect to the delay and the variance of stochastic gradients. Another implication can be made from Theorem 1 is that the convergence rate is closely related to the delay and the step size such that when the delay is large it may reduce the coefficient, $1 - 2\mu\eta\tau$, to speed up the convergence. However, correspondingly the upper bound of the step size is also reduced. Hence, there is a tradeoff between the step size and the delay in PC-ASGD. Theorem 1 also suggests that when the objective function only satisfies the PL condition and is smooth, the convergence to the neighborhood of $\mathbf{x}^*$ in a linear rate can still be achieved. The PL condition may not necessarily imply convexity and hence the conclusion can even apply to some nonconvex functions.

We next investigate the convergence for the non-convex objectives. For PC-ASGD, we show that it converges to a first-order stationary point in a sublinear rate. It should be noted that such a result may not absolutely guarantee a feasible minimizer due to lack of some necessary second-order information. However, for most nonconvex optimization problem, this is generic, though some existing works have discussed about the second-order stationary points [28], which is out of our investigation scope.

Theorem 2. Let Assumptions 1, 2 and 3 hold. Assume that the delay compensated gradients are uniformly bounded, i.e., there exists a a scalar $B > 0$ such that for all $T \geq 1$

$$\|\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \leq B, \;\; \forall t \geq 0 \;\; and \;\; 0 \leq r \leq \tau - 1, \tag{9}$$

and that

$$\mathbb{E}[\|\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2] \leq M. \tag{10}$$

Then for the iterations generated by PC-ASGD, there exists $0 < \eta < \frac{1}{\gamma_m}$, such that

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla F(\mathbf{x}_t)\|^2] \leq \frac{2(F(\mathbf{x}_1) - F^*)}{T\eta} + \frac{R}{\eta}, \tag{11}$$

where, $R = 2GC_1 + \frac{\tau^2\eta^2\gamma_m M}{2} + \frac{\eta\sigma^2}{2} + \eta\sigma\tau B + 2\eta\gamma_m(\tau B + G)C_1, C_1 = \frac{G+(\tau-1)B\theta_m}{1-\delta_2}$.

Remark 2. Theorem 2 states that with a properly chosen constant step size, PC-ASGD is able to converge the iterates $\{\mathbf{x}_T\}$ to the neighborhood of a stationary point $\mathbf{x}^*$ in a rate of $O(T^{-1})$, whose radius is determined by $\frac{R}{\eta}$. Additionally, based on R, we can know that the error bound is mainly caused by the variance of stochastic gradients and the network errors. One can also observe that the error bound increases when the delay becomes larger. As the length of delay can have an impact on the prediction steps used in the delay compensated gradient, a short term prediction may help alleviate the negative effect caused by the delay. Otherwise, the compounding error in the delay compensated gradient could deteriorate the performance of the algorithm.

5 Experiments

5.1 Practical Variant

So far, we have analyzed theoretically in detail how the proposed PC-ASGD converges with some mild assumptions. In practical implementation, we need to choose a suitable θ_t to enable the training fast with clipping steps and allow the unreliable neighbors to be involved in training with predicting steps. In this context, we develop a heuristic practical variant with a criterion for determining the tradeoff parameter value. Intuitively, if the delay messages from the unreliable neighbors do not influence the training negatively, they should be included in the prediction. This can be determined by the comparison with the algorithm without making use of these messages. The criterion is shown as follows:

$$x_i^{t+1} = \begin{cases} x_{t+1,pre}^i & \frac{\langle x_{t+1,pre}^i - x_t^i, g_i(x_t^i)\rangle}{\|x_{t+1,pre}^i - x_t^i\|} \geq \frac{\langle x_{t+1,cli}^i - x_t^i, g_i(x_t^i)\rangle}{\|x_{t+1,cli}^i - x_t^i\|} \\ x_{t+1,cli}^i & o.w. \end{cases}, \tag{12}$$

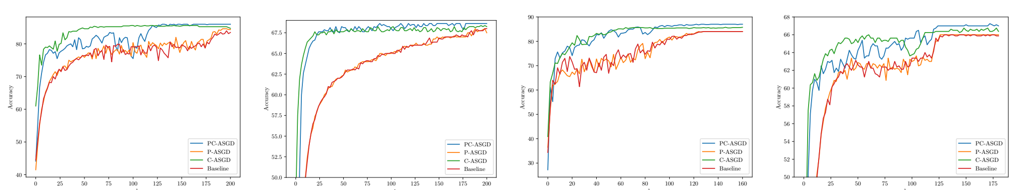
where we choose the *cosine distance* to compare the distances for predicting and clipping steps. The prediction step is selected if it has the larger cosine distance, which implies that the update due to the predicting step yields the larger loss descent. Otherwise, the clipping step should be chosen by only trusting reliable neighbors. Our practical variant with this criterion still converges since we just set θ_t as 0 or 1 for each iteration and the previous analysis in our paper still holds. To facilitate the understanding of predicting and clipping steps, in the following experiments, we also have two other variants P-ASGD and C-ASGD. While the former corresponds to an "optimistic" scenario to only rely on the predicting step, the latter presents a "pessimistic" scenario by dropping all outdated agents. Both of variants follow the same convergence rates induced by PC-ASGD.

5.2 Distributed Network and Learning Setting

Models and Data sets. Decentralized asynchronous SGD (D-ASGD) is adopted as the baseline algorithm. Two deep learning structures, PreResNet110 and DenseNet (noted as *model 1* and *model 2*), are employed. The detailed model structures are illustrated in the supplementary material. CIFAR-10 and CIFAR-100 are used in the experiments following the settings in [29]. The training data is randomly assigned to each agent, and the parameters of the deep learning structure are maintained within each agent and communicated with the predefined delays. The testing set is utilized for each agent to verify the performance, where our metric is the average accuracy among the agents. 6 runs are carried out for each case and the mean and variance are obtained and listed in Table 3.

Delay setting. The delay is set as τ as discussed before, which means the parameters received from the agents outside of the reliable cluster are the ones that were obtained τ iterations before. For *model 1* and *model 2*, τ is both fixed at 20 to test the performances of different algorithms including our different variants (D-ASGD, P-ASGD, C-ASGD, and PC-ASGD) and baseline algorithms in Section 5.3 and 5.5. We also try to exploit its impact in Section 5.4.

Distributed network setting. A distributed network (noted as *distributed network 1*) with 8 agents (nodes) in a fully connected graph is first applied with *model 1* and *model 2*, and 2 clusters of reliable agents are defined within the graph consisting of 3 agents and 5 agents, respectively. Then two distributed networks (with 5-agent and 20-agent, respectively) are used for scalability analysis, noted as *distributed network 2* and *distributed network 3*, respectively. For *distributed network 2*, we construct 2 clusters of reliable agents with 3 and 2 agents. In *distributed network 3*, four clusters are formed and 3 clusters consist of 6 agents while the rest has 2 agents.



(a) DenseNet CIFAR-10 (b) DenseNet CIFAR-100 (c) PreResNet110 CIFAR-10 (d) PreResNet110 CIFAR-100

Figure 1: Testing accuracy on CIFAR-10 and CIFAR-100.

5.3 Performance evaluation

The testing accuracies on the CIFAR-10 and CIFAR-100 data sets with *model 1* and *model 2* in *distributed network 1* are shown in Fig. 1. It shows that the proposed PC-ASGD outperforms the other single variants and it presents an accuracy increment greater than 2.3% (nearly 4% for DenseNet with CIFAR-10) compared to the baseline algorithm. For other variants P-ASGD or C-ASGD, the testing accuracies are also higher than that of the baseline algorithm. Moreover, PC-ASGD shows faster convergence than P-ASGD as the updating rule overcomes the staleness, and achieves better accuracy than the C-ASGD as it includes the messages from the unreliable neighbors. This is consistent with the analysis in this work. We also show the detailed results of both *distributed network 1* and *distributed network 3* in Table 2.

Table 2: Performance evaluation of PC-ASGD on CIFAR-10 and CIFAR-100

8 agents							
Model & dataset	PC-ASGD		P-ASGD		C-ASGD		Baseline
	acc. (%)	o.p. (%)	acc. (%)	o.p. (%)	acc.(%)	o.p. (%)	acc. (%)
Pre110, CIFAR-10	$\mathbf{87.3 \pm 1.1}$	$\mathbf{3.3 \pm 1.1}$	84.9 ± 0.9	0.9 ± 0.9	86.0 ± 1.0	2.0 ± 1.0	84.0 ± 0.3
Pre110, CIFAR-100	$\mathbf{67.4 \pm 1.4}$	$\mathbf{3.1 \pm 1.9}$	64.8 ± 1.3	1.3 ± 1.5	66.4 ± 1.2	1.9 ± 1.6	64.5 ± 1.5
Des, CIFAR-10	$\mathbf{86.9 \pm 0.9}$	$\mathbf{3.6 \pm 1.8}$	84.4 ± 0.6	1.0 ± 1.5	85.9 ± 0.9	2.7 ± 1.7	83.3 ± 0.9
Des, CIFAR-100	$\mathbf{68.6 \pm 0.6}$	$\mathbf{2.3 \pm 1.7}$	66.8 ± 1.5	1.6 ± 1.6	66.8 ± 1.6	1.8 ± 1.6	66.1 ± 1.9
20 agents							
Model & dataset	PC-ASGD		P-ASGD		C-ASGD		Baseline
	acc. (%)	o.p. (%)	acc. (%)	o.p. (%)	acc.(%)	o.p. (%)	acc. (%)
Pre110, CIFAR-10	$\mathbf{84.7 \pm 0.9}$	$\mathbf{4.2 \pm 1.0}$	83.3 ± 0.9	2.7 ± 0.9	82.5 ± 1.0	1.9 ± 1.4	80.4 ± 0.7
Pre110, CIFAR-100	$\mathbf{62.4 \pm 0.8}$	$\mathbf{3.3 \pm 2.0}$	61.7 ± 1.0	2.0 ± 1.6	61.5 ± 1.0	2.5 ± 2.3	59.3 ± 1.7
Des, CIFAR-10	$\mathbf{82.9 \pm 0.9}$	$\mathbf{2.4 \pm 0.9}$	82.0 ± 0.7	1.4 ± 1.3	81.8 ± 0.6	1.8 ± 1.0	80.1 ± 0.9
Des, CIFAR-100	$\mathbf{64.5 \pm 0.7}$	$\mathbf{3.8 \pm 1.7}$	62.5 ± 1.3	2.9 ± 2.0	62.0 ± 1.5	1.3 ± 1.4	60.4 ± 1.7

acc.–accuracy, o.p.–outperformed comparing to baseline.

We then compare our proposed algorithm with other delay-tolerant algorithms, including the baseline algorithm D-ASGD (aka AD-PSGD), DC-s3gd [23], DASGD with IS [22], and Adaptive Braking [25]. The *distributed network 1* is applied for the comparisons.

Table 3: Performance comparison for different delay tolerant algorithms

Model & dataset	Pre110,CIFAR-10	Pre110,CIFAR-100	Des,CIFAR-10	Des,CIFAR-100
PC-ASGD	$\mathbf{87.3 \pm 1.1}$	$\mathbf{67.4 \pm 1.4}$	$\mathbf{86.9 \pm 0.6}$	$\mathbf{68.6 \pm 0.6}$
AD-PSGD [2]	84.0 ± 0.3	64.5 ± 1.5	83.3 ± 0.9	66.1 ± 1.9
DC-s3gd [23]	86.3 ± 0.8	63.5 ± 1.7	85.7 ± 0.8	66.2 ± 1.3
DASGD with IS [22]	85.0 ± 0.3	64.6 ± 1.2	84.6 ± 0.4	66.2 ± 0.8
Adaptive Braking [25]	86.8 ± 0.9	66.5 ± 1.2	85.3 ± 1.0	67.3 ± 1.1

From the Table 3, the proposed PC-ASGD obtains the best results in the four cases. It should be noted that some of above listed algorithms are not designed specifically for this kind of peer-to-peer applications (e.g., Adaptive Braking) or may not consider the modelling of severe delays in their works (e.g., DASGD with IS and DC-s3gd). In this context, they may not perform well in the test cases.

5.4 Impacts of different delay settings

To further show our algorithm's effectiveness, we also implement experiments with different delays. As discussed above, a more severe delay could cause significant drop on the accuracy. More numerical studies with different steps of delay are carried out here. The delays are set as $5, 20, 60$ with our PreResNet110 (*model 1*) of 8 agents (synchronous network without delay is also tested). We use CIFAR-10 in the studies and the topology is *distributed network 1*. The results are shown in Fig. 2.

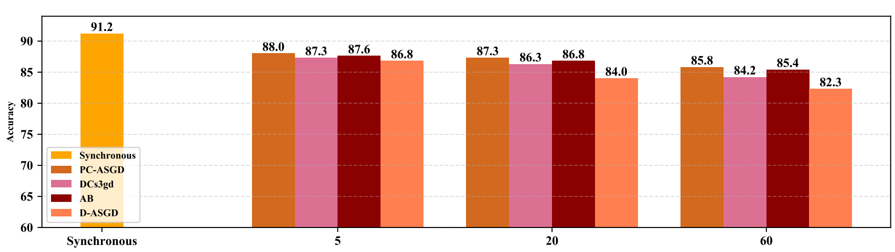


Figure 2: Performance evaluation for different steps of delay

We can find out as the delay increases, the accuracy decreases. For the synchronous setting, the testing accuracy is close to that in the centralized scenario [30] but with higher batch size. When the delay is 60, the accuracy for the D-ASGD reduces significantly, and this validates that the large delay significantly influences the performance and causes difficulties in the training process. However, the

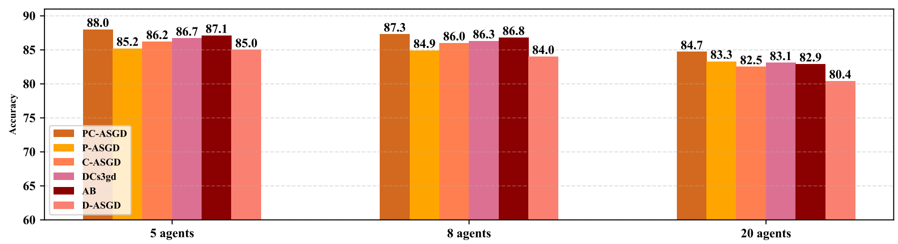


Figure 3: Performance evaluation for different numbers of agents.

delays are practical in the real implementations such as industrial IoT platforms. For our proposed PC-ASGD, it outperforms other algorithms in all cases with different delays. Moreover, the accuracy drop is relatively smaller in cases with larger delays, which suggests that PC-ASGD is more robust to different communication delays.

5.5 Impacts of network size

For evaluating the performances in different structure sizes of distributed networks, *distributed network 2* and *distributed network 3* follow the same setting as in the *distributed network 1* (delay $\tau = 20$, *model 1*, CIFAR-10). The results are shown in Fig. 3. According to both Table 2 and Fig. 3, as the number of agents increases, the accuracy decreases. It shows the large size of the network has negative impact on the training. We also find out that our proposed PC-ASGD outperforms all other approaches, which further validates the efficacy and scalability of the proposed algorithm.

To further show the effectiveness and stability of our proposed algorithm, additional comparisons and results are provided in the supplementary materials including some additional results about influence analysis caused by θ settings and computational costs. These results reflect that our proposed algorithm is able to work well on different distributed systems.

6 Conclusion

This paper presents a novel learning algorithm for distributed deep learning with heterogeneous delay characteristics in agent-communication-network systems. We propose PC-ASGD algorithm consisting of a predicting step, a clipping step, and the corresponding update law for optimizing the positive effects introduced by gradient prediction for reducing the staleness and negative effects caused by the outdated weights. We present theoretical analysis for the convergence rate of the proposed algorithm with constant step size when the objective functions are weakly strongly convex and nonconvex. The numerical studies show the effectiveness of our proposed algorithms in different distributed systems with delays. In future work, the cases for distributed networks with diverse delays and dynamic topology will be further studied and tested.

References

[1] H. Gijzen, "Big data for a sustainable future," *Nature*, vol. 502, no. 7469, pp. 38–38, 2013.

[2] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," *arXiv: Optimization and Control*, 2017.

[3] A. Hard, C. Kiddon, D. Ramage, F. Beaufays, H. Eichner, K. Rao, R. Mathews, and S. Augenstein, "Federated learning for mobile keyboard prediction." *arXiv: Computation and Language*, 2018.

[4] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passeratpalmbach, "A generic framework for privacy preserving deep learning." *arXiv: Learning*, 2018.

[5] Y. Deng, M. M. Kamani, and M. Mahdavi, "Distributionally robust federated averaging," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[6] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[7] S. Cen, H. Zhang, Y. Chi, W. Chen, and T.-Y. Liu, "Convergence of distributed stochastic variance reduced methods without sampling extra data," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3976–3989, 2020.

[8] M. Blot, D. Picard, M. Cord, and N. Thome, "Gossip training for deep learning." *arXiv: Computer Vision and Pattern Recognition*, 2016.

[9] M. Even, H. Hendrikx, and L. Massoulié, "Asynchrony and acceleration in gossip algorithms," *arXiv preprint arXiv:2011.02379*, 2020.

[10] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 5904–5914.

[11] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," *arXiv preprint arXiv:1604.00981*, 2016.

[12] K. Tsianos, S. Lawlor, and M. G. Rabbat, "Communication/computation tradeoffs in consensus-based distributed optimization," in *Advances in neural information processing systems*, 2012, pp. 1943–1951.

[13] J. Dean, G. S. Corrado, R. Monga, K. Chen, and A. Y. Ng, "Large scale distributed deep networks," *Advances in neural information processing systems*, 2013.

[14] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," *arXiv: Optimization and Control*, 2011.

[15] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson, "An asynchronous mini-batch algorithm for regularized stochastic optimization," *conference on decision and control*, vol. 61, no. 12, pp. 1384–1389, 2015.

[16] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," *Advances in neural information processing systems*, vol. 24, pp. 693–701, 2011.

[17] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, and T.-Y. Liu, "Asynchronous stochastic gradient descent with delay compensation," in *International Conference on Machine Learning*. PMLR, 2017, pp. 4120–4129.

[18] X. Liang, A. M. Javid, M. Skoglund, and S. Chatterjee, "Asynchrounous decentralized learning of a neural network," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3947–3951.

[19] R. Nair and S. Gupta, "Wildfire: approximate synchronization of parameters in distributed deep learning," *Ibm Journal of Research and Development*, vol. 61, no. 4, p. 7, 2017.

[20] K. I. Tsianos and M. G. Rabbat, "Efficient distributed online prediction and stochastic optimization with approximate distributed averaging," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 489–506, 2016.

[21] G. Lan, S. Lee, and Y. Zhou, "Communication-efficient algorithms for decentralized and stochastic optimization," *Mathematical Programming*, vol. 180, no. 1, pp. 237–284, 2020.

[22] Y. Du, K. You, and Y. Mo, "Asynchronous stochastic gradient descent over decentralized datasets," in *2020 IEEE 16th International Conference on Control & Automation (ICCA)*. IEEE, 2020, pp. 216–221.

[23] A. Rigazzi, "Dc-s3gd: Delay-compensated stale-synchronous sgd for large-scale decentralized neural network training." *arXiv: Learning*, 2019.

[24] M. Zakharov, "Asynchronous Consensus Algorithm," *arXiv e-prints*, p. arXiv:2001.07704, Jan 2020.

[25] A. Venigalla, A. Kosson, V. Chiley, and U. Köster, "Adaptive braking for mitigating gradient delay," *arXiv preprint arXiv:2007.01397*, 2020.

[26] S. Abbasloo and H. J. Chao, "SharpEdge: An Asynchronous and Core-Agnostic Solution to Guarantee Bounded-Delays," *arXiv e-prints*, p. arXiv:2001.00112, Dec 2019.

[27] H. Karimi, J. Nutini, and M. Schmidt, "Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 795–811.

[28] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, "Accelerated methods for nonconvex optimization," *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1751–1772, 2018.

[29] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[30] W. Yang, "pytorch-classification," https://github.com/bearpaw/pytorch-classification, 2019, accessed: 2019-01-24.

[31] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[32] S. Becker and Y. Lecun, "Improving the convergence of back-propagation learning with second-order methods," in *Proceedings of the 1988 Connectionist Models Summer School, San Mateo*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. Morgan Kaufmann, 1989, pp. 29–37.

[33] S. K. Mishra, "Some new test functions for global optimization and performance of repulsive particle swarm method," *Available at SSRN 926132*, 2006.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3-5.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code and the data are proprietary.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E in Supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section Table 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We simply use the 4 cpus Xeon(R) with 12 cores and GTX 1080 Ti 12 GB for 8 in GPU training, which could be commonly used in deep learning training.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.
   (b) Did you mention the license of the assets? [Yes] See Section 5.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We don't have new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We just use open dataset.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] The dataset used is open dataset.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [Yes]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes]