Distributed Stochastic Gradient Descent with Event-Triggered Communication

Jemin George,[1] **Prudhvi Gurram**[2,1]
[1]CCDC Army Research Laboratory
Adelphi, MD 20783
jemin.george.civ@mail.mil
[2]Booz Allen Hamilton
McLean, VA 22102
gurram_prudhvi@bah.com

Abstract

We develop a Distributed Event-Triggered Stochastic GRAdient Descent (DETSGRAD) algorithm for solving non-convex optimization problems typically encountered in distributed deep learning. We propose a novel communication triggering mechanism that would allow the networked agents to update their model parameters aperiodically and provide sufficient conditions on the algorithm step-sizes that guarantee the asymptotic mean-square convergence. The algorithm is applied to a distributed supervised-learning problem, in which a set of networked agents collaboratively train their individual neural networks to perform image classification, while aperiodically sharing the model parameters with their one-hop neighbors. Results indicate that all agents report similar performance that is also comparable to the performance of a centrally trained neural network, while the event-triggered communication provides significant reduction in inter-agent communication. Results also show that the proposed algorithm allows the individual agents to classify the images even though the training data corresponding to all the classes are not locally available to each agent.

Introduction

With the advent of smart devices, there has been an exponential growth in the amount of data collected and stored locally on individual devices. Applying machine learning to extract value from such massive data to provide data-driven insights, decisions, and predictions has been a popular research topic as well as the focus of numerous businesses. However, porting these vast amounts of data to a data center to conduct traditional machine learning has raised two main issues: (i) the communication challenge associated with transferring vast amounts of data from a large number of devices to a central location and (ii) the privacy issues associated with sharing raw data. Distributed machine learning techniques based on the server-client architecture (Li et al. 2014a; 2014b; Zhang, Alqahtani, and Demirbas 2017) have been proposed as solutions to this problem. On one extreme end of this architecture, we have the *parameter server* approach, where a server or group of servers initiate distributed learning by pushing the current model to a set of client nodes that host the data. The client nodes compute the local gradients or parameter updates and communicate them to the server nodes. Server nodes aggregate these values and update the current model (Zhang et al. 2018; Li et al. 2014b). On the other extreme, we have *federated learning*, where each client node obtains a local solution to the learning problem and the server node computes a global model by averaging the local models (Konečnú et al. 2016; McMahan et al. 2017). Besides the server-client architecture, a *shared-memory* (multicore/multiGPU) architecture, where different processors independently compute the gradients and update the global model parameter using a shared memory, has also been proposed as a solution to the distributed machine learning problem (Recht et al. 2011; De Sa et al. 2015; Chaturapruek, Duchi, and Ré 2015; Feyzmahdavian, Aytekin, and Johansson 2016). However, none of the above-mentioned learning techniques are truly distributed since they follow a master-slave architecture and do not involve any peer-to-peer communication. Furthermore, these techniques are not always robust and they are rendered useless if the master/server node or the shared-memory fails. Therefore, we aim to develop a fully distributed machine learning architecture enabled by client-to-client interaction.

For large-scale machine learning, stochastic gradient descent (SGD) methods are often preferred over batch gradient methods (Bottou, Curtis, and Nocedal 2018) because (i) in many large-scale problems, there is a good deal of redundancy in data and therefore it is inefficient to use all the data in every optimization iteration, (ii) the computational cost involved in computing the batch gradient is much higher than that of the stochastic gradient, and (iii) stochastic methods are more suitable for online learning where data are arriving sequentially. Since most machine learning problems are non-convex, there is a need for distributed stochastic gradient methods for non-convex problems. Therefore, here we present a communication efficient, distributed stochastic gradient algorithm for non-convex problems and demonstrate its utility for distributed machine learning.

Related work

Distributed Non-Convex Optimization: A few early examples of (non-stochastic or deterministic) distributed non-

convex optimization algorithms include the Distributed Approximate Dual Subgradient (DADS) Algorithm (Zhu and Martínez 2013), NonconvEx primal-dual SpliTTing (NESTT) algorithm (Hajinezhad et al. 2016), and the Proximal Primal-Dual Algorithm (Prox-PDA) (Hong, Hajinezhad, and Zhao 2017). More recently, a non-convex version of the accelerated distributed augmented Lagrangians (ADAL) algorithm is presented in Chatzipanagiotis and Zavlanos(2017) and successive convex approximation (SCA)-based algorithms such as iNner cOnVex Approximation (NOVA) and in-Network succEssive conveX approximaTion algorithm (NEXT) are given in Scutari, Facchinei, and Lampariello(2017) and Lorenzo and Scutari(2016), respectively. References (Hong 2018; Guo, Hug, and Tonguz 2017; Hong, Luo, and Razaviyayn 2016) provide several distributed alternating direction method of multipliers (ADMM) based non-convex optimization algorithms. Non-convex versions of Decentralized Gradient Descent (DGD) and Proximal Decentralized Gradient Descent (Prox-DGD) are given in Zeng and Yin(2018). Finally, Zeroth-Order NonconvEx (ZONE) optimization algorithms for mesh network (ZONE-M) and star network (ZONE-S) are presented in Hajinezhad, Hong, and Garcia(2019). However, almost all aforementioned *consensus optimization* algorithms focus on non-stochastic problems and are extremely communication heavy because they require constant communication among the agents.

Distributed Convex SGD: Within the consensus optimization literature, there exist several works on distributed stochastic gradient methods, but mainly for strongly convex optimization problems. These include the stochastic subgradient-push method for distributed optimization over time-varying directed graphs given in Nedić and Olshevsky(2016), distributed stochastic optimization over random networks given in Jakovetic et al.(2018), the Stochastic Unbiased Curvature-aided Gradient (SUCAG) method given in Wai et al.(2018), and distributed stochastic gradient tracking methods Pu and Nedić(2018). There are very few works on distributed stochastic gradient methods for non-convex optimization (Tatarenko and Touri 2017; Bianchi and Jakubowicz 2013); however, the push-sum algorithm given in Tatarenko and Touri(2017) assumes there are no saddle-points and it often requires up to 3 times as many internal variables as the proposed algorithm. Compared to Bianchi and Jakubowicz(2013) and Tatarenko and Touri(2017), the proposed algorithm provides an explicit consensus rate and allows the parallel execution of the consensus communication and gradient computation steps.

Parallel SGD: There exist numerous asynchronous SGD algorithms aimed at parallelizing the data-intensive machine learning tasks. The two popular asynchronous parallel implementations of SGD are the computer network implementation originally proposed in Agarwal and Duchi(2011) and the shared memory implementation introduced in Recht et al.(2011). Computer network implementation follows the master-slave architecture and Agarwal and Duchi(2011) showed that for smooth convex problems, the delays due to asynchrony are asymptotically negligible. Feyzmahdavian, Aytekin, and Johansson(2016) extend the results in Agarwal and Duchi(2011) for regularized SGD. Extensions of the computer network implementation of asynchronous SGD with variance reduction and polynomially growing delays are given in Huo and Huang(2016) and Zhou et al.(2018), respectively. Recht et al.(2011) proposed a lock-free asynchronous parallel implementation of SGD on a shared memory system and proved a sublinear convergence rate for strongly convex smooth objectives. The lock-free algorithm, HOGWILD!, proposed in Recht et al.(2011) has been applied to PageRank approximation (Mitliagkas et al. 2015), deep learning (Noel and Osindero 2014), and recommender systems (Yu et al. 2012). In Duchi, Jordan, and McMahan(2013), authors extended the HOGWILD! algorithm to a dual averaging algorithm that works for non-smooth, non-strongly convex problems with sparse gradients. An extension of HOGWILD! called BUCK-WILD! is introduced in De Sa et al.(2015) to account for quantization errors introduced by fixed-point arithmetic. In Chaturapruek, Duchi, and Ré(2015), the authors show that because of the noise inherent to the sampling process within SGD, the errors introduced by asynchrony in the shared-memory implementation are asymptotically negligible. Recently, several parallel SGD works focus on adjusting the worker-server interaction period or frequency as a way to decrease the communication overhead. For example, (Yu, Jin, and Yang 2019) and (Yu, Yang, and Zhu 2019) used a fixed period, while (Yu and Jin 2019) and (Lin, Stich, and Jaggi 2018) propose an increasing period as a way to reduce communication. A detailed comparison of both computer network and shared memory implementation is given in Lian et al.(2015). Again, the aforementioned asynchronous algorithms are not distributed since they rely on a shared-memory or central coordinator.

Decentralized SGD: Recently, numerous *decentralized SGD* algorithms for non-convex optimization have been proposed as a solution to the communication bottleneck often encountered in the server-client architecture (Lian et al. 2017; Jiang et al. 2017; Tang et al. 2018; Lian et al. 2018; Wang and Joshi 2018; Haddadpour et al. 2019; Assran et al. 2019; Wang et al. 2019). However almost all these works primarily focus on the performance of the algorithm during a fixed time interval, and the constant algorithm step-size, which often depends on the final time, is selected to speed-up the convergence rate. These SGD algorithms with constant step-size can only guarantee convergence to some ϵ-ball of the stationary point. Furthermore, most of the aforementioned decentralized SGD algorithms provide convergence rates in terms of the average of all local estimates of the global minimizer without ever proving a similar or faster consensus rate. In fact, most decentralized SGD algorithms can only provide bounded consensus and they require a centralized averaging step after running the algorithm until the final-time (Lian et al. 2017; Tang et al. 2018; Lian et al. 2018; Haddadpour et al. 2019; Wang et al. 2019). Finally, most application of decentralized SGD focus on distributed learning scenarios where the data is distributed identically across all agents.

Contribution: Currently, there exists no distributed SGD algorithm for the non-convex problems that doesn't require constant or periodic communication among the agents. In fact, algorithms in (Lian et al. 2017; Jiang et al. 2017; Tang et al. 2018; Lian et al. 2018; Wang and Joshi 2018;

Haddadpour et al. 2019; Assran et al. 2019; Wang et al. 2019) all rely on periodic communication despite the local model has not changed from previously communicated model. This is a waste of resources, especially in wireless setting and therefore we propose an approach that would allow the nodes to transmit only if the local model has significantly changed from previously communicated model. The contributions of this paper are three-fold: (i) we propose a fully distributed machine learning architecture, (ii) we present a distributed SGD algorithm built on a novel communication triggering mechanism, and provide sufficient conditions on step-sizes such that the algorithm is mean-square convergent, and (iii) we demonstrate the efficacy of the proposed event-triggered SGD algorithm for distributed supervised learning with i.i.d. and more importantly, non-i.i.d. data.

Notation: Let $\mathbb{R}^{n\times m}$ denote the set of $n\times m$ real matrices. For a vector $\boldsymbol{\phi}$, ϕ_i is the $i-$ th entry of $\boldsymbol{\phi}$. An $n\times n$ identity matrix is denoted as I_n and $\mathbf{1}_n$ denotes an n-dimensional vector of all ones. For $p\in[1,\,\infty]$, the p-norm of a vector $\mathbf{x}$ is denoted as $\|\mathbf{x}\|_p$. For matrices $A\in\mathbb{R}^{m\times n}$ and $B\in\mathbb{R}^{p\times q}$, $A\otimes B\in\mathbb{R}^{mp\times nq}$ denotes their Kronecker product. For a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ of order n, $\mathcal{V}\triangleq\{1,\ldots,n\}$ represents the agents or nodes and the communication links between the agents are represented as $\mathcal{E}\triangleq\{e_1,\ldots,e_\ell\}\subseteq\mathcal{V}\times\mathcal{V}$. Let $\mathcal{N}_i\triangleq\{j\in\mathcal{V}\,:\,(i,j)\in\mathcal{E}\}$ denote the set of neighbors of node i. Let $\mathcal{A}=[a_{ij}]\in\mathbb{R}^{n\times n}$ be the *adjacency matrix* with entries of $a_{ij}=1$ if $(i,j)\in\mathcal{E}$ and zero otherwise. Define $\Delta=\mathrm{diag}\,(\mathcal{A}\mathbf{1}_n)$ as the in-degree matrix and $\mathcal{L}=\Delta-\mathcal{A}$ as the graph *Laplacian*.

Distributed machine learning

Our problem formulation closely follows the centralized machine learning problem discussed in Bottou, Curtis, and Nocedal(2018). Consider a networked set of n agents, each with a set of m_i, $i=1,\ldots,n$, independently drawn input-output samples $\{\boldsymbol{x}_i^j,\,\boldsymbol{y}_i^j\}_{j=1}^{j=m_i}$, where $\boldsymbol{x}_i^j\in\mathbb{R}^{d_x}$ and $\boldsymbol{y}_i^j\in\mathbb{R}^{d_y}$ are the j-th input and output data, respectively, associated with the i-th agent. For example, the input data could be images and the outputs could be labels. Let $h\left(\cdot\,;\,\cdot\right):\mathbb{R}^{d_x}\times\mathbb{R}^{d_w}\mapsto\mathbb{R}^{d_y}$ denote the prediction function, fully parameterized by the vector $\boldsymbol{w}\in\mathbb{R}^{d_w}$. Each agent aims to find the parameter vector that minimizes the losses, $\ell\left(\cdot\,;\,\cdot\right):\mathbb{R}^{d_y}\times\mathbb{R}^{d_y}\mapsto\mathbb{R}$, incurred from inaccurate predictions. Thus, the loss function $\ell\left(h\left(\boldsymbol{x}_i;\boldsymbol{w}\right),\boldsymbol{y}_i\right)$ yields the loss incurred by the i-th agent, where $h\left(\boldsymbol{x}_i;\boldsymbol{w}\right)$ and $\boldsymbol{y}_i$ are the predicted and true outputs, respectively, for the i-th node.

Assuming the input output space $\mathbb{R}^{d_x}\times\mathbb{R}^{d_y}$ associated with the i-th agent is endowed with a probability measure $P_i\,:\,\mathbb{R}^{d_x}\times\mathbb{R}^{d_y}\mapsto[0,\,1]$, the objective function an agent wishes to minimize is

$$\begin{aligned}R_i(\boldsymbol{w})&=\int_{\mathbb{R}^{d_x}\times\mathbb{R}^{d_y}}\ell\left(h\left(\boldsymbol{x}_i;\boldsymbol{w}\right),\boldsymbol{y}_i\right)\,dP_i\left(\boldsymbol{x}_i,\boldsymbol{y}_i\right)\\&=\mathbb{E}_{P_i}\left[\ell\left(h\left(\boldsymbol{x}_i;\boldsymbol{w}\right),\boldsymbol{y}_i\right)\right].\end{aligned}\tag{1}$$

Here $R_i(\boldsymbol{w})$ denotes the expected risk given a parameter vector $\boldsymbol{w}$ with respect to the probability distribution P_i. The total expected risk across all networked agents is given as

$$R(\boldsymbol{w})=\sum_{i=1}^{n}R_i(\boldsymbol{w})=\sum_{i=1}^{n}\mathbb{E}_{P_i}\left[\ell\left(h\left(\boldsymbol{x}_i;\boldsymbol{w}\right),\boldsymbol{y}_i\right)\right].\tag{2}$$

Minimizing the expected risk is desirable but often unattainable since the distributions P_i are unknown. Thus, in practice, each agent chooses to minimize the empirical risk $\bar{R}_i(\boldsymbol{w})$ defined as

$$\bar{R}_i(\boldsymbol{w})=\frac{1}{m_i}\sum_{j=1}^{m_i}\ell\left(h\left(\boldsymbol{x}_i^j;\boldsymbol{w}\right),\boldsymbol{y}_i^j\right).\tag{3}$$

Here, the assumption is that m_i is large enough so that $\bar{R}_i(\boldsymbol{w})\approx R_i(\boldsymbol{w})$. The total empirical risk across all networked agents is

$$\bar{R}(\boldsymbol{w})=\sum_{i=1}^{n}\bar{R}_i(\boldsymbol{w})=\sum_{i=1}^{n}\frac{1}{m_i}\sum_{j=1}^{m_i}\ell\left(h(\boldsymbol{x}_i^j;\boldsymbol{w}),\boldsymbol{y}_i^j\right)\tag{4}$$

To simplify the notation, let us represent a sample input-output pair $(\boldsymbol{x}_i,\,\boldsymbol{y}_i)$ by a random seed $\boldsymbol{\xi}_i$ and let $\boldsymbol{\xi}_i^j$ denote the j-th sample associated with the i-th agent. Define the loss incurred for a given $\left(\boldsymbol{w},\boldsymbol{\xi}_i^j\right)$ as $\ell\left(\boldsymbol{w},\boldsymbol{\xi}_i^j\right)$. Now, the distributed learning problem can be posed as an optimization involving sum of local empirical risks, i.e.,

$$\min_{\boldsymbol{w}}\,f(\boldsymbol{w})=\min_{\boldsymbol{w}}\sum_{i=1}^{n}f_i\left(\boldsymbol{w}\right),\tag{5}$$

where $f_i\left(\boldsymbol{w}\right)=\frac{1}{m_i}\sum_{j=1}^{m_i}\ell\left(\boldsymbol{w},\boldsymbol{\xi}_i^j\right)$.

Distributed event-triggered SGD

Here we propose a distributed event-triggered stochastic gradient method to solve (5). Let $\boldsymbol{w}_i(k)\in\mathbb{R}^{d_w}$ denote agent i's estimate of the optimizer at time instant k. Thus, for an arbitrary initial condition $\boldsymbol{w}_i(0)$, the update rule at node i is as follows:

$$\begin{aligned}\boldsymbol{w}_i(k+1)=\boldsymbol{w}_i(k)&-\beta_k\sum_{j=1}^{n}a_{ij}\left(\hat{\boldsymbol{w}}_i(k)-\hat{\boldsymbol{w}}_j(k)\right)\\&-\alpha_k\,\mathbf{g}_i\left(\boldsymbol{w}_i(k),\boldsymbol{\xi}_i(k)\right),\end{aligned}\tag{6}$$

where α_k and β_k are hyper parameters to be specified, a_{ij} are the entries of the adjacency matrix and $\mathbf{g}_i\left(\boldsymbol{w}_i(k),\boldsymbol{\xi}_i(k)\right)$ represents either a simple stochastic gradient, mini-batch stochastic gradient or a stochastic quasi-Newton direction, i.e.,

$$\mathbf{g}_i\left(\boldsymbol{w}_i(k),\boldsymbol{\xi}_i(k)\right)=\begin{cases}\nabla\ell\left(\boldsymbol{w}_i(k),\boldsymbol{\xi}_i^k\right),\ \text{or}\\\frac{1}{n_i(k)}\sum\limits_{s=1}^{n_i(k)}\nabla\ell\left(\boldsymbol{w}_i(k),\boldsymbol{\xi}_i^{k,s}\right),\ \text{or}\\H_i(k)\frac{1}{n_i(k)}\sum\limits_{s=1}^{n_i(k)}\nabla\ell\left(\boldsymbol{w}_i(k),\boldsymbol{\xi}_i^{k,s}\right),\end{cases}$$

where $n_i(k)$ denotes the mini-batch size, $H_i(k)$ is a positive definite scaling matrix, $\boldsymbol{\xi}_i^k$ represents the single random input-output pair sampled at time instant k, and $(\boldsymbol{\xi}_i^{k,s})$ denotes the

s-th input-output pair out of the $n_i(k)$ random input-output pairs sampled at time instant k. For $i = 1, \ldots, n$, the piecewise constant signal $\hat{\boldsymbol{w}}_i(k)$ defined as

$$\hat{\boldsymbol{w}}_i(k) = \boldsymbol{w}_i(t_q^i),\ \forall\, k \in \left\{t_q^i,\ t_q^i + 1,\ \ldots,\ t_{q+1}^i - 1\right\}, \quad (7)$$

denote agent i's last broadcasted estimate of the optimizer. Here $\left\{t_q^i,\ \ q = 0, 1, \ldots\right\}$ with $t_0^i = 0$ denotes triggering instants, i.e., the time instants when agent i broadcasts $\boldsymbol{w}_i$ to its neighbors. Define $\mathbf{w}(k) \triangleq \left[\boldsymbol{w}_1^\top(k) \quad \ldots \quad \boldsymbol{w}_n^\top(k)\right]^\top \in \mathbb{R}^{nd_w}$ and $\hat{\mathbf{w}}(k) \triangleq \left[\hat{\boldsymbol{w}}_1^\top(k) \quad \ldots \quad \hat{\boldsymbol{w}}_n^\top(k)\right]^\top \in \mathbb{R}^{nd_w}$. Now (6) can be written as

$$\begin{aligned}\mathbf{w}(k+1) = \mathbf{w}(k) &- \beta_k \left(\mathcal{L} \otimes I_{d_w}\right) \hat{\mathbf{w}}(k) \\ &- \alpha_k\, \mathbf{g}(\mathbf{w}(k), \boldsymbol{\xi}(k)),\end{aligned} \quad (8)$$

where $\mathcal{L}$ is the network Laplacian and

$$\mathbf{g}(\mathbf{w}(k), \boldsymbol{\xi}(k)) \triangleq \begin{bmatrix} \mathbf{g}_1\left(\boldsymbol{w}_1(k), \boldsymbol{\xi}_1(k)\right) \\ \vdots \\ \mathbf{g}_n\left(\boldsymbol{w}_n(k), \boldsymbol{\xi}_n(k)\right) \end{bmatrix} \in \mathbb{R}^{nd_w}.$$

Let $\boldsymbol{e}_i(k) = \boldsymbol{w}_i(k) - \hat{\boldsymbol{w}}_i(k)$ and $\mathbf{e}(k) = \mathbf{w}(k) - \hat{\mathbf{w}}(k)$. Now (8) can be written as

$$\begin{aligned}\mathbf{w}(k+1) = \left(\mathcal{W}_k \otimes I_{d_w}\right) \mathbf{w}(k) &+ \beta_k \left(\mathcal{L} \otimes I_{d_w}\right) \mathbf{e}(k) \\ &- \alpha_k\, \mathbf{g}(\mathbf{w}(k), \boldsymbol{\xi}(k)),\end{aligned} \quad (9)$$

where $\mathcal{W}_k = (I_n - \beta_k \mathcal{L})$. The event instants are defined as

$$t_{q+1}^i = \inf\left\{k > t_q^i \mid \ \|\boldsymbol{e}_i(k)\|_1 \geq \upsilon_0\, \alpha_k\right\}, \quad (10)$$

where υ_0 is a positive constant to be defined. Pseudo-code of the proposed distributed event-triggered SGD is given in Algorithm 1 (see supplementary material).

Now we state the following assumption on the individual objective functions:

Assumption 1. Objective functions $f_i(\,\cdot\,)$ and its gradients $\nabla f_i(\,\cdot\,) : \mathbb{R}^{d_w} \mapsto \mathbb{R}^{d_w}$ are Lipschitz continuous with Lipschitz constants $L_i^0 > 0$ and $L_i > 0$, respectively, i.e., $\forall\, \boldsymbol{w}_a,\ \boldsymbol{w}_b \in \mathbb{R}^{d_w},\ i = 1, \ldots, n$, we have

$$\begin{aligned}\|f_i(\,\boldsymbol{w}_a\,) - f_i(\,\boldsymbol{w}_b\,)\|_2 &\leq L_i^0 \|\boldsymbol{w}_a - \boldsymbol{w}_b\|_2 \text{ and} \\ \|\nabla f_i(\,\boldsymbol{w}_a\,) - \nabla f_i(\,\boldsymbol{w}_b\,)\|_2 &\leq L_i \|\boldsymbol{w}_a - \boldsymbol{w}_b\|_2.\end{aligned}$$

Now we introduce $F(\cdot) : \mathbb{R}^{nd_w} \mapsto \mathbb{R}$, an aggregate objective function of local variables

$$F(\mathbf{w}(k)) = \sum_{i=1}^{n} f_i\left(\boldsymbol{w}_i(k)\right). \quad (11)$$

Following Assumption 1, the function $F(\cdot)$ is Lipschitz continuous with Lipschitz continuous gradient $\nabla F(\cdot)$, i.e., $\forall\, \mathbf{w}_a,\ \mathbf{w}_b \in \mathbb{R}^{nd_w}$, we have $\|\nabla F(\,\mathbf{w}_a\,) - \nabla F(\,\mathbf{w}_b\,)\|_2 \leq L\|\mathbf{w}_a - \mathbf{w}_b\|_2$, with constant $L = \max_i\{L_i\}$ and $\nabla F(\,\mathbf{w}\,) \triangleq \left[\nabla f_1(\,\boldsymbol{w}_1\,)^\top \quad \ldots \quad \nabla f_n(\,\boldsymbol{w}_n\,)^\top\right]^\top \in \mathbb{R}^{nd_w}$.

Assumption 2. The function $F(\cdot)$ is lower bounded by $F_{\inf}$, i.e., $F_{\inf} \leq F(\mathbf{w}),\ \forall\, \mathbf{w} \in \mathbb{R}^{nd_w}$.

Without loss of generality, we assume that $F_{\inf} \geq 0$. Now we make the following assumption regarding $\{\alpha_k\}$ and $\{\beta_k\}$:

Assumption 3. Sequences $\{\alpha_k\}$ and $\{\beta_k\}$ are selected as

$$\alpha_k = \frac{a}{(k+1)^{\delta_2}} \quad \text{and} \quad \beta_k = \frac{b}{(k+1)^{\delta_1}}, \quad (12)$$

where $a > 0$, $b > 0$, $0 < 3\delta_1 < \delta_2 \leq 1$, $\delta_1/2 + \delta_2 > 1$, and $\delta_2 > 1/2$.

For sequences $\{\alpha_k\}$ and $\{\beta_k\}$ that satisfy Assumption 3, we have $\sum_{k=1}^{\infty} \alpha_k = \infty$, $\sum_{k=1}^{\infty} \beta_k = \infty$, $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ and $\sum_{k=1}^{\infty} \alpha_k \beta_k^{1/2} < \infty$. Thus α_k and β_k are not summable sequences. However, α_k is square-summable and $\alpha_k\sqrt{\beta_k}$ is summable.

Assumption 4. The interaction topology of n networked agents is given as a connected undirected graph $\mathcal{G}\left(\mathcal{V}, \mathcal{E}\right)$.

Assumption 5. Parameter b in sequence $\{\beta_k\}$ is selected such that

$$\mathcal{W}_0 = (I_n - b\mathcal{L}) \quad (13)$$

has a single eigenvalue at 1 corresponding to the right eigenvector $\mathbf{1}_n$ and the remaining $n-1$ eigenvalues of $\mathcal{W}_0$ are strictly inside the unit circle.

In other words, b is selected such that $b < 1/\sigma_{\max}(\mathcal{L})$, where $\sigma_{\max}(\cdot)$ denotes the largest singular value. Thus, $b\sigma_{\max}(\mathcal{L}) < 1$. Let $\mathbb{E}_\xi[\cdot]$ denote the expected value taken with respect to the distribution of the random variable $\boldsymbol{\xi}_k$ given the filtration $\mathcal{F}_k$ generated by the sequence $\{\mathbf{w}_0, \ldots, \mathbf{w}_k\}$, i.e.,

$$\begin{aligned}\mathbb{E}_\xi[\,\mathbf{w}_{k+1}\,] &= \mathbb{E}[\,\mathbf{w}_{k+1}\,|\mathcal{F}_k] \\ &= \left(\mathcal{W}_k \otimes I_{d_w}\right) \mathbf{w}_k - \alpha_k \mathbb{E}[\,\mathbf{g}(\mathbf{w}_k, \boldsymbol{\xi}_k)\,|\mathcal{F}_k] \text{ a.s.},\end{aligned}$$

where a.s. (almost surely) denote events that occur with probability one. Now we make the following assumptions regarding the stochastic gradient term $\mathbf{g}(\mathbf{w}(k), \boldsymbol{\xi}(k))$.

Assumption 6. Stochastic gradients are unbiased such that

$$\mathbb{E}_\xi\left[\,\mathbf{g}(\mathbf{w}_k, \boldsymbol{\xi}_k)\,\right] = \nabla F(\mathbf{w}_k),\ \text{a.s.} \quad (14)$$

That is to say $\mathbb{E}_\xi\left[\,\mathbf{g}(\mathbf{w}_k, \boldsymbol{\xi}_k)\,\right] = \begin{bmatrix} \mathbb{E}_{\xi_1}\left[\,\mathbf{g}_1\left(\boldsymbol{w}_1(k), \boldsymbol{\xi}_1(k)\right)\,\right] \\ \vdots \\ \mathbb{E}_{\xi_n}\left[\,\mathbf{g}_n\left(\boldsymbol{w}_n(k), \boldsymbol{\xi}_n(k)\right)\,\right] \end{bmatrix} = \begin{bmatrix} \nabla f_1(\,\boldsymbol{w}_1(k)\,) \\ \vdots \\ \nabla f_n(\,\boldsymbol{w}_n(k)\,) \end{bmatrix}$

Assumption 7. Stochastic gradients have conditionally bounded second moment, i.e., there exist scalars $\bar{\mu}_{\upsilon_1} \geq 0$ and $\bar{\mu}_{\upsilon_2} \geq 0$ such that

$$\mathbb{E}_\xi\left[\|\mathbf{g}(\mathbf{w}_k, \boldsymbol{\xi}_k)\|_2^2\right] \leq \bar{\mu}_{\upsilon_1} + \bar{\mu}_{\upsilon_2} \|\nabla F(\mathbf{w}_k)\|_2^2,\ \text{a.s.} \quad (15)$$

Assumption 7 is the bounded variance assumption typically made in all SGD literature.

Pseudo-code of the proposed distributed event-triggered SGD is given in Algorithm 1.

Convergence analysis

Define the average-consensus error as $\tilde{\mathbf{w}}_k = (M \otimes I_{d_w})\, \mathbf{w}_k$, where $M = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$. Note that $M\mathcal{L} = \mathcal{L}$ and $(\mathcal{L} \otimes I_{d_w})\, \tilde{\mathbf{w}}_k = (\mathcal{L} \otimes I_{d_w})\, \mathbf{w}_k$. Thus from (9) we have

$$\begin{aligned}\tilde{\mathbf{w}}_{k+1} = \left(\mathcal{W}_k \otimes I_{d_w}\right) \tilde{\mathbf{w}}_k &+ \beta_k \left(\mathcal{L} \otimes I_{d_w}\right) \mathbf{e}(k) \\ &- \alpha_k \left(M \otimes I_{d_w}\right) \mathbf{g}(\mathbf{w}_k, \boldsymbol{\xi}_k).\end{aligned} \quad (16)$$

Algorithm 1 DETSGRAD algorithm

Input : a, b, υ_0, δ_1 and δ_2
Initialization : $\mathbf{w}(0) = \left[\boldsymbol{w}_1^\top(0) \ \ldots \ \boldsymbol{w}_n^\top(0)\right]^\top$
for Agent $i = 1$ to n **do**
  Sample $\boldsymbol{\xi}_i(0)$ & *compute* $g_i\left(\boldsymbol{w}_i(0), \boldsymbol{\xi}_i(0)\right)$
  Send $\boldsymbol{w}_i(0)$ & *let* $\hat{\boldsymbol{w}}_i^{(i)} = \boldsymbol{w}_i(0)$
  Receive $\boldsymbol{w}_j(0)$ & *let* $\hat{\boldsymbol{w}}_j^{(i)} = \boldsymbol{w}_j(0), \ \forall\, j \in \mathcal{N}_i$
  Update $\boldsymbol{w}_i(1) = \boldsymbol{w}_i(0) - \alpha_0\, \mathbf{g}_i\left(\boldsymbol{w}_i(0), \boldsymbol{\xi}_i(0)\right) - \beta_0 \sum_{j \in \mathcal{N}_i} a_{ij} \left(\hat{\boldsymbol{w}}_i^{(i)} - \hat{\boldsymbol{w}}_j^{(i)}\right)$
end for
for Iteration $k \geq 1$ **do**
  for Agent $i = 1$ to n **do**
    Sample $\boldsymbol{\xi}_i(k)$ & *compute* $g_i\left(\boldsymbol{w}_i(k), \boldsymbol{\xi}_i(k)\right)$
    Compute $\boldsymbol{e}_i(k) = \boldsymbol{w}_i(k) - \hat{\boldsymbol{w}}_i^{(i)}$
    if $\|\boldsymbol{e}_i(k)\|_1 \geq \upsilon_0\, \alpha_k$ **then**
      Send $\boldsymbol{w}_i(k)$ & *let* $\hat{\boldsymbol{w}}_i^{(i)} = \boldsymbol{w}_i(k)$
    end if
    if any $\boldsymbol{w}_j(k)$ received **then**
      Let $\hat{\boldsymbol{w}}_j^{(i)} = \boldsymbol{w}_j(k)$
    end if
    Update $\boldsymbol{w}_i(k+1) = \boldsymbol{w}_i(k) - \alpha_k\, \mathbf{g}_i\left(\boldsymbol{w}_i(k), \boldsymbol{\xi}_i(k)\right) - \beta_k \sum_{j \in \mathcal{N}_i} a_{ij} \left(\hat{\boldsymbol{w}}_i^{(i)} - \hat{\boldsymbol{w}}_j^{(i)}\right)$
  end for
end for

Our strategy for proving the convergence of the proposed distributed event-triggered SGD algorithm to a critical point is as follows. First we show that the consensus error among the agents are diminishing at the rate of $O\left(\frac{1}{(k+1)^{\delta_2}}\right)$ (see Theorem 1). Asymptotic convergence of the algorithm is then proved in Theorem 3. Theorem 4 then establishes that the weighted expected average gradient norm is a summable sequence. Convergence rate of the algorithm in the typical weak sense is given in Theorem 5. Finally, Theorem 6 proves the asymptotic mean-square convergence of the algorithm to a critical point.

Theorem 1. Consider the event-triggered SGD algorithm (6) under Assumptions 1-7. Then, there holds:

$$\mathbb{E}\left[\|\tilde{\mathbf{w}}_k\|_2^2\right] = O\left(\frac{1}{(k+1)^{\delta_2}}\right). \tag{17}$$

Proof of Theorem 1 is given in George and Gurram(2019). Define

$$\gamma_k = \frac{\alpha_k}{\beta_k} = \frac{a/b}{(k+1)^{\delta_2 - \delta_1}}. \tag{18}$$

Now define a non-negative function $V(\gamma_k, \mathbf{w}_k)$ as

$$V(\gamma_k, \mathbf{w}_k) = F(\mathbf{w}_k) + \frac{1}{2\gamma_k}\, \mathbf{w}_k^\top \left(\mathcal{L} \otimes I_{d_w}\right) \mathbf{w}_k. \tag{19}$$

Taking the gradient with respect to $\mathbf{w}_k$ yields

$$\nabla V(\gamma_k, \mathbf{w}_k) = \nabla F(\mathbf{w}_k) + \frac{1}{\gamma_k}\, \left(\mathcal{L} \otimes I_{d_w}\right) \mathbf{w}_k. \tag{20}$$

Theorem 2. Consider the distributed event-triggered SGD algorithm (6) under Assumptions 1-7. Then, for the gradient $\nabla V(\gamma_k, \mathbf{w}_k)$ given in (20), there holds:

$$\sum_{k=0}^{\infty} \alpha_k \mathbb{E}\left[\|\nabla V(\gamma_k, \mathbf{w}_k)\|_2^2\right] < \infty. \tag{21}$$

Proof. See Theorem 2 in George and Gurram(2019). □

Theorem 3. For the distributed event-triggered SGD algorithm (6) under Assumptions 1-7, we have

$$\sum_{k=0}^{\infty} \mathbb{E}\left[\|\mathbf{w}_{k+1} - \mathbf{w}_k\|_2^2\right] < \infty \quad \text{and} \tag{22}$$

$$\lim_{k \to \infty} \mathbb{E}\left[\|\mathbf{w}_{k+1} - \mathbf{w}_k\|_2^2\right] = 0. \tag{23}$$

See the supplementary material section of George and Gurram(2019) for the proof of Theorem 3. Define $\bar{\mathbf{w}}_k = \frac{1}{n}\left(\mathbf{1}_n \mathbf{1}_n^\top \otimes I_{d_w}\right) \mathbf{w}_k$ and $\overline{\nabla F}(\mathbf{w}_k) = \frac{1}{n}\left(\mathbf{1}_n \mathbf{1}_n^\top \otimes I_{d_w}\right) \nabla F(\mathbf{w}_k)$. Note that $\|\overline{\nabla F}(\mathbf{w}_k)\|_2^2 = \frac{1}{n}\| \left(\mathbf{1}_n^\top \otimes I_{d_w}\right) \nabla F(\mathbf{w}_k)\|_2^2 = \frac{1}{n}\| \sum_{i=1}^n \nabla f_i(\boldsymbol{w}_i(k))\|_2^2$.

Theorem 4. For the distributed event-triggered SGD algorithm (6) under Assumptions 1-7, we have

$$\sum_{k=0}^{\infty} \alpha_k\, \mathbb{E}\left[\left\|\overline{\nabla F}(\mathbf{w}_k)\right\|_2^2\right] < \infty. \tag{24}$$

Proof. See Theorem 4 in George and Gurram(2019). □

Theorem 4 establishes results about the weighted sum of expected average gradient norm and the key take-away from this result is that, for the distributed SGD in (8) or (6) with appropriate step-sizes, the expected average gradient norms cannot stay bounded away from zero (See Theorem 9 of (Bottou, Curtis, and Nocedal 2018)), i.e., $\liminf_{k\to\infty} \mathbb{E}\left[\left\|\overline{\nabla F}(\mathbf{w}_k)\right\|_2^2\right] = 0$ or equivalently $\liminf_{k\to\infty} \mathbb{E}\left[\|\sum_{i=1}^n \nabla f_i(\boldsymbol{w}_i(k))\|_2^2\right] = 0$. The rate of such weak convergence results can be obtained as shown in Theorem 5.

Theorem 5. Let $\{\mathbf{w}_k\}_{k=0}^K$ be generated according to the distributed event-triggered SGD given in (6) under Assumptions 1-7. Then for $\delta_2 = 1$ we have

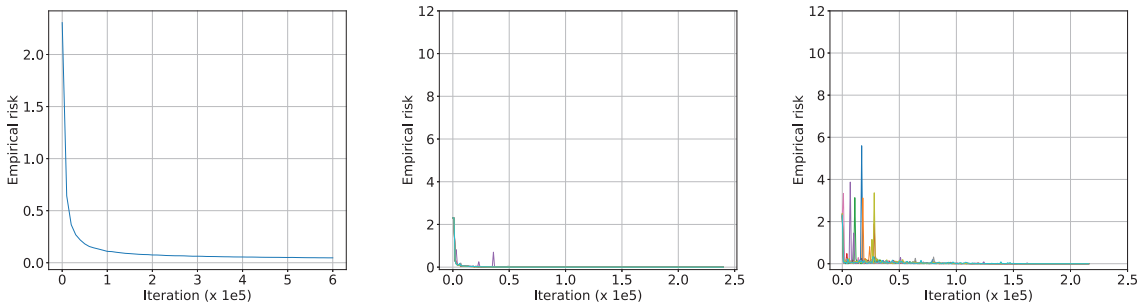
$$\mathbb{E}\left[\left\|\sum_{i=1}^n \nabla f_i(\, \boldsymbol{z}_i^K\,)\right\|_2^2\right] = O\left(\frac{1}{\log(K+1)}\right) \tag{25}$$

and for $\delta_2 \in (0.5,\ 1)$ we have

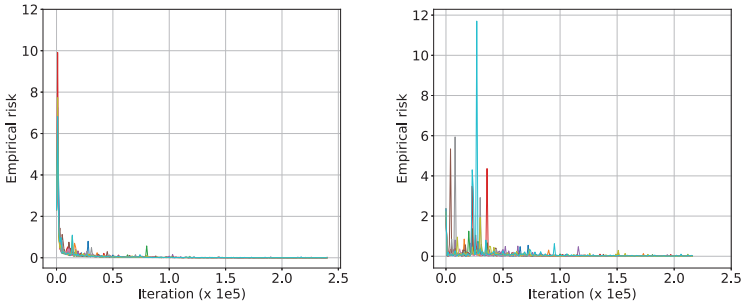
$$\mathbb{E}\left[\left\|\sum_{i=1}^n \nabla f_i(\, \boldsymbol{z}_i^K\,)\right\|_2^2\right] = O\left(\frac{1}{(K+1)^{1-\delta_2}}\right). \tag{26}$$

Here $\mathbf{z}^K \triangleq \left[(\boldsymbol{z}_1^K)^\top \ \ \ldots \ \ (\boldsymbol{z}_n^K)^\top\right]^\top$ is a random sample from $\{\mathbf{w}_k\}_{k=0}^K$ with probability $\mathbb{P}\left(\mathbf{z}^K = \mathbf{w}_k\right) = \frac{\alpha_k}{\sum_{j=0}^K \alpha_j}$.

Proof. See Theorem 5 in George and Gurram(2019). □

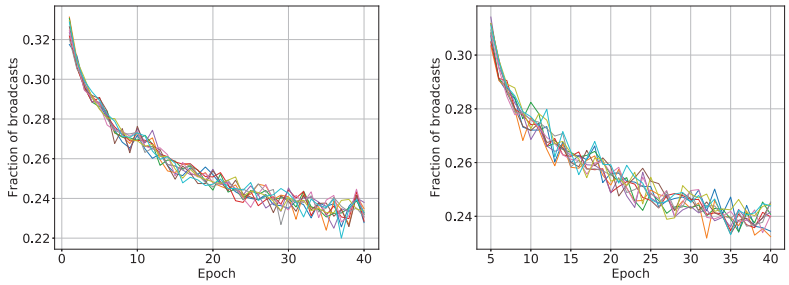


(a) Centralized SGD (b) Distributed SGD-r (c) Distributed SGD-s



(d) DETSGRAD-r (e) DETSGRAD-s

Figure 1: Empirical risk for all five experiments on MNIST dataset.



(a) DETSGRAD-r (b) DETSGRAD-s

Figure 2: Fraction of event-triggered broadcast events for the 10 agents compared to continuous broadcasting case for MNIST dataset.

Agent	1	2	3	4	5	6	7	8	9	10
Dist. SGD-r	98.97	98.97	98.97	98.97	98.97	98.97	98.97	98.97	98.97	98.97
Dist. SGD-s	98.86	98.86	98.86	98.87	98.86	98.86	98.86	98.87	98.85	98.87
DETSGRAD-r	98.34	98.35	98.32	98.27	98.31	98.31	98.38	98.29	98.23	98.33
DETSGRAD-s	98.46	98.49	98.49	98.51	98.5	98.45	98.13	98.49	98.42	98.51

Table 1: MNIST - Final classification accuracies (%) of the 10 agents after 40 epochs (240000 iterations for the random sampling/i.i.d. case and 216840 iterations for the single class/non-i.i.d case) using different algorithms. The final accuracy of a single agent using centralized SGD after 10 epochs (600000 iterations) is 98.63%.

Agent	1	2	3	4	5	6	7	8	9	10
DETSGRAD-r	61759	61455	61504	61636	61738	61822	61746	61712	61850	61795
DETSGRAD-s	71756	71718	71762	71983	71976	71773	71762	72159	72233	72208

Table 2: MNIST - Total number of event-triggered broadcast events for the 10 agents after 40 epochs. The total number of continuous broadcast events for each agent after 40 epochs is 240000 in the random sampling case, and 216840 in the single class case.

Finally, we present the following result to illustrate that stronger convergence results follows from the continuity assumption on the Hessian, which has not been utilized in our analysis so far.

Assumption 8. The Hessians $\nabla^2 f_i(\,\cdot\,) : \mathbb{R}^{d_w} \mapsto \mathbb{R}^{d_w \times d_w}$ are Lipschitz continuous with Lipschitz constants L_{H_i}, i.e., $\forall\, \boldsymbol{w}_a,\, \boldsymbol{w}_b \in \mathbb{R}^{d_w},\, i = 1, \ldots, n$, we have

$$\|\nabla^2 f_i(\,\boldsymbol{w}_a\,) - \nabla^2 f_i(\,\boldsymbol{w}_b\,)\|_2 \leq L_{H_i} \|\boldsymbol{w}_a - \boldsymbol{w}_b\|_2. \quad (27)$$

It follows from Assumption 8 that the Hessian $\nabla^2 F(\cdot)$ is Lipschitz continuous, i.e., $\forall\, \mathbf{w}_a,\, \mathbf{w}_b \in \mathbb{R}^{nd_w}$,

$$\|\nabla^2 F(\,\mathbf{w}_a\,) - \nabla^2 F(\,\mathbf{w}_b\,)\|_2 \leq L_H \|\mathbf{w}_a - \mathbf{w}_b\|_2, \quad (28)$$

with constant $L_H = \max_i \{L_{H_i}\}$.

Theorem 6. For the distributed SGD algorithm (6) under Assumptions 1-8 we have

$$\lim_{k\to\infty} \mathbb{E}\left[\, \left\|\overline{\nabla F}(\mathbf{w}_k)\right\|_2^2 \right] = 0 \quad \text{and} \quad (29)$$

$$\lim_{k\to\infty} \mathbb{E}\left[\left\| \sum_{i=1}^{n} \nabla f_i(\boldsymbol{w}_i(k)) \right\|_2^2 \right] = 0. \quad (30)$$

Proof. See Theorem 6 in George and Gurram(2019). □

Similar to the centralized SGD (Bottou, Curtis, and Nocedal 2018), the analysis given here shows the mean-square convergence of the distributed algorithm to a critical point, which include the saddle points. Though SGD has shown to escape saddle points efficiently (Lee et al. 2017; Fang, Lin, and Zhang 2019; Jin et al. 2019), extension of such results for distributed SGD is currently nonexistent and is a topic for future research.

Application to distributed supervised learning

We apply the proposed algorithm to distributedly train neural network agents for image classification task. We present extensive results on two different datasets - MNIST[1] and CIFAR-10[2].

MNIST

MNIST data set is a handwritten digit recognition data set containing 60000 grayscale images of 10 digits (0-9) for training and 10000 images are used for testing. We distributedly train 10 agents that are connected in an undirected unweighted ring topology. The 10-node ring was selected only since it is one of the least connected network (besides the path) and MNIST contains 10 classes. Proposed algorithm would work for any undirected graph as along as it is connected.

Each agent aims to train its own neural network, which is a randomly initialized LeNet-5 (LeCun et al. 1998). During training, each agent broadcasts its weights to its neighbors at every iteration or aperiodically as described in the proposed algorithm. Here we conduct the following five experiments: (i) Centralized SGD, where a centralized version of the SGD is implemented by a central node having access to all 60000 training images from all classes; (ii) Distributed SGD-r, where all the agents broadcast their respective weights at every iteration, and each agent has access to 6000 training images, *randomly* sampled from the entire training set, which forms the i.i.d. case; (iii) Distributed SGD-s, where all the agents broadcast their weights at every iteration, and each agent has access to the images corresponding to a *single* class, which forms the non-i.i.d. case; (iv) DETSGRAD-r, where the agents aperiodically broadcast their weights using the triggering mechanism in (10), and each agent has access to 6000 training images, *randomly* sampled from the entire training set, i.e., i.i.d. case; (v) DETSGRAD-s, where the agents aperiodically broadcast their weights using the triggering mechanism in (10), and each agent has access to the images corresponding to a *single* class, i.e., non-i.i.d. case. In the single class case, for ease of programming, we set the number of training images available for each agent to 5421 (the minimum number of training images available in a single class, which is digit 5 in MNIST data set). Here we select $\alpha_k = \frac{0.1}{(\varepsilon k+1)}$ and $\beta_k = \frac{0.2525}{(\varepsilon k+1)^{1/10}}$, where $\varepsilon = 10^{-5}$ for Distributed SGD and DETSGRAD. We select $\alpha_k = \frac{0.001}{(\varepsilon k+1)}$ for centralized SGD. Note that using a scale factor ε does not affect the theoretical results provided in the previous sections. For the DETSGRAD experiments, we select the broadcast event trigger threshold $\upsilon_0 = 0.2 \times N_{parameters}$, where $N_{parameters}$ is the total number of parameters in each neural network.

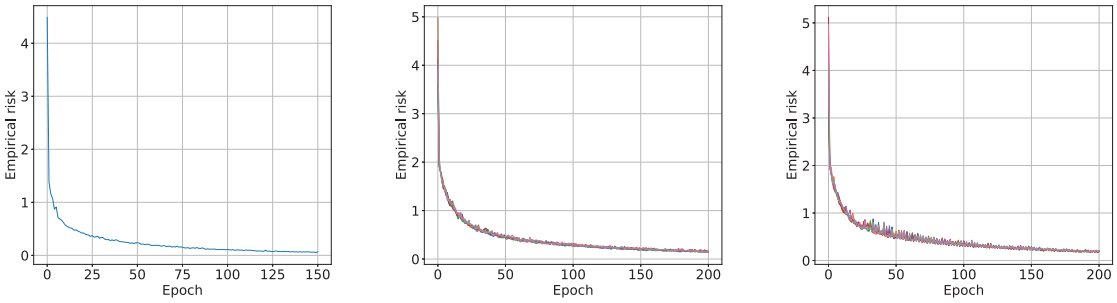
The plots of the empirical risk vs. the iterations (parameter update steps), illustrated in Figure 1, show the convergence of the proposed algorithm. The final test accuracies of the 10 agents after 40 training epochs using different algorithms and different training settings are shown in Table 1. Results obtained here indicate that regardless of how the data are distributed (random or single class), the agents are able to train their network and the distributedly trained networks are able to yield similar performance as that of a centrally trained network. More importantly, in the single class case, agents were

[1] http://yann.lecun.com/exdb/mnist/

[2] https://www.cs.toronto.edu/ kriz/cifar.html

Agent	1	2	3	4	5	6	7	8
Dist. SGD-r	85.02	85.04	84.88	84.93	85.2	85.7	84.47	84.92
DETSGRAD-r	84.96	84.84	84.42	84.37	84.46	84.86	84.84	84.6

Table 3: CIFAR-10 - Final classification accuracies (%) of the 8 agents after 200 epochs of training. The final accuracy of a single agent using centralized SGD after 150 epochs is 85.12%.



(a) Centralized SGD (b) Distributed SGD-r (c) DETSGRAD-r

Figure 3: Empirical risk vs. epoch for ResNet20 trained on CIFAR-10 dataset.

Agent	1	2	3	4	5	6	7	8
DETSGRAD-r	5484	5479	5471	5483	5483	5485	5493	5481

Table 4: CIFAR-10 - Total number of event-triggered broadcast events for the 8 agents after 200 epochs. The total number of continuous broadcast events for each agent after 200 epochs is 9800.

able to recognize images from all 10 classes even though they had access to data corresponding only to a single class during the training phase. This result has numerous implications for the machine learning community, specifically for federated multi-task learning under information flow constraints.

The total number of event-triggered parameter broadcast events for the 10 agents using the DETSGRAD algorithm are shown in Table 2. In the random sampling case, by employing broadcast event-triggering mechanism, we are able to reduce the inter-agent communications from 240000 to an average of 61702 over 40 epochs leading to a reduction of 74.2% in network communications. In the single class case, the agents broadcast the parameters continuously for the first 4 epochs, after which the event-trigger mechanism is started. Here, we are able to reduce the parameter broadcasts for each agent from 216840 to an average of 71933 over 40 epochs leading to a reduction of 66.8% in network communications. Yet, as can be seen in Table 1, DETSGRAD gives similar classification performance as distributed SGD with continuous parameter sharing with significant reduction in network communications. The fractions of the broadcast events for the 10 agents over 40 epochs are presented in Figure 2. As expected, the number of broadcast events reduces with the increase in epoch number as the agents converge to the critical point of the empirical risk function.

CIFAR-10

CIFAR-10 data set is an image classification data set containing 50000 color images of 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck) for training and 10000 images are used for testing. We distributedly train 8 agents that are connected in an undirected unweighted ring topology. Each agent trains its own neural network, which is a randomly initialized ResNet-20[3] (He et al. 2016). We conducted the following three experiments: (i) Centralized SGD, where a centralized version of the SGD is implemented by a central node having access to all 50000 training images from all classes; (ii) Distributed SGD-r, where all the agents broadcast their respective weights at every iteration, and each agent has access to 6250 training images, *randomly* sampled from the entire training set; (iii) DETSGRAD-r, where the agents aperiodically broadcast their weights using the triggering mechanism in (10), and each agent has access to 6250 training images, *randomly* sampled from the entire training set. Here we select $\alpha_k = \frac{0.1}{(\varepsilon k+1)}$ and $\beta_k = \frac{0.2525}{(\varepsilon k+1)^{1/10}}$, where $\varepsilon = 0.00025$ for Distributed SGD and DETSGRAD. We select $\alpha_k = \frac{0.1}{(\varepsilon k+1)}$, where $\varepsilon = 10^{-5}$ for centralized SGD. For the DETSGRAD experiment, we select the broadcast event trigger threshold $\upsilon_0 = 0.01 \times N_{parameters}$, where $N_{parameters}$ is the total number of parameters in each neural

[3]https://github.com/akamaster/pytorch_resnet_cifar10

network, with the agents broadcasting the parameters continuously for the first 25 epochs.

The plots of the empirical risk vs. epochs, illustrated in Figure 3, show the convergence of the proposed algorithm. The final test accuracies of the 8 agents after 200 training epochs using two different algorithms are shown in Table 3. Similar to the previous case, results obtained here indicate that the distributedly trained networks are able to yield similar performance as that of a centrally trained network. The total number of event-triggered parameter broadcast events for the 8 agents using the DETSGRAD algorithm are shown in Table 4. By employing broadcast event-triggering mechanism, we are able to reduce the inter-agent communications from 9800 to an average of 5482 over 200 epochs leading to a reduction of 44.1% in network communications. Yet, as can be seen in Table 3, DETSGRAD gives similar classification performance as distributed SGD with continuous parameter sharing with significant reduction in network communications.

Conclusion

This paper presented the development of a distributed stochastic gradient descent algorithm with event-triggered communication mechanism for solving non-convex optimization problems. We presented a novel communication triggering mechanism, which allowed the agents to decidedly reduce the communication overhead by communicating only when the local model has significantly changed from previously communicated model. We presented the sufficient conditions on algorithm step-sizes to guarantee asymptotic mean-square convergence of the proposed algorithm to a critical point and provided the convergence rate of the proposed algorithm. We applied the developed algorithm to distributed supervised-learning problem, in which a set of networked agents collaboratively train their individual neural nets to perform image classification. Results indicate that the distributedly trained networks are able to yield similar performance to that of a centrally trained network. Numerical results also show that the proposed event-triggered communication mechanism significantly reduced the inter-agent communication while yielding similar performance to that of a distributedly trained network with constant communication.

References

Agarwal, A., and Duchi, J. C. 2011. Distributed delayed stochastic optimization. In *NIPS*. 873–881.

Assran, M.; Loizou, N.; Ballas, N.; and Rabbat, M. 2019. Stochastic Gradient Push for Distributed Deep Learning. In *ICML*, 344–353.

Bianchi, P., and Jakubowicz, J. 2013. Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE TAC* 58(2):391–405.

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

Chaturapruek, S.; Duchi, J. C.; and Ré, C. 2015. Asynchronous stochastic convex optimization: the noise is in the noise and sgd don't care. In *NIPS*. 1531–1539.

Chatzipanagiotis, N., and Zavlanos, M. M. 2017. On the convergence of a distributed augmented lagrangian method for nonconvex optimization. *IEEE TAC* 62(9):4405–4420.

De Sa, C. M.; Zhang, C.; Olukotun, K.; Ré, C.; and Ré, C. 2015. Taming the wild: A unified analysis of hogwild-style algorithms. In *NIPS*. 2674–2682.

Duchi, J.; Jordan, M. I.; and McMahan, B. 2013. Estimation, optimization, and parallelism when data is sparse. In *NIPS*. 2832–2840.

Fang, C.; Lin, Z.; and Zhang, T. 2019. Sharp analysis for nonconvex sgd escaping from saddle points. arXiv:1902.00247.

Feyzmahdavian, H. R.; Aytekin, A.; and Johansson, M. 2016. An asynchronous mini-batch algorithm for regularized stochastic optimization. *IEEE TAC* 61(12):3740–3754.

George, J., and Gurram, P. 2019. Distributed Deep Learning with Event-Triggered Communication. *arXiv* 1909.05020.

Guo, J.; Hug, G.; and Tonguz, O. K. 2017. A case for nonconvex distributed optimization in large-scale power systems. *IEEE TPS* 32(5):3842 – 3851.

Haddadpour, F.; Kamani, M. M.; Mahdavi, M.; and Cadambe, V. 2019. Trading redundancy for communication: Speeding up distributed SGD for non-convex optimization. In *ICML*, 2545–2554.

Hajinezhad, D.; Hong, M.; Zhao, T.; and Wang, Z. 2016. Nestt: A nonconvex primal-dual splitting method for distributed and stochastic optimization. In *NIPS*. 3215–3223.

Hajinezhad, D.; Hong, M.; and Garcia, A. 2019. Zone: Zeroth order nonconvex multi-agent optimization over networks. *IEEE TAC*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *IEEE CVPR*, 770–778.

Hong, M.; Hajinezhad, D.; and Zhao, M.-M. 2017. Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks. In *ICML*, 1529 – 1538.

Hong, M.; Luo, Z.; and Razaviyayn, M. 2016. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM JO* 26(1):337–364.

Hong, M. 2018. A distributed, asynchronous, and incremental algorithm for nonconvex optimization: An admm approach. *IEEE TCNS* 5(3):935–945.

Huo, Z., and Huang, H. 2016. Asynchronous Stochastic Gradient Descent with Variance Reduction for Non-Convex Optimization. arXiv:1604.03584.

Jakovetic, D.; Bajovic, D.; Sahu, A. K.; and Kar, S. 2018. Convergence rates for distributed stochastic optimization over random networks. In *IEEE CDC*, 4238–4245.

Jiang, Z.; Balu, A.; Hegde, C.; and Sarkar, S. 2017. Collaborative deep learning in fixed topology networks. In *NIPS*. 5904–5914.

Jin, C.; Netrapalli, P.; Ge, R.; Kakade, S. M.; and Jordan, M. I. 2019. Stochastic gradient descent escapes saddle points efficiently. arXiv:1902.04811.

Konečnú, J.; McMahan, H. B.; Yu, F. X.; Richtarik, P.; Suresh, A. T.; and Bacon, D. 2016. Federated learning: Strategies for improving communication efficiency. In *NIPSW*.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.; et al. 1998. Gradient-based learning applied to document recognition. *Proc. of the IEEE* 86(11):2278–2324.

Lee, J. D.; Panageas, I.; Piliouras, G.; Simchowitz, M.; Jordan, M. I.; and Recht, B. 2017. First-order methods almost always avoid saddle points. arXiv:1710.07406.

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

Li, M.; Andersen, D. G.; Smola, A. J.; and Yu, K. 2014b. Communication efficient distributed machine learning with the parameter server. In *NIPS*. 19–27.

Lian, X.; Huang, Y.; Li, Y.; and Liu, J. 2015. Asynchronous Parallel Stochastic Gradient for Nonconvex Optimization. arXiv:1506.08272.

Lian, X.; Zhang, C.; Zhang, H.; Hsieh, C.-J.; Zhang, W.; and Liu, J. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *NIPS*. 5330–5340.

Lian, X.; Zhang, W.; Zhang, C.; and Liu, J. 2018. Asynchronous decentralized parallel stochastic gradient descent. In *ICML*, 3043–3052.

Lin, T.; Stich, S. U.; and Jaggi, M. 2018. Don't use large mini-batches, use local SGD. *arXiv* 1808.07217.

Lorenzo, P. D., and Scutari, G. 2016. NEXT: In-network nonconvex optimization. *IEEE TSIPN* 2(2):120–136.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*.

Mitliagkas, I.; Borokhovich, M.; Dimakis, A. G.; and Caramanis, C. 2015. Frogwild!: Fast pagerank approximations on graph engines. *Proc. VLDB Endow.* 8(8):874–885.

Nedić, A., and Olshevsky, A. 2016. Stochastic gradient-push for strongly convex functions on time-varying directed graphs. *IEEE TAC* 61(12):3936–3947.

Noel, C., and Osindero, S. 2014. Dogwild!-distributed hogwild for cpu & gpu. In *NIPSW*.

Pu, S., and Nedić, A. 2018. Distributed Stochastic Gradient Tracking Methods. arXiv:1805.11454.

Recht, B.; Re, C.; Wright, S.; and Niu, F. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*. 693–701.

Scutari, G.; Facchinei, F.; and Lampariello, L. 2017. Parallel and distributed methods for constrained nonconvex optimization–part i: Theory. *IEEE TSP* 65(8):1929 – 1944.

Tang, H.; Lian, X.; Yan, M.; Zhang, C.; and Liu, J. 2018. d^2: Decentralized training over decentralized data. In *ICML*, 4848–4856.

Tatarenko, T., and Touri, B. 2017. Non-convex distributed optimization. *IEEE TAC* 62(8):3744 – 3757.

Wai, H.; Freris, N. M.; Nedic, A.; and Scaglione, A. 2018. Sucag: Stochastic unbiased curvature-aided gradient method for distributed optimization. In *IEEE CDC*, 1751–1756.

Wang, J., and Joshi, G. 2018. Cooperative SGD: A unified Framework for the Design and Analysis of Communication-Efficient SGD Algorithms. arXiv:1808.07576.

Wang, J.; Sahu, A. K.; Yang, Z.; Joshi, G.; and Kar, S. 2019. MATCHA: Speeding Up Decentralized SGD via Matching Decomposition Sampling. arXiv:1905.09435.

Yu, H., and Jin, R. 2019. On the computation and communication complexity of parallel SGD with dynamic batch sizes for stochastic non-convex optimization. In *ICML*, 7174–7183.

Yu, H.; Hsieh, C.; Si, S.; and Dhillon, I. 2012. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *IEEE ICDM*, 765–774.

Yu, H.; Jin, R.; and Yang, S. 2019. On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. In *ICML*, 7184–7193.

Yu, H.; Yang, S.; and Zhu, S. 2019. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *AAAI-19*, 5693–5700.

Zeng, J., and Yin, W. 2018. On nonconvex decentralized gradient descent. *IEEE TSP* 66(11):2834–2848.

Zhang, K.; Alqahtani, S.; and Demirbas, M. 2017. A comparison of distributed machine learning platforms. In *ICCCN*, 1–9.

Zhang, J.; Tu, H.; Ren, Y.; Wan, J.; Zhou, L.; Li, M.; and Wang, J. 2018. An adaptive synchronous parallel strategy for distributed machine learning. *IEEE Access* 6:19222–19230.

Zhou, Z.; Mertikopoulos, P.; Bambos, N.; Glynn, P.; Ye, Y.; Li, L.-J.; and Fei-Fei, L. 2018. Distributed asynchronous optimization with unbounded delays: How slow can you go? In *ICML*, 5970–5979.

Zhu, M., and Martínez, S. 2013. An approximate dual subgradient algorithm for multi-agent non-convex optimization. *IEEE TAC* 58(6):1534 – 1539.