On the Convergence of Decentralized Adaptive Gradient Methods

Xiangyi Chen XIANGYICHEN1900@GMAIL.COM
Belhal Karimi BELHAL.KARIMI@GMAIL.COM
Weijie Zhao ZHAOWEIJIE12@GMAIL.COM
Ping Li PINGLI98@GMAIL.COM
Cognitive Computing Lab, Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA

Editors: Emtiyaz Khan and Mehmet Gönen

Abstract

[1]Adaptive gradient methods including Adam, AdaGrad, and their variants have been very successful for training deep learning models, such as neural networks. Meanwhile, given the need for distributed computing, distributed optimization algorithms are rapidly becoming a focal point. With the growth of computing power and the need for using machine learning models on mobile devices, the communication cost of distributed training algorithms needs careful consideration. In this paper, we introduce novel convergent decentralized adaptive gradient methods and rigorously incorporate adaptive gradient methods into decentralized training procedures. Specifically, we propose a general algorithmic framework that can convert existing adaptive gradient methods to their decentralized counterparts. In addition, we thoroughly analyze the convergence behavior of the proposed algorithmic framework and show that if a given adaptive gradient method converges, under some specific conditions, then its decentralized counterpart is also convergent. We illustrate the benefit of our generic decentralized framework on prototype methods, AMSGrad and AdaGrad.

Keywords: Decentralized, Adaptive, Gradient, Convergence, Optimization.

1. Introduction

Distributed training of machine learning models is drawing growing attention in the past few years due to its practical benefits and necessities, for example, for training massive-scale CTR (click-through rate) models in commercial advertising (Zhao et al., 2019). Given the evolution of computing capabilities of CPUs and GPUs, computation time in distributed setting is gradually being dominated by the communication time under many circumstances (Chilimbi et al., 2014; McMahan et al., 2017; Zhao et al., 2022). As a result, a large amount of recent works has been focussing on reducing the communication cost for distributed learning (Alistarh et al., 2017; Lin et al., 2018; Wangni et al., 2018; Stich et al., 2018; Wang et al., 2018; Tang et al., 2019). In the traditional parameter (central) server setting, where a parameter server is employed to manage the communication in the whole network, many effective communication reduction techniques have been proposed based on gradient (and/or weight) compression and quantization (Aji and Heafield, 2017; Xu et al., 2021; Li et al., 2022). Despite these communication reduction techniques, its cost usually still scales linearly with the number of workers. Due to this limitation and with the sheer size of decentralized devices, the *decentralized training paradigm* (Duchi et al., 2012) is drawing a lot of attention. In this paradigm, the parameter server is removed and each node only communicates with its neighbors. It is shown

1. The work was conducted and initially submitted in 2019.

in Lian et al. (2017) that, when the training bottleneck is the communication cost, decentralized training algorithms can outperform parameter server-based procedures. The decentralized paradigm is also preferred when a central parameter server is by construction not available.

In light of recent advances in nonconvex optimization, an effective way to accelerate training is by using adaptive gradient methods like AdaGrad (Duchi et al., 2011), Adam (Kingma and Ba, 2015), AMSGrad (Reddi et al., 2018), or OPT-AMSGrad (Wang et al., 2021). Their popularity are due to their practical benefits in training neural networks, featured by faster convergence and ease of parameter tuning compared with Stochastic Gradient Descent (SGD) (Robbins and Monro, 1951). Several works have applied adaptive gradient methods to distributed training systems. Reddi et al. (2021) develop a distributed adaptive optimization framework with applications to federated learning. Their method employs local SGD computations and a global adaptive learning rate step. Li et al. (2022) study distributed training with AMSGrad under communication compression. The work of Chen et al. (2020) propose an algorithm called *local AMSGrad*, where local workers perform adaptive gradient updates, instead of local SGD, and a global server aggregates the local model updates. Also see the extension to *local Adam* (Zhao et al., 2022) and its application in training commercial ads CTR models. In these papers, all the training systems exhibit a server/worker communication structure. On the contrary, in the setting of our paper, *nodes can only communicate to their neighbors* on a fixed communication graph. Designing adaptive methods in such setting is highly non-trivial due to the already complex update rules and to the interaction between the effect of using adaptive learning rates and the decentralized communication protocols. Therefore, while our work can be partially viewed as an extension of Chen et al. (2020) to the decentralized scenario, our algorithm design and analysis involve additional challenges and techniques.

Contributions. This paper attempts to bridge the gap between two realms in nonconvex optimization:

- We investigate the application of adaptive gradient methods in the decentralized training paradigm, where nodes have only a local view of the whole communication graph. We develop a general technique that converts a *centralized* adaptive gradient method to its *decentralized* counterpart and highlight the importance of adaptive learning rate consensus.
- By using our proposed technique, we present a new decentralized optimization algorithm, called decentralized AMSGrad, as the decentralized counterpart of AMSGrad.
- We provide a theoretical verification interface, in Theorem 2, for analyzing the behavior of decentralized adaptive gradient methods obtained via our technique. Thus, we characterize the convergence rate of decentralized AMSGrad, which is, to the best of our knowledge, the first convergent decentralized adaptive gradient method. In particular, our Theorem 3 provides a convergence rate of order $\mathcal{O}(\sqrt{d}/\sqrt{T})$, for the decentralized AMSGrad, when the number of iterations T is large, hence matching state-of-the-art rates.
- Similar analysis and bounds are provided for the decentralized counterpart of AdaGrad.

In our algorithm, one key ingredient is the *consensus on adaptive learning rates* at different nodes, which is also employed in Chen et al. (2020); Karimi et al. (2021). We show the importance of consensus on adaptive learning rates in the algorithm design by proving a divergent problem instance for a recently proposed decentralized adaptive gradient method, namely DADAM (Nazari et al., 2019), a decentralized version of ADAM. Though consensus is performed on the model parameter, DADAM lacks consensus principles on the adaptive learning rates.

Organization. An overview of prior works is provided in Section 2. In Section 3, we show the importance of adaptive learning rate consensus. In Section 4, we develop our general framework for converting adaptive gradient methods into their decentralized counterparts along with convergence analysis and converted algorithms. Illustrative experiments are presented in Section 5.

Notations. $x_{t,i}$ denotes variable x at node i and iteration t. $\|\cdot\|_{abs}$ denotes the entry-wise L_1 matrix norm, i.e. $\|A\|_{abs} = \sum_{i,j} |A_{i,j}|$. We introduce important notations used throughout the paper: for any $t > 0$, $G_t := [g_{t,N}]$ where $[g_{t,N}]$ denotes the matrix $[g_{t,1}, g_{t,2}, \cdots, g_{t,N}]$ (where $g_{t,i}$ is a column vector), $M_t := [m_{t,N}]$, $X_t := [x_{t,N}]$, $\overline{\nabla f}(X_t) := \frac{1}{N}\sum_{i=1}^N \nabla f_i(x_{t,i})$, $U_t := [u_{t,N}]$, $\tilde{U}_t := [\tilde{u}_{t,N}]$, $V_t := [v_{t,N}]$, $\hat{V}_t := [\hat{v}_{t,N}]$, $\overline{X}_t := \frac{1}{N}\sum_{i=1}^N x_{t,i}$, $\overline{U}_t := \frac{1}{N}\sum_{i=1}^N u_{t,i}$ and $\overline{\tilde{U}_t} := \frac{1}{N}\sum_{i=1}^N \tilde{u}_{t,i}$.

2. Preliminaries

2.1. Related Work

Decentralized optimization. Traditional decentralized optimization methods include well-know algorithms such as ADMM (Boyd et al., 2011), Dual Averaging (Duchi et al., 2012), Distributed Subgradient Descent (Nedic and Ozdaglar, 2009). More recent algorithms include EXTRA (Shi et al., 2015), Next (Lorenzo and Scutari, 2016), Prox-PDA (Hong et al., 2017), GNSD (Lu et al., 2019), and Choco-SGD (Koloskova et al., 2019). While these algorithms are commonly used in applications other than deep learning, recent algorithmic advances in the machine learning community have shown that decentralized optimization can also be useful for training deep models such as neural networks. Lian et al. (2017) demonstrate that a stochastic version of Decentralized Subgradient Descent can outperform parameter-server-based algorithms when the communication cost is high. Tang et al. (2018) propose the D^2 algorithm improving the convergence rate over Stochastic Subgradient Descent. Assran et al. (2019) develop the Stochastic Gradient Push that is more robust to network failures for training neural networks. The study of decentralized training algorithms in the machine learning community is at its early stage. One noteworthy work (Nazari et al., 2019) presents a decentralized version of AMSGrad (Reddi et al., 2018) and it is proven to satisfy some non-standard regret.

Adaptive gradient methods. Adaptive gradient methods have been popular in recent years due to their superior performance in training neural networks. Most commonly used adaptive methods include AdaGrad (Duchi et al., 2011) or Adam (Kingma and Ba, 2015) and their variants. Key features of such methods lie in the use of momentum and adaptive learning rates (which means that the learning rate is changing during the optimization and is anisotropic, i.e. depends on the dimension). Adam has been analyzed in Reddi et al. (2018) where the authors point out an error in previous convergence analyses. A variety of papers have been focusing on analyzing the convergence behavior of the numerous existing adaptive gradient methods. Ward et al. (2019), Li and Orabona (2019) derive convergence guarantees for a variant of AdaGrad without coordinate-wise learning rates. Chen et al. (2019) analyze the convergence behavior of a broad class of algorithms including AMSGrad and AdaGrad. Zhou et al. (2018) give a more refined analysis of AMSGrad with better convergence rate. Zou and Shen (2018) provide a unified convergence analysis for AdaGrad with momentum. Chen et al. (2020); Karimi et al. (2021); Zhao et al. (2022) propose locally adaptive algorithms in distributed training, and its layerwise accelerated variant and application to real-world ads CTR models. Li et al. (2022) analyze the convergence of distributed AMSGrad when the communication between worker and server is compressed. More recent works on adaptive methods can be found in Zaheer et al. (2018); Agarwal et al. (2019); Luo et al. (2019); Zhou et al. (2020).

2.2. Decentralized Optimization Framework

In distributed optimization (with N nodes), we aim at solving the following optimization problem

$$\min_{x\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} f_i(x)\,, \tag{1}$$

where x denotes the vector of parameters and f_i is only accessible by the ith node. Through the prism of empirical risk minimization procedures, f_i can be viewed as the average loss of the data samples located at node i, for all $i \in [N]$. We make the following mild assumptions required for analyzing the convergence behavior of the different decentralized optimization algorithms:

A1 *For all $i \in [N]$, f_i is differentiable and the gradients are L-Lipschitz, i.e., for all $(x, y) \in \mathbb{R}^d$, $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$.*

A2 *We assume that, at iteration t, node i accesses a stochastic gradient noted $g_{t,i}$. The stochastic gradients and the gradients of f_i have bounded L_∞ norms, i.e. $\|g_{t,i}\| \leq G_\infty$, $\|\nabla f_i(x)\|_\infty \leq G_\infty$.*

A3 *The gradient estimators are unbiased and has bounded variance for each coordinate, i.e. $\mathbb{E}[g_{t,i}] = \nabla f_i(x_{t,i})$ and $\mathbb{E}[([g_{t,i} - f_i(x_{t,i})]_j)^2] \leq \sigma^2$, for all t, i, j .*

Assumptions A1 and A3 are standard in distributed optimization literature. A2 is slightly stronger than the traditional bounded variance assumption, but is commonly used for the analysis of adaptive gradient methods (Chen et al., 2019; Ward et al., 2019). Note that the bounded gradient estimator assumption in A2 implies the bounded variance assumption in A3. We willingly denote the variance bound and the estimator bound differently to avoid confusion when we use them for different purposes. In decentralized optimization, the nodes are connected as a graph and each node only communicates to its neighbors. In such a case, one usually constructs a $N \times N$ matrix W for information sharing for the design of novel algorithms. We denote λ_i to be its ith largest eigenvalue and define $\lambda \triangleq \max(|\lambda_2|, |\lambda_N|)$. We assume the following for the matrix W:

A4 *The matrix W satisfies:* (I) $\sum_{j=1}^{N} W_{i,j} = 1$, $\sum_{i=1}^{N} W_{i,j} = 1$, $W_{i,j} \geq 0$, (II) $\lambda_1 = 1$, $|\lambda_2| < 1$, $|\lambda_N| < 1$ *and* (III) $W_{i,j} = 0$ *if node i and node j are not neighbors.*

3. Decentralized Optimization and Adaptive Gradient Methods

We first present the convergence failure of current decentralized adaptive method, such as DADAM (Nazari et al., 2019), then introduce our general framework for decentralized adaptive methods.

3.1. Divergence of DADAM

Recently, Nazari et al. (2019) initiated an attempt to bring adaptive gradient methods into the decentralized optimization realm by introducing Decentralized ADAM (DADAM), described in Algorithm 1. DADAM is essentially a decentralized version of ADAM and the key modification consists in using a consensus step on the optimization variable x to transmit information across the network, encouraging its convergence. The matrix W is a doubly stochastic matrix, which satisfies A4, for achieving average consensus of x. Introducing this mixing matrix is standard for decentralized algorithm, such as distributed gradient descent (Nedic and Ozdaglar, 2009; Yuan et al., 2016). It is

proven in Nazari et al. (2019) that DADAM admits a non-standard regret bound in the online setting. Nevertheless, whether the algorithm can converge to stationary points in standard offline settings such as training neural networks is still unknown. The convergence failure of DADAM in the offline settings is established below:

Theorem 1 *There exists a problem satisfying the assumptions A1-A4 where DADAM fails to converge to a stationary point such that* $\nabla f(\bar{X}_t) = 0$.

Proof Consider a two-node setting with objective function $f(x) = 1/2\sum_{i=1}^{2} f_i(x)$ and $f_1(x) = \mathbb{1}[|x| \leq 1]2x^2 + \mathbb{1}[|x| > 1](4|x| - 2)$, $f_2(x) = \mathbb{1}[|x-1| \leq 1](x-1)^2 + \mathbb{1}[|x-1| > 1](2|x-1| - 1)$. We set the mixing matrix $W = [0.5, 0.5; 0.5, 0.5]$. The optimal solution is $x^* = 1/3$. Both f_1 and f_2 are smooth and convex with bounded gradient norm 4 and 2, respectively. We also have $L = 4$ (defined in A1). If we initialize with $x_{1,1} = x_{1,2} = -1$ and run DADAM with $\beta_1 = \beta_2 = \beta_3 = 0$ and $\epsilon \leq 1$, we will get $\hat{v}_{1,1} = 16$ and $\hat{v}_{1,2} = 4$. Since $|g_{t,1}| \leq 4, |g_{t,2}| \leq 2$ due to bounded gradient, and $(\hat{v}_{t,1}, \hat{v}_{t,2})$ are non-decreasing, we have $\hat{v}_{t,1} = 16, \hat{v}_{t,2} = 4$, for all $t \geq 1$. Thus, after $t = 1$, DADAM is equivalent to running decentralized gradient descent (D-PSGD) (Yuan et al., 2016) with a re-scaled f_1 and f_2, *i.e.* running D-PSGD on $f'(x) = \sum_{i=1}^{2} f_i'(x)$ with $f_1'(x) = 0.25 f_1(x)$ and $f_2'(x) = 0.5 f_2(x)$, which unique optimal $x' = 0.5$. Define $\bar{x}_t = (x_{t,1} + x_{t,2})/2$, then by Theorem 2 in Yuan et al. (2016), we have when $\alpha < 1/4$, $f'(\bar{x}_t) - f(x') = O(1/(\alpha t))$. Since f' has a unique optima x', the above bound implies $\bar{x}_t$ is converging to $x' = 0.5$ which has non-zero gradient on function $\nabla f(0.5) = 0.5$. ■

Theorem 1 shows that, even though DADAM is proven to satisfy some regret bounds (Nazari et al., 2019), it can fail to converge to stationary points in the non-convex offline setting, commonly used for training neural networks. We conjecture that this inconsistency in the convergence behavior of DADAM is due to the definition of the regret in Nazari et al. (2019). The next section presents a unifying decentralized adaptive gradient framework alongside a characterization of a finite-time and independent of the initialization convergence to some stationary point. In this section, we discuss the difficulties of designing adaptive gradient methods for decentralized optimization and introduce an algorithmic framework that can turn existing convergent adaptive gradient methods to their decentralized counterparts.

Algorithm 1 DADAM (with N nodes)

Input: α, current point X_t, $u_{\frac{1}{2},i} = \hat{v}_{0,i} = \epsilon\mathbf{1}$, $m_0 = 0$ and mixing matrix W
for $t = 1, 2, \cdots, T$ **do**
  for all $i \in [N]$ **do in parallel**
    $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
    $m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i}$
    $v_{t,i} = \beta_2 v_{t-1,i} + (1-\beta_2) g_{t,i}^2$
    $\hat{v}_{t,i} = \beta_3 \hat{v}_{t,i} + (1-\beta_3)\max(\hat{v}_{t-1,i}, v_{t,i})$
    $x_{t+\frac{1}{2},i} = \sum_{j=1}^{N} W_{ij} x_{t,j}$
    $x_{t+1,i} = x_{t+\frac{1}{2},i} - \alpha \frac{m_{t,i}}{\sqrt{\hat{v}_{t,i}}}$
end for

3.2. Importance and Difficulties of Consensus on Adaptive Learning Rates

The divergent example in the previous section implies that we should synchronize the adaptive learning rates on different nodes. This can be easily achieved in the parameter server setting where all the nodes are sending their gradients to a central server at each iteration. The parameter server can then exploit the received gradients to maintain a sequence of synchronized adaptive learning

rates when updating the parameters, see Reddi et al. (2021) or Chen et al. (2021) for respectively a federated and distributed variant of ADAM. However, in our decentralized setting, every node can *only communicate with its neighbors* and such central server does not exist, hence Reddi et al. (2021) and Chen et al. (2021) are not applicable here. Under our setting, the information for updating the adaptive learning rates can only be shared locally instead of broadcast over the whole network. This makes it impossible to obtain, in one iteration, a synchronized adaptive learning rate update using all the information of the network.

– *Systemic Approach:* On a systemic level, one way to alleviate this bottleneck is to design communication protocols in order to give each node access to the same aggregated gradients over the whole network, at least periodically if not at every iteration. Therefore, the nodes can update their individual adaptive learning rates based on the same shared information. However, such a solution may introduce an extra communication cost since it involves broadcasting the information over the whole network.

– *Algorithmic Approach:* Our contributions being on an algorithmic level, another way to solve the aforementioned problem is by letting the sequences of adaptive learning rates, present on different nodes, to gradually *consent*, through the iterations. Intuitively, if the adaptive learning rates can consent fast enough, the difference among the adaptive learning rates on different nodes will not affect the convergence behavior of the algorithm. Consequently, no extra communication costs need to be introduced. We now develop this exact idea within the existing adaptive methods stressing on the need for a relatively low-cost and easy-to-implement consensus of adaptive learning rates. Below is the main archetype of the adaptive rates consensus mechanism within a decentralized framework.

4. A Unifying Decentralized Adaptive Gradient Framework

While each node can have different $\hat{v}_{t,i}$ in DADAM (Algorithm 1), one can keep track of the min/max/average of these learning rates and use that quantity as the new adaptive learning rate. The predefinition of some convergent lower and upper bounds may also lead to a gradual synchronization of the adaptive learning rates on different nodes as developed in Luo et al. (2019) for AdaBound. In this paper, we present an algorithmic framework for decentralized adaptive gradient methods, see Algorithm 2, which uses average consensus of $\hat{v}_{t,i}$ (see consensus update in line 8 and 11) to help with the convergence. When choosing $\hat{v}_{t,i} = \frac{1}{t}\sum_{k=1}^{t} g_{k,i}^2$, Algorithm 2 becomes a decentralized version of AdaGrad or if $\hat{v}_{t,i}$ is the adaptive learning rate for AMSGrad, we obtain the decentralized AMSGrad (Algorithm 3). The intuition of using average consensus is that for adaptive gradient methods such as AdaGrad or Adam, $\hat{v}_{t,i}$ approximates the second moment of the gradient estimator, the average of the estimations of those second moments from different nodes is an estimation of the second moment on the whole network. This design does

Algorithm 2 Decentralized Adaptive Gradient Method (with N nodes)

1: **Input:** α, initial point $x_{1,i} = x_{init}, u_{\frac{1}{2},i} = \hat{v}_{0,i}, m_{0,i} = 0$, mixing matrix W
2: **for** $t = 1, 2, \cdots, T$ **do**
3: **for all** $i \in [N]$ **do in parallel**
4: $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
5: $m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i}$
6: $\hat{v}_{t,i} = r_t(g_{1,i}, \cdots, g_{t,i})$
7: $x_{t+\frac{1}{2},i} = \sum_{j=1}^{N} W_{ij} x_{t,j}$
8: $\tilde{u}_{t,i} = \sum_{j=1}^{N} W_{ij} \tilde{u}_{t-\frac{1}{2},j}$
9: $u_{t,i} = \max(\tilde{u}_{t,i}, \epsilon)$
10: $x_{t+1,i} = x_{t+\frac{1}{2},i} - \alpha \frac{m_{t,i}}{\sqrt{u_{t,i}}}$
11: $\tilde{u}_{t+\frac{1}{2},i} = \tilde{u}_{t,i} - \hat{v}_{t-1,i} + \hat{v}_{t,i}$
12: **end for**

not introduce any extra hyperparameters that can potentially complicate the tuning process (ϵ in line 9 is important for numerical stability as in vanilla Adam). The following result gives a finite-time convergence rate for our framework in Algorithm 2.

Theorem 2 *Assume A1-A4. When $\alpha \leq \frac{\epsilon^{0.5}}{16L}$, Algorithm 2 yields the following regret bound*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\frac{\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\right\|^2\right] \leq C_1\left(\frac{1}{T\alpha}(\mathbb{E}[f(Z_1)]-\min_x f(x))+\alpha\frac{d\sigma^2}{N}\right)+C_2\alpha^2 d \\ +C_3\alpha^3 d+\frac{1}{T\sqrt{N}}(C_4+C_5\alpha)\mathbb{E}\left[\sum_{t=1}^{T}\|(-\hat{V}_{t-2}+\hat{V}_{t-1})\|_{abs}\right], \tag{2}$$

where $\|\cdot\|_{abs}$ denotes the entry-wise L_1 norm of a matrix (i.e $\|A\|_{abs}=\sum_{i,j}|A_{ij}|$). The constants $C_1=\max(4,4L/\epsilon)$, $C_2=6((\beta_1/(1-\beta_1))^2+1/(1-\lambda)^2)LG_\infty^2/\epsilon^{1.5}$, $C_3=16L^2(1-\lambda)G_\infty^2/\epsilon^2$, $C_4=2/(\epsilon^{1.5}(1-\lambda))(\lambda+\beta_1/(1-\beta_1))G_\infty^2$, $C_5=2/(\epsilon^2(1-\lambda))L(\lambda+\beta_1/(1-\beta_1))G_\infty^2+4/(\epsilon^2(1-\lambda))LG_\infty^2$ are independent of d, T and N. In addition, $\frac{1}{N}\sum_{i=1}^{N}\|x_{t,i}-\overline{X}_t\|^2\leq\alpha^2\left(\frac{1}{1-\lambda}\right)^2 dG_\infty^2\frac{1}{\epsilon}$ which quantifies the consensus error.

It is readily seen that the last term on the RHS of (2) is the key to ensure convergence of converted algorithm since it may not diminish with changing α or T. The growth of this term depends mostly on the update rule of adaptive learning rate of the original algorithms before conversion. More study on the growth rate of this term can be found in Chen et al. (2019). Another point worth mentioning is that $\alpha\leq\frac{\epsilon^{0.5}}{16L}$ is to guarantee descent for worst case of the class of algorithms (which has $\overline{U}_t=\epsilon I,\forall t$), typically this is not required since for most adaptive algorithms, $\overline{U}_t$ will grow quickly to be way larger than ϵ. Interest readers may refer to the final step of the proof.

One can specify α to express convergence in terms of T, d, and N. An immediate result, shown in Corollary 1, is derived by setting $\alpha:=\sqrt{N}/\sqrt{Td}$:

Corollary 1 *Assume A1-A4. Set $\alpha=\sqrt{N}/\sqrt{Td}$. When $\alpha\leq\frac{\epsilon^{0.5}}{16L}$, Algorithm 2 yields:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\frac{\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\right\|^2\right] \leq C_1\frac{\sqrt{d}}{\sqrt{TN}}\left(\mathbb{E}[f(Z_1)]-\min_x f(x)+\sigma^2\right)+C_2\frac{N}{T} \\ +C_3\frac{N^{1.5}}{T^{1.5}d^{0.5}}+\left(C_4\frac{1}{T\sqrt{N}}+C_5\frac{1}{T^{1.5}d^{0.5}}\right)\mathbb{E}\left[\mathcal{V}_T\right], \tag{3}$$

where $\mathcal{V}_T:=\sum_{t=1}^{T}\|(-\hat{V}_{t-2}+\hat{V}_{t-1})\|_{abs}$ and C_1, C_2, C_3, C_4, C_5 are defined in Theorem 2.

Corollary 1 indicates that if $\mathbb{E}[\mathcal{V}_T]=o(T)$ and if $\bar{U}_t$ is bounded from above, then Algorithm 2 is guaranteed to converge to stationary points of the objective loss function. Intuitively, this means that if the adaptive learning rates on different nodes do not change drastically, the algorithm converges. In the convergence analysis, the term $\mathbb{E}[\mathcal{V}_T]$ upper bounds the total bias in the updated direction caused by the correlation between $m_{t,i}$ and $\hat{v}_{t,i}$. It is shown in Chen et al. (2019) that when $N=1$, $\mathbb{E}[\mathcal{V}_T]=\tilde{O}(d)$ for AdaGrad and AMSGrad. Besides, $\mathbb{E}[\mathcal{V}_T]=\tilde{O}(Td)$ for Adam which does not converge. Later, we will show the convergence of decentralized versions of AMSGrad and AdaGrad by bounding this latter term by $O(Nd)$ and $O(Nd\log(T))$, respectively. The intuition behind the

fact that $\mathbb{E}[\mathcal{V}_T] = o(T)$ can guarantee divergence is that the correlation between $\hat{v}_{t,i}$ and $m_{t,i}$ (due to their shared dependency on past gradients) can make update direction negatively correlated with true gradient in expectation, leading to a non-negligible bias in the updates. However, the total bias across T iterations introduced by such a correlation is bounded by the term $\mathbb{E}[\mathcal{V}_T]$. Hence, if $\mathbb{E}[\mathcal{V}_T]$ grows sublinearly with T, convergence can still be guaranteed. Corollary 1 also conveys the benefits of using more nodes in the graph. When T is large enough such that the term $O(\sqrt{d}/\sqrt{TN})$ dominates the right hand side of (3), then linear speedup can be achieved by increasing the number of nodes N. An additional point worth discussing is the importance of the choice for matrix W since the convergence rate depends on λ which depends on W. A common method to define W for undirected graph is the Maximum-Degree Method (MDM), see Boyd et al. (2004). Denote d_i as the degree of vertex i and $d_{\max} = \max_i d_i$, MDM sets $W_{i,i} = 1 - d_i/d_{\max}$, $W_{i,j} = 1/d_{\max}$ if $i \neq j$ and (i, j) is an edge, and $W_{i,j} = 0$ otherwise. This W ensures Assumption A4 for many common connected graph types, so does the variant $\gamma I + (1-\gamma)W$ for any $\gamma \in [0, 1)$. A refined choice of W coupled with a comprehensive discussion on λ of Theorem 2 can be found in Boyd et al. (2009), e.g., $1 - \lambda = O(1/N^2)$ for cycle graphs, $1 - \lambda = O(1/\log(N))$ for hypercube graphs, $\lambda = 0$ for fully connected graph. Intuitively, λ can be close to 1 for sparse graphs and to 0 for dense graphs. This is consistent with (2), which RHS is large for λ close to 1 and small for λ close to 0.

4.1. Application to AMSGrad algorithm

We now present, in Algorithm 3, a special case of our algorithmic framework, namely Decentralized AMSGrad, which is a decentralized variant of AMSGrad. Compared with DADAM, this algorithm exhibits a dynamic average consensus mechanism to keep track of the average of $\{\hat{v}_{t,i}\}_{i=1}^N$, stored as $\tilde{u}_{t,i}$ on the ith node, and uses $u_{t,i} := \max(\tilde{u}_{t,i}, \epsilon)$ for updating the adaptive learning rate for node i. As the number of iteration grows, even though $\hat{v}_{t,i}$ on different nodes can converge to different constants, the quantity $u_{t,i}$ will converge to the same number $\lim_{t\to\infty} \frac{1}{N}\sum_{i=1}^N \hat{v}_{t,i}$ if the limit exists.

Algorithm 3 Decentralized AMSGrad (N nodes)

1: **Input:** learning rate α, initial point $x_{1,i} = x_{init}, u_{\frac{1}{2},i} = \hat{v}_{0,i} = \epsilon\mathbf{1}$ (with $\epsilon \geq 0$), $m_{0,i} = 0$, mixing matrix W
2: **for** $t = 1, 2, \cdots, T$ **do**
3: **for all** $i \in [N]$ **do in parallel**
4: $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
5: $m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i}$
6: $v_{t,i} = \beta_2 v_{t-1,i} + (1-\beta_2) g_{t,i}^2$
7: $\hat{v}_{t,i} = \max(\hat{v}_{t-1,i}, v_{t,i})$
8: $x_{t+\frac{1}{2},i} = \sum_{j=1}^N W_{ij} x_{t,j}$
9: $\tilde{u}_{t,i} = \sum_{j=1}^N W_{ij} \tilde{u}_{t-\frac{1}{2},j}$
10: $u_{t,i} = \max(\tilde{u}_{t,i}, \epsilon)$
11: $x_{t+1,i} = x_{t+\frac{1}{2},i} - \alpha \frac{m_{t,i}}{\sqrt{u_{t,i}}}$
12: $\tilde{u}_{t+\frac{1}{2},i} = \tilde{u}_{t,i} - \hat{v}_{t-1,i} + \hat{v}_{t,i}$
13: **end for**

This average consensus mechanism enables the consensus of adaptive learning rates on different nodes, which accordingly guarantees the convergence of the method to stationary points. The consensus of adaptive learning rates is the key difference between decentralized AMSGrad and DADAM and is the reason why decentralized AMSGrad is convergent while DADAM is not. One may notice that decentralized AMSGrad does not reduce to AMSGrad for $N = 1$ since the quantity $u_{t,i}$ in line 10 is calculated based on $v_{t-1,i}$ instead of $v_{t,i}$. This design encourages the execution of gradient computation and communication in a parallel manner. Specifically, line 4-7 (line 4-6) in Algorithm 3 (Algorithm 2) can be executed in parallel with line 8-9 (line 7-8) to overlap communication and computation time. If $u_{t,i}$ depends on $v_{t,i}$ which in turn depends on $g_{t,i}$, the

gradient computation must finish before the consensus of the adaptive learning rate in line 9. This can slow down the running time per-iteration of the algorithm. To avoid such delayed adaptive learning, adding $\tilde{u}_{t-\frac{1}{2},i} = \tilde{u}_{t,i} - \hat{v}_{t-1,i} + \hat{v}_{t,i}$ before line 9 and getting rid of line 12 in Algorithm 2 is an option. Similar convergence guarantees will hold since one can easily modify our proof of Theorem 2 for such update rule. As stated above, Algorithm 3 converges, with the following rate:

Theorem 3 *Assume A1-A4. Set $\alpha = 1/\sqrt{Td}$. When $\alpha \leq \frac{\epsilon^{0.5}}{16L}$, then Algorithm 3 satisfies:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\frac{\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\right\|^2\right] \leq C_1'\frac{\sqrt{d}}{\sqrt{TN}}\left(D_f+\sigma^2\right) + C_2'\frac{N}{T} + C_3'\frac{N^{1.5}}{T^{1.5}d^{0.5}} + C_4'\frac{\sqrt{N}d}{T} + C_5'\frac{Nd^{0.5}}{T^{1.5}},$$

where $D_f := \mathbb{E}[f(Z_1)] - \min_x f(x)$, $C_1' = C_1$, $C_2' = C_2$, $C_3' = C_3$, $C_4' = C_4 G_\infty^2$ and $C_5' = C_5 G_\infty^2$. C_1, C_2, C_3, C_4, C_5 are independent of d, T and N defined in Theorem 2. In addition, the consensus of variables at different nodes is given by $\frac{1}{N}\sum_{i=1}^{N}\left\|x_{t,i} - \overline{X}_t\right\|^2 \leq \frac{N}{T}\left(\frac{1}{1-\lambda}\right)^2 G_\infty^2 \frac{1}{\epsilon}$.

Theorem 3 shows that Algorithm 3 converges with a rate of $\mathcal{O}(\sqrt{d}/\sqrt{T})$ when T is large, which is the best known convergence rate under the given assumptions. Note that in some related works, SGD admits a convergence rate of $\mathcal{O}(1/\sqrt{T})$ without any dependence on the dimension of the problem. Such improved convergence rate is derived under the assumption that the gradient estimator has a bounded L_2 norm, hence hiding a dependency of $\sqrt{d}$ in the final convergence rate. Another remark is that the convergence measure can be converted to $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla f(\overline{X}_t)\right\|^2\right]$ using the fact that $\|\overline{U}_t\|_\infty \leq G_\infty^2$ (by update rule of Algorithm 3), for easier comparison with prior works.

In the next section, we provide a decentralized version of AdaGrad (Duchi et al., 2011) (optionally with momentum) converted by Algorithm 2, further supporting the usefulness of our decentralization framework. The required modification for decentralized AdaGrad is to specify line 4 of Algorithm 2 as follows: $\hat{v}_{t,i} = \frac{t-1}{t}\hat{v}_{t-1,i} + \frac{1}{t}g_{t,i}^2$, which is equivalent to $\hat{v}_{t,i} = \frac{1}{t}\sum_{k=1}^{t} g_{k,i}^2$. We call this algorithm Decentralized AdaGrad. The pseudo code of the algorithm is shown in Algorithm 4. There are two details in Algorithm 4 worth mentioning. The first one is that the introduced framework leverages momentum $m_{t,i}$ in the updates, while original AdaGrad does not use momentum. The momentum can be turned off by setting $\beta_1 = 0$ and the convergence results will still hold. The other one is that in Decentralized AdaGrad, we use the average instead of the sum in the term $\hat{v}_{t,i}$. In other words, we write $\hat{v}_{t,i} = \frac{1}{t}\sum_{k=1}^{t} g_{k,i}^2$. This latter point is different from the original AdaGrad which actually uses $\hat{v}_{t,i} = \sum_{k=1}^{t} g_{k,i}^2$.

Algorithm 4 Decentralized AdaGrad (N nodes)

1: **Input:** learning rate α, initial point $x_{1,i} = x_{init}, u_{\frac{1}{2},i} = \hat{v}_{0,i} = \epsilon\mathbf{1}$ (with $\epsilon \geq 0$), $m_{0,i} = 0$, mixing matrix W
2: **for** $t = 1, 2, \cdots, T$ **do**
3: **for all** $i \in [N]$ **do in parallel**
4: $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
5: $m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i}$
6: $\hat{v}_{t,i} = \frac{t-1}{t}\hat{v}_{t-1,i} + \frac{1}{t}g_{t,i}^2$
7: $x_{t+\frac{1}{2},i} = \sum_{j=1}^{N} W_{ij} x_{t,j}$
8: $\tilde{u}_{t,i} = \sum_{j=1}^{N} W_{ij}\tilde{u}_{t-\frac{1}{2},j}$
9: $u_{t,i} = \max(\tilde{u}_{t,i}, \epsilon)$
10: $x_{t+1,i} = x_{t+\frac{1}{2},i} - \alpha\frac{m_{t,i}}{\sqrt{u_{t,i}}}$
11: $\tilde{u}_{t+\frac{1}{2},i} = \tilde{u}_{t,i} - \hat{v}_{t-1,i} + \hat{v}_{t,i}$
12: **end for**

4.2. Application to AdaGrad algorithm

In the original AdaGrad, a constant stepsize (α independent of t or T) is used with $\hat{v}_{t,i} = \sum_{k=1}^{t} g_{k,i}^2$. This is equivalent to using a well-known decreasing stepsize sequence $\alpha_t = \frac{1}{\sqrt{t}}$ with $\hat{v}_{t,i} = \frac{1}{t}\sum_{k=1}^{t} g_{k,i}^2$. In our convergence analysis, which can be found below, we use a constant stepsize $\alpha = O(\frac{1}{\sqrt{T}})$ to replace the decreasing stepsize sequence $\alpha_t = O(\frac{1}{\sqrt{t}})$. This is popularly used in Stochastic Gradient Descent analysis for the sake of simplicity and to achieve a better convergence rate. In addition, it is easy to modify our theoretical framework to include decreasing stepsize sequences such as $\alpha_t = O(\frac{1}{\sqrt{t}})$. The convergence analysis for decentralized AdaGrad is given in Theorem 4.

Theorem 4 *Under A1-A4, if $\alpha = \sqrt{N}/\sqrt{Td}$. When $\alpha \leq \frac{\epsilon^{0.5}}{16L}$, then Algorithm 4 satisfies:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\frac{\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\right\|^2\right] \leq \frac{C_1'\sqrt{d}}{\sqrt{TN}}D_f' + \frac{C_2'}{T} + \frac{C_3' N^{1.5}}{T^{1.5}d^{0.5}} + \frac{\sqrt{N}(1+\log(T))}{T}(dC_4' + \frac{\sqrt{d}}{T^{0.5}}C_5'),$$

where $D_f' := \mathbb{E}[f(Z_1)] - \min_z f(z)] + \sigma^2$, $C_1' = C_1$, $C_2' = C_2$, $C_3' = C_3$, $C_4' = C_4 G_\infty^2$ and $C_5' = C_5 G_\infty^2$. C_1, C_2, C_3, C_4, C_5 are defined in Theorem 2 independent of d, T and N. In addition, the consensus of variables at different nodes is given by $\frac{1}{N}\sum_{i=1}^{N}\left\|x_{t,i} - \overline{X}_t\right\|^2 \leq \frac{N}{T}\left(\frac{1}{1-\lambda}\right)^2 G_\infty^2 \frac{1}{\epsilon}$.

5. Numerical Experiments

In this section, we conduct some experiments to test the performance of Decentralized AMSGrad, developed in Algorithm 3, on both *homogeneous* data and *heterogeneous* (i.e. the data generating distribution on different nodes are assumed to be different) data distribution. Comparison with DADAM and the decentralized parallel stochastic gradient descent (D-PSGD) developed in Lian et al. (2017) are conducted. We train a Convolutional Neural Network (CNN) with 3 convolution layers followed by a fully connected layer on MNIST (LeCun, 1998). We set $\epsilon = 10^{-6}$ for both Decentralized AMSGrad and DADAM. The learning rate is chosen from the grid $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]$ based on validation accuracy for all algorithms. In the following experiments, the graph contains 5 nodes and each node can only communicate with its two adjacent neighbors forming a cycle. Regarding the mixing matrix W, we set $W_{ij} = 1/3$ if nodes i and j are neighbors and $W_{ij} = 0$ otherwise. More details on experiments can be found in the supplementary material of our paper.

5.1. Effect of heterogeneity

Homogeneous data: The whole dataset is shuffled and evenly split into different nodes. Such a setting is possible when the nodes are in a computer cluster. We note, Figure 1, that decentralized AMSGrad and DADAM perform quite similarly while D-PSGD (labelled as DGD) is much slower both in terms of training loss and test accuracy. Though the (possible) non convergence of DADAM, mentioned prior, its performance are empirically good on homogeneous data. The reason is that the adaptive learning rates tend to be similar on different nodes under homogeneous data distribution.

Heterogeneous data: Here, each node only contains training data with two labels out of ten. Such a setting is common when data shuffling is prohibited, such as in federated learning. We can

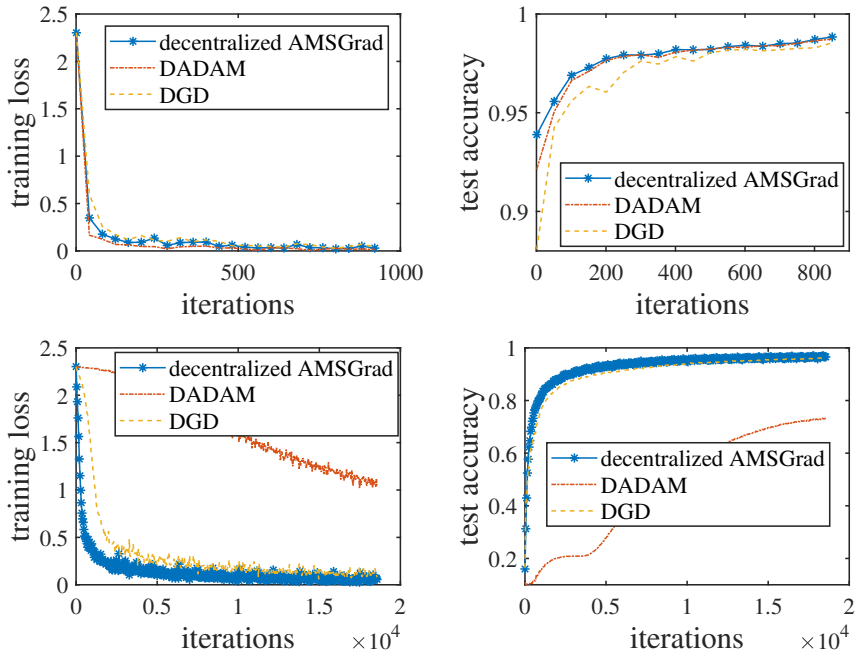

Figure 1: Training loss and Testing accuracy for homogeneous (upper panels) and heterogeneous data (bottom panels)

see that each algorithm converges significantly slower than with homogeneous data. Especially, the performance of DADAM deteriorates significantly. Decentralized AMSGrad achieves the best training and testing performance in that setting as observed in Figure 1.

5.2. Sensitivity to the Learning Rate

We compare the training loss and testing accuracies of different D-PSGD, DADAM, and our proposed Decentralized AMSGrad, with different stepsizes on *heterogeneous* data distribution. We use 5 nodes and the heterogeneous data distribution is created by assigning each node with data of only two labels. Note that there are no overlapping labels between different nodes. We observe Figure 2 that the stepsize 10^{-3} works best for D-PSGD in terms of test accuracy and 10^{-1} works best in terms of training loss. This difference is caused by the inconsistency among the model parameters on different nodes when the stepsize is large. Figure 2 shows the performance of decentralized AMSGrad with different stepsizes. We see that its best performance is better than the one of D-PSGD and displays a more stable test accuracy, i.e. less sensitive to the stepsize tuning. As expected, the performance of DADAM is not as good as D-PSGD or Decentralized AMSGrad, see Figure 2. Its divergence characteristic, highlighted Section 3.1, coupled with the heterogeneity in the data amplify its non-convergence issue. We note the advantages of Decentralized AMSGrad both in terms of performance and ease of tuning.

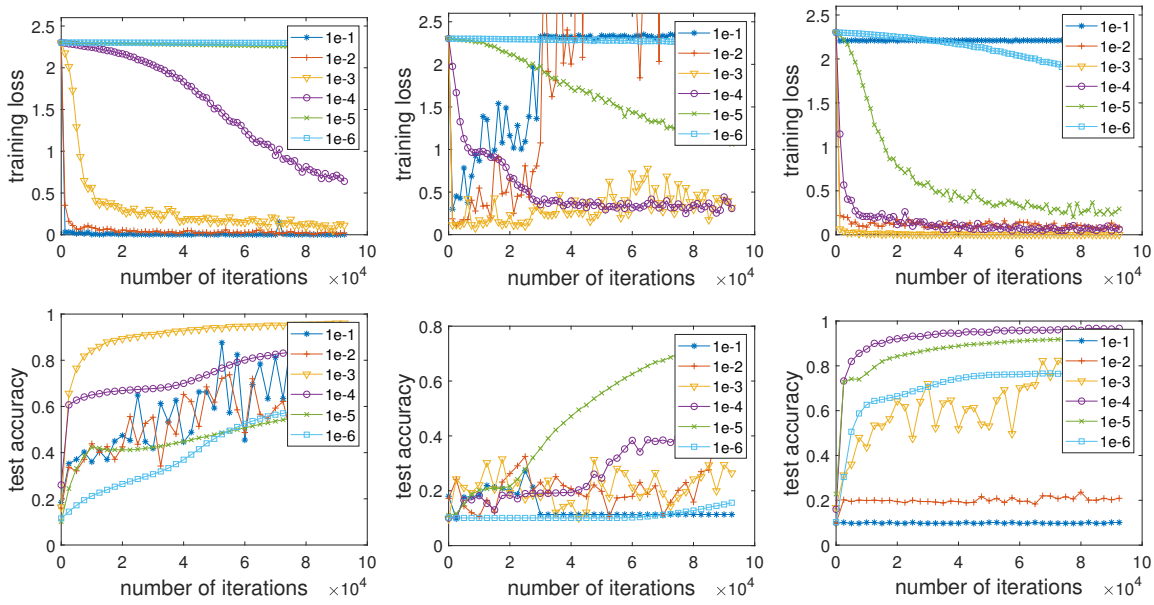

Figure 2: Training Loss (upper panels) and Testing Accuracy (bottom panels) comparison of different stepsizes for various methods. Order of plots (left to right): DP-SGD, DADAM and DAMS.

6. Conclusion

This paper studies the problem of designing adaptive gradient methods for decentralized training. We propose a unifying algorithmic framework that can convert existing adaptive gradient methods to decentralized settings. We rigorously show that if the original algorithm converges under some minor conditions, the converted algorithm obtained using our proposed framework is guaranteed to converge to stationary points of the objective loss function. By applying our framework to AMSGrad, we propose the first convergent adaptive gradient methods, namely Decentralized AMSGrad. We also give an extension to a decentralized variant of AdaGrad for completeness of our converting scheme.

References

Naman Agarwal, Brian Bullins, Xinyi Chen, Elad Hazan, Karan Singh, Cyril Zhang, and Yi Zhang. Efficient full-matrix adaptive regularization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 102–110, Long Beach, CA, 2019.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 440–445, Copenhagen, Denmark, 2017.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1709–1720, Long Beach, CA, 2017.

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Michael G. Rabbat. Stochastic gradient push for distributed deep learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 344–353, Long Beach, CA, 2019.

Stephen P. Boyd, Persi Diaconis, and Lin Xiao. Fastest mixing markov chain on a graph. *SIAM Rev.*, 46(4):667–689, 2004.

Stephen P. Boyd, Persi Diaconis, Pablo A. Parrilo, and Lin Xiao. Fastest mixing markov chain on graphs with symmetries. *SIAM J. Optim.*, 20(2):792–819, 2009.

Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011.

Tianyi Chen, Ziye Guo, Yuejiao Sun, and Wotao Yin. Cada: Communication-adaptive distributed adam. In *International Conference on Artificial Intelligence and Statistics*, pages 613–621. PMLR, 2021.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of A class of adam-type algorithms for non-convex optimization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, 2019.

Xiangyi Chen, Xiaoyun Li, and Ping Li. Toward communication efficient adaptive gradient method. In *Proceedings of the ACM-IMS Foundations of Data Science Conference (FODS)*, pages 119–128, Virtual Event, USA, 2020.

Trishul M. Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 571–582, Broomfield, CO, 2014.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

John C. Duchi, Alekh Agarwal, and Martin J. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Trans. Autom. Control.*, 57(3):592–606, 2012.

Mingyi Hong, Davood Hajinezhad, and Ming-Min Zhao. Prox-pda: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1529–1538, Sydney, Australia, 2017.

Belhal Karimi, Xiaoyun Li, and Ping Li. Fed-lamb: Layerwise and dimensionwise locally adaptive optimization algorithm. *arXiv preprint arXiv:2110.00532*, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015.

Anastasia Koloskova, Sebastian U. Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 3478–3487, Long Beach, CA, 2019.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 983–992, Naha, Japan, 2019.

Xiaoyun Li, Belhal Karimi, and Ping Li. On distributed adaptive optimization with gradient compression. In *Proceedings of the Tenth International Conference on Learning Representations (ICLR)*, Virtual Event, 2022.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5330–5340, 2017.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

Paolo Di Lorenzo and Gesualdo Scutari. NEXT: in-network nonconvex optimization. *IEEE Trans. Signal Inf. Process. over Networks*, 2(2):120–136, 2016.

Songtao Lu, Xinwei Zhang, Haoran Sun, and Mingyi Hong. GNSD: a gradient-tracking based nonconvex stochastic algorithm for decentralized optimization. In *Proceedings of the IEEE Data Science Workshop (DSW)*, pages 315–321, Minneapolis, MN, 2019.

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, 2019.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282, Fort Lauderdale, FL, 2017.

Parvin Nazari, Davoud Ataee Tarzanagh, and George Michailidis. Dadam: A consensus-based distributed adaptive gradient method for online optimization. *arXiv preprint arXiv:1901.09109*, 2019.

Angelia Nedic and Asuman E. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Autom. Control.*, 54(1):48–61, 2009.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, Austria, 2021.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. EXTRA: an exact first-order algorithm for decentralized consensus optimization. *SIAM J. Optim.*, 25(2):944–966, 2015.

Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4452–4463, Montréal, Canada, 2018.

Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D^2: Decentralized training over decentralized data. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 4855–4863, Stockholmsmässan, Sweden, 2018.

Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6155–6165, Long Beach, CA, 2019.

Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary B. Charles, Dimitris S. Papailiopoulos, and Stephen J. Wright. ATOMO: communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9872–9883, Montréal, Canada, 2018.

Jun-Kun Wang, Xiaoyun Li, Belhal Karimi, and Ping Li. An optimistic acceleration of amsgrad for nonconvex optimization. In *Proceedings of the Asian Conference on Machine Learning (ACML)*, pages 422–437, Virtual Event, 2021.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1306–1316, Montréal, Canada, 2018.

Rachel Ward, Xiaoxia Wu, and Léon Bottou. Adagrad stepsizes: sharp convergence over nonconvex landscapes. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6677–6686, Long Beach, CA, 2019.

Zhiqiang Xu, Dong Li, Weijie Zhao, Xing Shen, Tianbo Huang, Xiaoyun Li, and Ping Li. Agile and accurate CTR prediction model training formassive-scale online advertising systems. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD)*, Online conference [Xi'an, China], 2021.

Yan Yan, Tianbao Yang, Zhe Li, Qihang Lin, and Yi Yang. A unified analysis of stochastic momentum methods for deep learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2955–2961, Stockholm, Sweden, 2018.

Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM J. Optim.*, 26(3):1835–1854, 2016.

Manzil Zaheer, Sashank J. Reddi, Devendra Singh Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9815–9825, Montréal, Canada, 2018.

Weijie Zhao, Jingyuan Zhang, Deping Xie, Yulei Qian, Ronglai Jia, and Ping Li. AIBox: CTR prediction model training on a single node. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 319–328, Beijing, China, 2019.

Weijie Zhao, Xuewu Jiao, Mingqing Hu, Xiaoyun Li, Xiangyu Zhang, and Ping Li. Communication-efficient terabyte-scale model training framework for online advertising. *arXiv preprint arXiv:2201.05500*, 2022.

Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.

Yingxue Zhou, Belhal Karimi, Jinxing Yu, Zhiqiang Xu, and Ping Li. Towards better generalization of adaptive gradient methods. In *Advances in Neural Information Processing Systems (NeurIPS)*, virtual, 2020.

Fangyu Zou and Li Shen. On the convergence of weighted adagrad with momentum for training deep neural networks. *arXiv preprint arXiv:1808.03408*, 2018.