On the Convergence of Decentralized Adaptive Gradient Methods

Anonymous Authors[1]

Abstract

Adaptive gradient methods including Adam, AdaGrad, and their variants have been very successful for training deep learning models, such as neural networks, in the past few years. Meanwhile, given the need for distributed training procedures, distributed optimization algorithms are at the center of attention. 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 that regard, more and more attention is shifted from the traditional parameter server training paradigm to the decentralized one, which usually requires lower communication costs. In this paper, we rigorously incorporate adaptive gradient methods into decentralized training procedures and introduce novel convergent decentralized adaptive gradient methods. 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.

1. Introduction

Distributed training of machine learning models is drawing growing attention in the past few years due to its practical benefits and necessities. Given the evolution of computing capabilities of CPUs and GPUs, computation time in distributed settings is gradually dominated by the communication time in many circumstances (Chilimbi et al., 2014; McMahan et al., 2017). As a result, a large amount of recent works has been focussing on reducing 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 communication in the whole network, many effective communication reductions have been proposed based on gradient compression (Aji & Heafield, 2017) and quantization (Chen et al., 2010; Ge et al., 2013; Jegou et al., 2010) techniques. Despite these communication reduction techniques, its cost still, usually, 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., 2011b), where the parameter server is removed and each node only communicates with its neighbors, is drawing attention. It has been shown in Lian et al. (2017) that decentralized training algorithms can outperform parameter server-based algorithms when the training bottleneck is the communication cost. The decentralized paradigm is also preferred when a central parameter server is 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., 2011a), Adam (Kingma & Ba, 2015) or AMSGrad (Reddi et al., 2018). 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 & Monro, 1951). Despite a large amount of studies within the distributed optimization literature, few works have considered bringing adaptive gradient methods into distributed training, largely due to the lack of understanding of their convergence behaviors. Notably, Reddi et al. (2020) develop the first decentralized ADAM method for distributed optimization problems with a direct application to federated learning. An inner loop is employed to compute mini-batch gradients on each node and a global adaptive step is applied to update the global parameter at each outer iteration. Yet, in the settings of our paper, nodes can only communicate *to their neighbors* on a fixed communication graph while a server/worker communication is required in Reddi et al. (2020). Designing adaptive methods in such settings 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. This paper is an attempt at bridging the gap between both realms in nonconvex optimization. Our contributions are summarized as follows:

- In this paper, we investigate the possibility of using 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 an adaptive gradient method from a centralized method to its decentralized variant.

- 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 Theroem 2, for analyzing the behavior of decentralized adaptive gradient methods obtained as a result of our technique. Thus, we characterize the convergence rate of decentralized AMSGrad, which is the first convergent decentralized adaptive gradient method, to the best of our knowledge.

A *novel technique* in our framework is a mechanism to enforce a *consensus on adaptive learning rates* at different nodes. We show the importance of consensus on adaptive learning rates by proving a divergent problem instance for a recently proposed decentralized adaptive gradient method, namely DADAM (Nazari et al., 2019), a decentralized version of AMSGrad. Though consensus is performed on the model parameter, DADAM lacks consensus principles on adaptive learning rates.

After having presented existing related work and important concepts of decentralized adaptive methods in Section 2, we develop our general framework for converting any adaptive gradient algorithm in its decentralized counterpart along with their rigorous finite-time convergence analysis in Section 3 concluded by some illustrative examples of our framework's behavior in practice. After having used AMSGrad as a proptype method for our decentralized framework, we give an interesting extension of the latter to AdaGrad in Section 4.

Notations: $x_{t,i}$ denotes variable x at node i and iteration t. $\|\cdot\|_{abs}$ denotes the entry-wise L_1 norm of a matrix, 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. Decentralized Adaptive Training and Divergence of DADAM

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., 2011b), Distributed Subgradient Descent (Nedic & Ozdaglar, 2009). More recent algorithms include Extra (Shi et al., 2015), Next (Di Lorenzo & 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) propose 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 only at its initial stage. No existing work, to our knowledge, has seriously considered integrating *adaptive gradient methods* in the setting of decentralized learning. One noteworthy work (Nazari et al., 2019) propose 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., 2011a) or Adam (Kingma & 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). The method of reference, called Adam, has been analyzed in Reddi et al. (2018) where the authors point out an error in previous convergence analyses. Since then, a variety of papers have been focussing on analyzing the convergence behavior of the numerous existing adaptive gradient methods. Ward et al. (2019), Li & 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. Zou & Shen (2018) provide a unified convergence analysis for AdaGrad with momentum. Noticeable recent works on adaptive gradient methods can be found in Agarwal et al. (2019); Luo et al. (2019); Zaheer et al. (2018).

2.2. Decentralized Optimization

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

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

where x is 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 $i \in [N]$. Throughout the paper, 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 is 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 $g_{t,i}$. 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 each coordinate have bounded variance, i.e. $\mathbb{E}[g_{t,i}] = \nabla f_i(x_{t,i})$ and $\mathbb{E}[([g_{t,i} - f_i(x_{t,i})]_j)^2] \leq \sigma^2, \forall t, i, j$.*

Assumptions A1 and A3 are standard in distributed optimization literature. A2 is slightly stronger than the traditional assumption that the estimator has bounded variance, 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. In decentralized optimization, the nodes are connected as a graph and each node only communicates to its neighbors. In such case, one usually constructs a $N \times N$ matrix W for information sharing when designing new algorithms. We denote λ_i to be its ith largest eigenvalue and define $\lambda \triangleq \max(|\lambda_2|, |\lambda_N|)$. The matrix W cannot be arbitrary, its required key properties are listed in the following assumption:

A 4. *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.*

We now present the failure to converge of current decentralized adaptive method before introducing our general framework for decentralized adaptive gradient methods.

2.3. Divergence of DADAM

Recently, Nazari et al. (2019) initiated an attempt to bring adaptive gradient methods into decentralized optimization with Decentralized ADAM (DADAM), shown in Algorithm 1. DADAM is essentially a decentralized version of ADAM and the key modification is the use of 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 such mixing matrix is standard for decentralizing an algorithm, such as distributed gradient descent (Nedic & 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 training neural networks is still unknown. The next theorem shows that DADAM may fail to converge in the offline settings.

Algorithm 1 DADAM (with N nodes)

```
1: Input: $\alpha$, current point $X_t$, $u_{\frac{1}{2},i} = \hat{v}_{0,i} = \epsilon\mathbf{1}$, $m_0 = 0$
   and 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:     $\hat{v}_{t,i} = \beta_3 \hat{v}_{t,i} + (1 - \beta_3) \max(\hat{v}_{t-1,i}, v_{t,i})$
8:     $x_{t+\frac{1}{2},i} = \sum_{j=1}^N W_{ij} x_{t,j}$
9:     $x_{t+1,i} = x_{t+\frac{1}{2},i} - \alpha \frac{m_{t,i}}{\sqrt{\hat{v}_{t,i}}}$
10: end for
```

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

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 $\hat{v}_{t,1} = 16, \hat{v}_{t,2} = 4, \forall t \geq 1$. Thus, after $t = 1$, DADAM is equivalent to running decentralized gradient descent (DGD) (Yuan et al., 2016) with a re-scaled f_1 and f_2, *i.e.* running DGD 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 Th. 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 $\nabla f(0.5) = 0.5$. $\square$

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 nonconvex offline setting (common 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 decentralized adaptive gradient methods that are guaranteed to converge to stationary points under assumptions and provide a characterization of that convergence in finite-time and independently of the initialization.

3. Convergence of Decentralized Adaptive Gradient Methods

In this section, we discuss the difficulties of designing adaptive gradient methods in decentralized optimization and introduce an algorithmic framework that can turn existing convergent adaptive gradient methods to their decentralized counterparts. We also develop the first convergent decentralized adaptive gradient method, converted from AMSGrad, *as an instance of this framework*.

3.1. 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. (2020). However, in our decentralized setting, every node can only communicate with its neighbors and such central server does not exist. Under that setting, the information for updating the adaptive learning rates can only be shared locally instead of broadcasted over the whole network. This makes it impossible to obtain, in a single iteration, a synchronized adaptive learning rate update using all the information in 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 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.

In the following subsection, we introduced the main archetype of the consensus of adaptive learning rates mechanism within a decentralized framework

3.2. Decentralized Adaptive Gradient Unifying 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 adaptive 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 for AdaBound in Luo et al. (2019). In this paper, we present an algorithm framework for decentralized adaptive gradient methods as Algorithm 2, which uses average consensus of $\hat{v}_{t,i}$ (see consensus update in line 8 and 11) to help convergence. Algorithm 2 can become different adaptive gradient methods by specifying r_t as different functions. E.g., when we choose $\hat{v}_{t,i} = \frac{1}{t}\sum_{k=1}^{t} g_{k,i}^2$, Algorithm 2 becomes a decentralized version of AdaGrad. When one chooses $\hat{v}_{t,i}$ to be the adaptive learning rate for AMSGrad, we get 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 second moment on the whole network. Also, this design will 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).

Algorithm 2 Decentralized Adaptive Gradient Method (with N nodes)

```
1: Input: $\alpha$, 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
```

The following result gives a finite-time convergence rate for our framework described in Algorithm 2. The convergence bound reads as follows:

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

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

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, $\frac{1}{N}\sum_{i=1}^{N}\left\|x_{t,i}-\overline{X}_t\right\|^2\le\alpha^2\left(\frac{1}{1-\lambda}\right)^2 dG_\infty^2\frac{1}{\epsilon}$ which quantifies the consensus error.

In addition, one can specify α to show convergence in terms of T, d, and N. An immediate result is by setting $\alpha=\sqrt{N}/\sqrt{Td}$, which is shown in Corollary 2.1.

Corollary 2.1. *Assume A1-A4. Set $\alpha=\sqrt{N}/\sqrt{Td}$. When $\alpha\le\frac{\epsilon^{0.5}}{16L}$, Algorithm 2 yields the following regret bound*

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

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 2.1 indicates that if $\mathbb{E}[\mathcal{V}_T]=o(T)$ and $\bar{U}_t$ is upper bounded, then Algorithm 2 is guaranteed to converge to stationary points of the loss function. Intuitively, this means that if the adaptive learning rates on different nodes do not change too fast, the algorithm can converge. In convergence analysis, the term $\mathbb{E}[\mathcal{V}_T]$ upper bounds the total bias in update 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 do not converge. Later, we will show convergence of decentralized versions of AMSGrad and AdaGrad by bounding this term as $O(Nd)$ and $O(Nd\log(T))$, respectively.

Corollary 2.1 also conveys the benefits of using more nodes in the graph employed. 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.

We now present, in Algorithm 3, a notable special case of our algorithmic framework, namely Decentralized AMSGrad, which is a decentralized variant of AMSGrad. Compared with DADAM, the above 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 ith node, and uses $u_{t,i}:=\max(\tilde{u}_{t,i},\epsilon)$ for updating the adaptive learning rate for ith node. As the number of iteration grows, even though $\hat{v}_{t,i}$ on different nodes can converge to different constants, the $u_{t,i}$ 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. 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.

Algorithm 3 Decentralized AMSGrad (with N nodes)

```
1: Input: learning rate $\alpha$, initial point $x_{1,i}=x_{init}, u_{\frac{1}{2},i}=\hat{v}_{0,i}=\epsilon\mathbf{1}$ (with $\epsilon\ge 0$), $m_{0,i}=0$, 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
```

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 step 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 get 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 $\alpha \leq \frac{\epsilon^{0.5}}{16L}$, Algorithm 3 yields the following regret bound*

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

where $D_f := \mathbb{E}[f(Z_1)] - \min_x f(x)$, $C_1' = C_1$, $C_2' = C_2$, $C_3' = C_3$, $C_4' = C_4G_\infty^2$ and $C_5' = C_5G_\infty^2$. C_1, C_2, C_3, C_4, C_5 are constants 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 have a bounded L_2 norm, which can thus hide a dependency of $\sqrt{d}$ in the final convergence rate. Another remark is 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 the ease of comparison with existing literature.

3.3. Convergence Analysis

The detailed proofs of this section are reported in the supplementary material.

Proof of Theorem 2: We now present a proof sketch for out main convergence result of Algorithm 2. *Step 1: Reparameterization.* Similarly to Yan et al. (2018); Chen et al. (2019) with SGD (with momentum) and centralized adaptive gradient methods, define the following auxiliary sequence:

$$Z_t = \overline{X}_t + \frac{\beta_1}{1-\beta_1}(\overline{X}_t - \overline{X}_{t-1})\,, \quad (5)$$

with $\overline{X}_0 \triangleq \overline{X}_1$. Such an auxiliary sequence can help us deal with the bias brought by the momentum and simplifies the convergence analysis. An intermediary result needed to conduct our proof reads:

Lemma 1. *For the sequence defined in (5), we have*

$$Z_{t+1} - Z_t = \alpha\frac{\beta_1}{1-\beta_1}\frac{1}{N}\sum_{i=1}^{N} m_{t-1,i}\odot\left(\frac{1}{\sqrt{u_{t-1,i}}}-\frac{1}{\sqrt{u_{t,i}}}\right) - \alpha\frac{1}{N}\sum_{i=1}^{N}\frac{g_{t,i}}{\sqrt{u_{t,i}}}\,. \quad (6)$$

Lemma 1 does not display any momentum term in $\frac{1}{N}\sum_{i=1}^{N}\frac{g_{t,i}}{\sqrt{u_{t,i}}}$. This simplification is convenient since it is directly related to the current gradients instead of the exponential average of past gradients.

Step 2: Smoothness. Using smoothness assumption A1 involves the following scalar product term: $\kappa_t := \langle\nabla f(Z_t), \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(x_{t,i})/\sqrt{\overline{U}_t}\rangle$ which can be lower bounded by:

$$\kappa_t \geq \frac{1}{2}\left\|\frac{\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\right\|^2 - \frac{3}{2}\left\|\frac{\nabla f(Z_t)-\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\right\|^2 - \frac{3}{2}\left\|\frac{\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(x_{t,i})-\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\right\|^2\,. \quad (7)$$

The above inequality substituted in the smoothness condition $f(Z_{t+1}) \leq f(Z_t) + \langle\nabla f(Z_t), Z_{t+1}-Z_t\rangle + \frac{L}{2}\|Z_{t+1}-Z_t\|^2$ yields:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\frac{\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\right\|^2\right] \leq \frac{2}{T\alpha}\mathbb{E}[\Delta_f] + \frac{2}{T}\frac{\beta_1 D_1}{1-\beta_1} + \frac{2D_2}{T} + \frac{3D_3}{T} + \frac{L}{T\alpha}\sum_{t=1}^{T}\mathbb{E}\left[\|Z_{t+1}-Z_t\|^2\right]\,, \quad (8)$$

where $\Delta_f := \mathbb{E}[f(Z_1)] - \mathbb{E}[f(Z_{T+1})]$ D_1, D_2 and D_3 are three terms, defined in the supplementary material, and which can be tightly bounded from above. We first bound D_3 using the following quantities of interest:

$$\sum_{t=1}^{T}\left\|Z_t-\overline{X}_t\right\|^2 \leq T\left(\frac{\beta_1}{1-\beta_1}\right)^2\alpha^2 d\frac{G_\infty^2}{\epsilon} \text{ and }$$
$$\sum_{t=1}^{T}\frac{1}{N}\sum_{i=1}^{N}\left\|x_{t,i}-\overline{X}_t\right\|^2 \leq T\alpha^2\left(\frac{1}{1-\lambda}\right)^2 dG_\infty^2\frac{1}{\epsilon}\,. \quad (9)$$

where $\lambda = \max(|\lambda_2|, |\lambda_N|)$ and recall that λ_i is ith largest eigenvalue of W.

Then, concerning the term D_2, few derivations, not detailed here for simplicity, yields:

$$D_2 \leq \frac{G_\infty^2}{N}\mathbb{E}\left[\sum_{t=1}^{T}\frac{1}{2\epsilon^{1.5}}\|-\sum_{l=2}^{N}\tilde{U}_t q_l q_l^T\|_{abs}\right]\,,$$

where q_l is the eigenvector corresponding to lth largest eigenvalue of W and $\| \cdot \|_{abs}$ is the entry-wise L_1 norm of matrices. We can also show that

$$\sum_{t=1}^{T} \| - \sum_{l=2}^{N} \tilde{U}_t q_l q_l^T \|_{abs} \leq \sqrt{N} \sum_{o=0}^{T-1} \frac{\lambda}{1-\lambda} \|(-\hat{V}_{o-1} + \hat{V}_o)\|_{abs} \,,$$

resulting in an upper bound for D_2 proportional to $\sum_{o=0}^{T-1} \|(-\hat{V}_{o-1} + \hat{V}_o)\|_{abs}$. Similarly:

$$D_1 \leq G_\infty^2 \frac{1}{2\epsilon^{1.5}} \frac{1}{\sqrt{N}} \mathbb{E}\left[\frac{1}{1-\lambda} \sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs} \right] .$$

Step 3: Bounding the drift term variance. An important term that needs upper bounding in our proof is the variance of the gradients multiplied (element-wise) by the adaptive learning rate:

$$\mathbb{E}\left[\left\| \frac{1}{N} \sum_{i=1}^{N} \frac{g_{t,i}}{\sqrt{u_{t,i}}} \right\|^2 \right] \leq \mathbb{E}[\|\Gamma_u^f\|^2] + \frac{d}{N} \frac{\sigma^2}{\epsilon} \,,$$

where $\Gamma_u^f := 1/N \sum_{i=1}^{N} \nabla f_i(x_{t,i}) / \sqrt{u_{t,i}}$. Two consecutive and simple bounding of the above yields:

$$\begin{aligned} \sum_{t=1}^{T} \mathbb{E}[\|\Gamma_u^f\|^2] &\leq 2 \sum_{t=1}^{T} \mathbb{E}[\|\Gamma_{\overline{U}}^f\|^2] \\ &+ 2 \sum_{t=1}^{T} \mathbb{E}\left[\frac{1}{N} \sum_{i=1}^{N} G_\infty^2 \frac{1}{\sqrt{\epsilon}} \left\| \frac{1}{\sqrt{u_{t,i}}} - \frac{1}{\sqrt{\overline{U}_t}} \right\|_1 \right] \end{aligned} \qquad (10)$$

and

$$\begin{aligned} \sum_{t=1}^{T} \mathbb{E}[\|\Gamma_{\overline{U}}^f\|^2] &\leq 2 \sum_{t=1}^{T} \mathbb{E}\left[\left\| \frac{\nabla f(\overline{X}_t)}{\sqrt{\overline{U}_t}} \right\|^2 \right] \\ &+ 2 \sum_{t=1}^{T} \mathbb{E}\left[\left\| \frac{1}{N} \sum_{i=1}^{N} \frac{\nabla f_i(\overline{X}_t) - \nabla f_i(x_{t,i})}{\sqrt{\overline{U}_t}} \right\|^2 \right] . \end{aligned} \qquad (11)$$

Then, by plugging the LHS of (11) in (8), and further bounding as operated for D_2, D_3 (see supplement), we obtain the desired bound in Theorem 2.

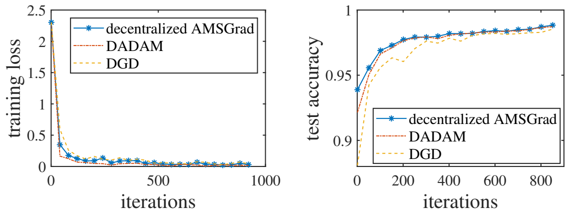
Proof of Theorem 3: Recall the bound in (3) of Theorem 2. Since Algorithm 3 is a special case of Algorithm 2, the remaining of the proof consists in characterizing the growth rate of $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}]$. By construction, $\hat{V}_t$ is non decreasing, then it can be shown that $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}] = \mathbb{E}[\sum_{i=1}^{N} \sum_{j=1}^{d} (-[\hat{v}_{0,i}]_j + [\hat{v}_{T-1,i}]_j)]$. Besides, since for all t, i, $\|g_{t,i}\|_\infty \leq G_\infty$ and $v_{t,i}$ is an exponential moving average of $g_{k,i}^2, k = 1, 2, \cdots, t$, we have $|[v_{t,i}]_j| \leq G_\infty^2$ for all t, i, j. By construction of $\hat{V}_t$, we also observe that each element of $\hat{V}_t$ cannot be greater than G_∞^2, i.e. $|[\hat{v}_{t,i}]_j| \leq G_\infty^2$ for all t, i, j. Given that $[\hat{v}_{0,i}]_j \geq 0$, we have

$$\begin{aligned} \mathbb{E}\left[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs} \right] &\leq \sum_{i=1}^{N} \sum_{j=1}^{d} \mathbb{E}[G_\infty^2] \\ &= NdG_\infty^2 \,. \end{aligned} \qquad (12)$$

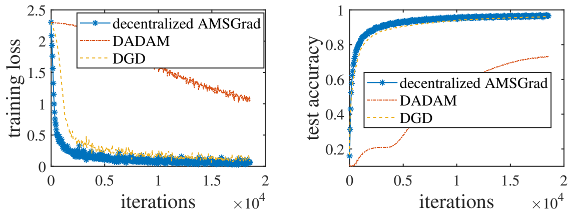
Substituting into (3) yields the desired convergence bound for Algorithm 3.

3.4. Illustrative 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* data distribution (i.e. the data generating distribution on different nodes are assumed to be different). Comparison with DADAM and the decentralized stochastic gradient descent (DGD) 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 and experiments can be found in the supplementary material of our paper.



(a) Homogeneous data



(b) Heterogeneous data

Figure 1. Training loss and Testing accuracy for homogeneous and heterogeneous data

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 see, Figure 1(a), that decentralized AMSGrad and DADAM perform quite similarly while DGD is much slower both in terms of training loss and test accuracy. Though the (possible) non convergence of DADAM, mentioned in this paper, its performance are empirically good on homogeneous data. The reason is that the adaptive learning rates tend to be similar on different nodes in presence of homogeneous data distribution. We thus compare these algorithms under the heterogeneous regime.

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 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 Figure 1(b).

4. Extension to AdaGrad

In this section, we provide a decentralized version of AdaGrad (Duchi et al., 2011a) (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 $\hat{v}_{t,i} = \frac{1}{t}\sum_{k=1}^{t} g_{k,i}^2$. Throughout this section, we will call this algorithm decentralized AdaGrad.

Algorithm 4 Decentralized AdaGrad (with N nodes)

```
1:  Input: learning rate $\alpha$, 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
```

The pseudo code of the algorithm is shown in Algorithm 4. There are two details in Algorithm 4 worth mentioning. The ifrst one is that the introduced framework leverages momentum $m_{t,i}$ in 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 use $\hat{v}_{t,i} = \sum_{k=1}^{t} g_{k,i}^2$. The reason is that 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 $\alpha_t = O(\frac{1}{\sqrt{t}})$. Such a replacement 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 shown in Theorem 4.

Theorem 4. *Assume A1-A4. Set $\alpha = \sqrt{N}/\sqrt{Td}$. When $\alpha \leq \frac{\epsilon^{0.5}}{16L}$, decentralized AdaGrad yields the following regret bound*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\left\|\frac{\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\right\|^2\right] \leq \frac{C_1'\sqrt{d}}{\sqrt{TN}}D_f' + \frac{C_2'}{T} + \frac{C_3' N^{1.5}}{T^{1.5} d^{0.5}} + \frac{\sqrt{N}(1+\log(T))}{T}(dC_4' + \frac{\sqrt{d}}{T^{0.5}}C_5')\,, \tag{13}$$

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. 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. With rigorous convergence analysis, we show that if the original algorithm satisfies some conditions, the converted algorithm obtained using our proposed framework is guaranteed to converge to stationary points of the loss function. By applying our framework to AMSGrad, we propose the first convergent decentralized adaptive gradient method, namely Decentralized AMSGrad. We also provide a decentralized variant of AdaGrad to support generality of our framework. Experiments show that the proposed algorithm achieves better performance than the baselines.

References

Agarwal, N., Bullins, B., Chen, X., Hazan, E., Singh, K., Zhang, C., and Zhang, Y. Efficient full-matrix adaptive regularization. In *International Conference on Machine Learning*, pp. 102–110, 2019.

Aji, A. F. and Heafield, K. Sparse communication for distributed gradient descent. In *Empirical Methods in Natural Language Processing*, pp. 440–445, 2017.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

Assran, M., Loizou, N., Ballas, N., and Rabbat, M. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pp. 344–353, 2019.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Chen, X., Liu, S., Sun, R., and Hong, M. On the convergence of a class of adam-type algorithms for non-convex optimization. In *International Conference for Learning Representations*, 2019.

Chen, Y., Guan, T., and Wang, C. Approximate nearest neighbor search by residual vector quantization. *Sensors*, 10(12):11259–11273, 2010.

Chilimbi, T., Suzue, Y., Apacible, J., and Kalyanaraman, K. Project adam: Building an efficient and scalable deep learning training system. In *Symposium on Operating Systems Design and Implementation*, pp. 571–582, 2014.

Di Lorenzo, P. and Scutari, G. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011a.

Duchi, J. C., Agarwal, A., and Wainwright, M. J. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2011b.

Ge, T., He, K., Ke, Q., and Sun, J. Optimized product quantization for approximate nearest neighbor search. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2946–2953, 2013.

Hong, M., Hajinezhad, D., and Zhao, M.-M. Prox-pda: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks. In *International Conference on Machine Learning*, pp. 1529–1538, 2017.

Jegou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Koloskova, A., Stich, S. U., and Jaggi, M. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pp. 3478–3487, 2019.

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

Li, X. and Orabona, F. On the convergence of stochastic gradient descent with adaptive stepsizes. In *International Conference on Artificial Intelligence and Statistics*, pp. 983–992, 2019.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 5330–5340, 2017.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. Deep gradient compression: Reducing the communication bandwidth for distributed training. *International Conference on Learning Representations*, 2018.

Lu, S., Zhang, X., Sun, H., and Hong, M. Gnsd: A gradient-tracking based nonconvex stochastic algorithm for decentralized optimization. In *2019 IEEE Data Science Workshop (DSW)*, pp. 315–321, 2019.

Luo, L., Xiong, Y., Liu, Y., and Sun, X. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference for Learning Representations*, 2019.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Nazari, P., Tarzanagh, D. A., and Michailidis, G. Dadam: A consensus-based distributed adaptive gradient method for online optimization. *arXiv preprint arXiv:1901.09109*, 2019.

Nedic, A. and Ozdaglar, A. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48, 2009.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Shi, W., Ling, Q., Wu, G., and Yin, W. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

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

Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. D^2: Decentralized training over decentralized data. In *International Conference on Machine Learning*, pp. 4848–4856, 2018.

Tang, H., Yu, C., Lian, X., Zhang, T., and Liu, J. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *International Conference on Machine Learning*, pp. 6155–6165, 2019.

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. Atomo: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems*, pp. 9850–9861, 2018.

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

Ward, R., Wu, X., and Bottou, L. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pp. 6677–6686, 2019.

Yan, Y., Yang, T., Li, Z., Lin, Q., and Yang, Y. A unified analysis of stochastic momentum methods for deep learning. In *International Joint Conference on Artificial Intelligence*, pp. 2955–2961, 2018.

Yuan, K., Ling, Q., and Yin, W. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

Zaheer, M., Reddi, S., Sachan, D., Kale, S., and Kumar, S. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pp. 9793–9803, 2018.

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