Toward Communication Efficient Adaptive Gradient Method

Anonymous Author(s)*

ABSTRACT

In recent years, distributed optimization is proven to be an effective approach to accelerate training of large scale machine learning models such as deep neural networks. With the increasing computation power of GPUs, the bottleneck of training speed in distributed training is gradually shifting from computation to communication. Meanwhile, in the hope of training machine learning models on mobile devices, a new distributed training paradigm federated learning is proposed. The communication time in federated learning is especially important due to the low bandwidth of mobile devices. Various approaches to improve communication efficiency are proposed for federated learning. Yet, most of them are designed with SGD as the prototype training algorithm. While adaptive gradient methods are very effective for training neural nets, the study of adaptive gradient methods in federated learning is scarce. In this paper, we study the design of adaptive gradient methods in federated learning. As a result, we propose an adaptive gradient method that can guarantee both convergence and communication efficiency for federated learning.

CCS CONCEPTS

• **Theory of computation** → **Nonconvex optimization**; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Federated Learning, Adaptive Method, Optimization, Convergence Analysis

ACM Reference Format:

Anonymous Author(s). 2020. Toward Communication Efficient Adaptive Gradient Method. In *FODS'20: ACM-IMS Foundations of Data Science Conference, October 18 – 20, 2020, Seattle, WA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Distributed training is proven to be a very successful way of accelerating training large scale machine learning models. With the advances of computing power and algorithmic design, one can now train models that need to be trained for days even weeks in the past within just a few minutes [27]. When the computing power is high compared with the network bandwidth connecting different machines in distributed training, the training speed can be bottlenecked by the transmission of gradients and parameters. Such situation occurs increasingly more often in the past a few years due to the rapid growth in computer power of GPUs. Therefore, reducing communication overhead is gradually becoming an important research direction in distributed training [2, 15, 24]. In addition, a new training paradigm called *Federated Learning* [11, 17] was proposed recently, where models are trained distributively with mobile devices as workers and data holders. Consider the case where the data is stored and the model is trained on each user's mobile device. The existence of limited bandwidth (e.g. 1MB/s) necessitates the development of communication efficient training algorithms. Moreover, it is unpractical for every user to keep communicating with the central server, due to the power condition or wireless connection of the device. To cope with these communication issues, an SGD-based algorithm with periodic model averaging called *Federated Averaging* is proposed in McMahan et al. [17].

Federated learning extends the traditional parameter server setting, where the data are located on different workers and a parameter server is set to coordinate the training. In the parameter server setting, many effective communication reduction techniques were proposed such as gradient compression [3, 15] and quantization [2, 24, 26] for distributed SGD. In federated learning, one can substantially reduce high communication cost by avoiding frequent transmission between local workers and central server. Instead, the workers train and maintain their own models locally, and the central server aggregates and averages the model parameters of all workers periodically. After averaging, new model parameter is fed back to each local worker, which starts another round of "local training + global averaging". Some of the aforementioned communication reduction techniques can also be incorporated into Federated Averaging to further reduce the communication cost [20].

Despite the active efforts in the community on improving algorithms based on periodic model averaging [8, 20], the prototype algorithm is still SGD. However, it is known that adaptive gradient methods such as AdaGrad [6], Adam [10] and AMSGrad [19] usually perform better than SGD when training neural nets, in terms of difficulty of parameter tuning and convergence speed at early stages. This motivates us to study adaptive gradient methods in federated learning.

Our contributions. In this paper, we study how to incorporate adaptive gradient method into federated learning. Specifically, we show that unlike SGD, a naive combination of adaptive gradient methods and periodic model averaging results in algorithms that may fail to converge. Furthermore, based on federated averaging and decentralized training, we propose an adaptive gradient method with communication cost sublinear in T that is guaranteed to converge to stationary points with a rate of $O(\sqrt{d}/\sqrt{TN})$. Our proposed method enjoys the benefit from both worlds: the fast convergence performance of adaptive gradient methods and low communication cost of federated learning.

FODS'20, October 18 – 20, 2020, Seattle, WA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

2 RELATED WORK

Federated learning. A classical framework for distributed training is the parameter server framework. In such a setting, a parameter server is used to coordinate training and the main computation (e.g. gradient computation) is offloaded to workers in parallel. For SGD under this framework [13, 18, 32], the gradients can be computed by workers on their local data and sent to the parameter server which will aggregates the gradients and update the model parameters. Recently, a variant of the parameter server setting called federated learning [11, 17] draws much attention. One of the key features of federated learning is that workers are likely to be mobile devices which share a low bandwidth with the parameter server. Thus, communication cost plays a more important role in federated learning compared with the traditional parameter server setting. To reduce the communication cost, McMahan et al. [17] proposed an algorithm called Federated Averaging which is a version of parallel SGD with local updates. In Federated Averaging, each worker updates their local parameters locally using SGD and the local parameters are synchronized by periodic averaging through the parameter server. The algorithm is also called local SGD or K-step SGD in some other works [23, 28, 31]. Theoretically, it is proven in [28] that local SGD can save a communication factor of $O(T^{1/4})$ while achieving the same convergence rate as vanilla SGD.

Adaptive gradient methods. Adaptive gradient methods usually refer to the class of gradient based optimization algorithms that adaptively update their learning rate (for each parameter coordinate) using historical gradients. Adaptive gradient methods such as Adam Kingma and Ba [10], AdaGrad [6], AdaDelta [29] are commonly used for training deep neural networks. It has been observed empirically that in many cases adaptive methods can outperform SGD or other methods in terms of convergence speed. There are also many variants trying to improve different aspects of these algorithms, e.g. Agarwal et al. [1], Chen et al. [5], Keskar and Socher [9], Luo et al. [16], Reddi et al. [19]. The work Reddi et al. [19] pointed out the divergence issue of Adam, and proposed AMSGrad algorithm for a fix. Moreover, increasing efforts are investigated into theoretical analysis of these algorithms Chen et al. [4], Li and Orabona [14], Staib et al. [22], Ward et al. [25], Zhou et al. [30], Zou and Shen [33], Zou et al. [34]. In this paper, we embed adaptive gradient methods into federated learning and provide rigorous convergence analysis. The proposed algorithm can achieve the same convergence rate as its vanilla version, while enjoying the communication reduction brought by periodic model averaging.

3 DISTRIBUTED TRAINING WITH PERIODIC MODEL AVERAGING

In this section, we will introduce our problem setting and the periodic model averaging framework.

Notations. Throughout this paper, $x_{t,i}$ is denote x at node i and iteration t, a/b is element-wise division when a and b are vectors of the same dimension, $\cdot$ denotes element-wise multiplication, $(a)_j$ denotes the j-th coordinate of vector a, a^p denotes element-wise power when a is a vector.

3.1 Distributed optimization

We consider the following formulation for distributed training (with N nodes)

$$\min_x \frac{1}{N}\sum_{i=1}^{N} f_i(x)$$

where f_i can be considered as the averaged loss over data at worker i and the function can only be accessed by the worker itself. For instance, for training neural nets, f_i can be viewed as the average loss of data located at the i-th node.

In this paper, we consider the case where f_i's might be nonconvex. For convergence analysis, we will make the following assumptions.

Assumptions:

A1: f_i is differentiable and L-smooth, i.e. $\|\nabla f_i(x) - \nabla f_i(y)\| \le L\|x-y\|,\ \forall x, y$.

A2: Unbiased gradient estimator, $\mathbb{E}[g_{t,i}] = \nabla f_i(x_{t,i})$.

A3: Coordinate-wise bounded variance for the gradient estimator $\mathbb{E}[(g_{t,i})_j^2] \le \sigma^2, \forall j \in [d]$.

A4: Bounded gradient estimator, $\|g_{t,i}\|_\infty \le G_\infty$.

The assumptions A1, A2, A3 are standard in stochastic optimization. A4 is a little stronger than bounded variance assumption (A2). Such an assumption is commonly used in analysis for adaptive gradient methods [4, 25] to simplify the convergence analysis by bounding possible adaptive learning rates.

3.2 Periodic model averaging

Recently, a new trend in algorithm design for distributed training is using periodic model averaging to reduce communication cost. This is motivated by the fact that in some circumstances where distributed optimization is used, the computation time is dominated by communication time. This phenomenon is significantly exacerbated in federated learning, where the bandwidth is relatively small.

Local SGD (i.e. Federated Averaging [11, 31]) is featured by the use of periodic model averaging with SGD (see Algorithm 1). Periodic model averaging can reduce the number of communication rounds and it is shown in Stich [23], Yu et al. [28] that by using periodic model averaging, one can achieve the same convergence rate as distributed SGD with a communication cost sublinear in T. Note that, in practical scenarios, samples allocated on each node may not be i.i.d.. In the example of training on mobile devices, if the data on each device is collected from each single user, samples on a node will no longer be randomly drawn from the population.

Algorithm 1: Local SGD (with N nodes)

Input: learning rate α, current point x_t, $\hat{v}_{0,i} = \epsilon\mathbf{1}, \forall i$,
$g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
if $t \mod k \neq 0$ **then**
  $\hat{v}_t = \hat{v}_{t-1}$
  $x_{t+1,i} \leftarrow x_{t,i} - \alpha g_{t,i}$
else
  $x_{t+1,i} \leftarrow \frac{1}{N}\sum_{j=1}^{N}\left(x_{t,j} - \alpha g_{t,i}\right)$
end if

Since local SGD is mainly used for training neural nets in federated learning, it is natural to consider using adaptive gradient methods in such a setting to bring the benefit of adaptive gradient methods.

In the remaining sections of this paper, we will study how to use periodic model averaging with adaptive gradient methods.

4 ADAPTIVE GRADIENT METHODS WITH PERIODIC MODEL AVERAGING

In this section, we explore the possibilities of combining adaptive gradient method with periodical model averaging. We use AMSGrad Reddi et al. [19] as our prototype algorithm due to its nice convergence guarantee and superior empirical performance. The proposed scheme will be called local AMSGrad.

4.1 Divergence of naive local AMSGrad

Similar to local SGD, the most straightforward way to combine AMSGrad with periodic model averaging works as follows:

1. Each node run AMSGrad locally.
2. The variable $\{x_{t,i}\}_{i=1}^N$ are averaged every k iterations.

The algorithm's pseudo code is shown in Algorithm 2.

Algorithm 2: Naive local AMSGrad (with N nodes)

Input: learning rate α, point x_t, $m_{0,i} = 0, \hat{v}_{0,i} = \epsilon \mathbf{1}, \forall i$
$g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
$m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i}$
$v_{t,i} = \beta_2 v_{t-1,i} + (1-\beta_2) g_{t,i}^2$
$\hat{v}_{t,i} = \max(v_{t,i}, \hat{v}_{t-1,i})$
if $t \mod k \neq 0$ **then**
  $x_{t+1,i} \leftarrow x_{t,i} - \alpha \frac{m_{t,i}}{\sqrt{\hat{v}_{t,i}}}$
else
  $x_{t+1,i} \leftarrow \frac{1}{N} \sum_{j=1}^N \left(x_{t,j} - \alpha \frac{m_{t,j}}{\sqrt{\hat{v}_{t,i}}} \right)$
end if

Since Algorithm 2 is very similar to Algorithm 1 except for the use of adaptive learning rate, given Algorithm 1 is guaranteed to converge to stationary points, one may expect that Algorithm 2 is also guaranteed to converge. However, this is not the case. Algorithm 2 can fail to converge to stationary points, due to the possibility that the adaptive learning rates on different nodes are different. We show this possibility in Theorem 4.1.

Theorem 4.1. *There exist a problem where Algorithm 2 converges to non-stationary points no matter how small the stepsize is.*

Proof: We prove by providing a counter example. Consider a simple 1-dimensional case where $N = 3$ with

$$f_1 = \begin{cases} 2x^2, & |x| \le 1, \\ 4|x| - 2, & |x| > 1. \end{cases} \qquad f_2 = f_3 = \begin{cases} -0.5x^2, & |x| \le 1, \\ -|x| + 0.5, & |x| > 1. \end{cases}$$

It is clear that $f(x) = \sum_{i=1}^3 f_i(x)$ has a unique stationary point at $x = 0$ such that $\nabla f(x) = 0$. Suppose $\alpha = 0.1$, $\beta_1 = 0$, $\beta_2 = 0.5$ and the initial point is $x_{0,i} = 5$ for $i = 1, 2, 3$. Also suppose that $k = 1$, i.e. we average local parameters after every iteration. At $t = 1$, for the first node associated with f_1, we have $m_{0,1} = g_{0,1} = 4$, and $\hat{v}_{0,1} = 0.5 \times 4^2 = 8$. For $i = 2, 3$, we have $m_{0,i} = g_{0,i} = -1$ and $\hat{v}_{0,i} = 0.5$. Since in the naive method every node keeps its own learning rate, after the first update we have $x_{1,1} = 5 - \frac{0.1 \times 4}{2\sqrt{2}} \approx 4.86$, $x_{1,2} = x_{1,3} = 5 + 0.1\sqrt{2} \approx 5.15$. By Algorithm 2, we have $x_1 \approx 5.05$, which heads towards the opposite direction of the true stationary point. We can then continue to show that for $t > 1$, $m_{t,1} = 4$, $\hat{v}_{t,1} = (1 - 0.5^t) \times 4^2$ and $m_{t,i} = -1$, $\hat{v}_{t,i} = 1 - 0.5^t$ for $i = 2, 3$. Therefore, we always update $x_{t,1}$ by $-0.1/\sqrt{1 - 0.5^t}$, while updating $x_{t,2}$ and $x_{t,3}$ by $0.1/\sqrt{1 - 0.5^t}$. As a result, the averaged model parameter will head towards $+\infty$, instead of 0. The above argument can be trivially extended to arbitrary stepsize since the gradients do not change in the linear region of the function. □

Given the divergence shown in the proof of Theorem 4.1, we know a naive combination of periodic model averaging and adaptive gradient methods may not be valid even in a very simple case. By diving into the example where the algorithm fails, one can notice that the divergence is caused by the non-consensus of adaptive learning rates on different nodes. This suggests that we should keep the adaptive learning rate same at different nodes. Next, we incorporate this idea into algorithm design to use shared adaptive learning rate on different nodes.

4.2 Local AMSGrad with shared adaptive learning rates

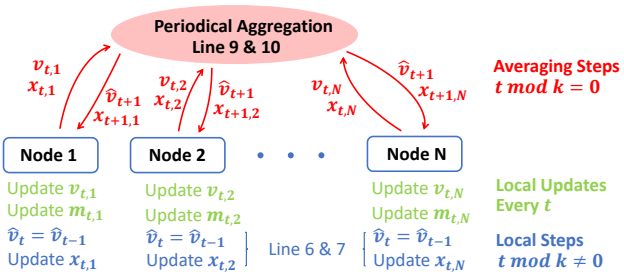

Figure 1: Illustration of the proposed scheme (Algorithm 3) for local AMSGrad with shared adaptive learning rate.

In the last section, we showed an example where a naive combination of AMSGrad and periodic model averaging may diverge. The key divergence mechanism is due to the use of different adaptive learning rates on different nodes. A natural way to improve it is to enforce different nodes having the same adaptive learning rate and we instantiate this idea in Figure 1 and Algorithm 3

Compared with Algorithm 2, Algorithm 3 introduces a periodic averaging step for $v_{t,i}$ and updates $\hat{v}_t$ at the server side, the same $\hat{v}_t$ is used for local updates of different nodes. The intuition behind the design is that, since $v_{t,i}$ can be viewed as second moment estimation of the gradients and the average of $v_{t,i}$ is also an estimation of second moment, we expect the performance of the proposed method to be close to original AMSGrad. Notice that this is not the only way to synchronize adaptive learning rate at different nodes, e.g. one can keep $\hat{v}_{t,i}$ locally and use the average of $\hat{v}_{t,i}$ obtained at the previous averaging step as the adaptive learning rate during

Algorithm 3: local AMSGrad (with N nodes)

Input: learning rate α, point x_t, $m_{t,i} = 0, \hat{v}_{0,i} = \epsilon \mathbf{1}, \forall i$
$g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
$m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i}$
$v_{t,i} = \beta_2 v_{t-1,i} + (1-\beta_2) g_{t,i}^2$
if $t \mod k \neq 0$ **then**
  $\hat{v}_t = \hat{v}_{t-1}$
  $x_{t+1,i} \leftarrow x_{t,i} - \alpha \frac{m_{t,i}}{\sqrt{\hat{v}_t}}$
else
  $\hat{v}_t = \max(\frac{1}{N}\sum_{i=1}^N v_{t,i}, \hat{v}_{t-1})$
  $x_{t+1,i} \leftarrow \frac{1}{N}\sum_{j=1}^N \left(x_{t,j} - \alpha \frac{m_{t,j}}{\sqrt{\hat{v}_t}}\right)$
end if

local updates. With the synchronization of adaptive learning rates, the divergence example in Theorem 4.1 is not valid. However, the convergence guarantee of the proposed algorithm is still not clear since it uses periodic averaging with adaptive learning rates and momentum. In the next section, we will establish the convergence guarantee of the proposed algorithm.

5 CONVERGENCE OF LOCAL AMSGRAD

In this section, we analyze the convergence behavior of Algorithm 3. We have Theorem 5.1 to characterize its convergence rate.

Theorem 5.1. *For Algorithm 3, if A1 - A4 are satisfied, define* $\overline{x}_t = \frac{1}{N}\sum_{i=1}^N x_{t,i}$, *set* $\alpha = \min(\frac{\sqrt{N}}{\sqrt{Td}}, \frac{\sqrt{\epsilon}}{4L})$, *we have for any* $T \geq \frac{16NL^2}{\epsilon d}$,

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^T \mathbb{E}\left[\left\|\frac{\nabla f(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2\right] \\
\leq& 8\frac{\sqrt{d}}{\sqrt{TN}}(\mathbb{E}[f(\overline{x}_1)] - \min_x f(x)) + 8L\frac{\sqrt{d}}{\sqrt{TN}}\sigma^2\frac{1}{\epsilon} \\
&+ 8\frac{d}{T}\frac{\beta_1}{1-\beta_1}G^2\frac{1}{\epsilon^{1/2}} + 8\frac{LN}{T^2}\frac{\beta_1^2}{(1-\beta_1)^2}\frac{G^2}{\epsilon} \\
&+ 8\frac{N}{T}L\left(\frac{\beta_1^2}{(1-\beta_1)^2} + 5(k-1)^2\right)\frac{G^2}{\epsilon^{1.5}} \qquad (1)
\end{aligned}$$

From Theorem 5.1, we can analyze the effect of different factors on the convergence rate of Algorithm 3. The first two terms are standard in convergence analysis, which are introduced by initial function value and the variance of the gradient estimator. Notice that the $\sqrt{d}$ factor in the two terms is due to the bounded coordinate-wise variance assumption, which makes the total variance of the gradient estimator upper bounded by $d\sigma^2$. One can remove the dependency on d on these two terms by assuming bounded total variance. The terms diminishes with β_1 are introduced by the use of momentum in $m_{t,i}$. The most important term for communication efficiency is the term grows with k. This is due to the bias on update directions introduced by local updates. It is clear that this term will not dominate RHS of (1) when $k \leq O(\frac{T^{1/4}d^{1/4}}{\sqrt{N}})$. Thus, one can achieve a convergence rate of $O(\sqrt{d}/\sqrt{TN})$ with communication rounds sublinear in T. This matches the convergence rate of vanilla AMSGrad $O(\sqrt{d}/\sqrt{T})$ proven in [4, 30]. Another aspect deserve some discussion is the dependency on ϵ. One may expect the RHS to be large when ϵ is small. However, this is only true when the gradients are also small such that their L_∞ norm is in the order of ϵ. All the dependencies on ϵ appears in lower bounding $\hat{v}_t$. With the update rule of $\hat{v}_t$, it will quickly become at least the same order as second moment of stochastic gradients, which does not achieve the worst case. Thus, one should expect that when ϵ is really small, the worst case convergence rate is not achieved in practice. As for the convergence measure, one can easily lower bound LHS of (5.1) by the traditional measure $\frac{1}{T}\sum_{t=1}^T \mathbb{E}[\|\nabla f(\bar{x}_t)\|^2]$ using the fact that $\|\hat{v}_t\| \leq G^2$. To wrap things up, we have simplified convergence rate in Corollary 5.2.

Corollary 5.2. *For Algorithm 3, if A1 - A4 are satisfied, set* $k \leq T^{1/4}d^{1/4}/\sqrt{N}$, *for* $T \geq \max(Nd, \frac{16NL^2}{\epsilon d})$ *when* $\beta_1 > 0$ *and for* $T \geq \frac{16NL^2}{\epsilon d}$ *when* $\beta_1 = 0$, *we have*

$$\frac{1}{T}\sum_{t=1}^T \mathbb{E}\left[\|\nabla f(\overline{x}_t)\|^2\right] \leq O\left(\frac{\sqrt{d}}{\sqrt{TN}}\right)$$

Again, the $\sqrt{d}$ factor is due to the bounded coordinate-wise variance assumption A3, one can easily remove the d dependency by assuming bounded total variance. In the remaining parts of this section, we will devote to the proof of Theorem 5.1.

Proof of Theorem 5.1: To prove the convergence of local AMSGrad, we first define an auxiliary sequence of iterates

$$\overline{z}_t = \overline{x}_t + \frac{\beta_1}{1-\beta_1}(\overline{x}_t - \overline{x}_{t-1}) \qquad (2)$$

where $\overline{x}_t = \frac{1}{N}\sum_{i=1}^N x_{t,i}$ and we define $x_0 \triangleq x_1$.

We have following property for the new sequence $\overline{z}_t$.

Lemma 5.3. *For* $\overline{z}_t$ *defined in* (2)

$$\overline{z}_{t+1} - \overline{z}_t = \alpha\frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}} - \frac{1}{\sqrt{\hat{v}_t}}\right)\odot \overline{m}_{t-1} - \alpha\frac{\overline{g}_t}{\sqrt{\hat{v}_t}} \qquad (3)$$

where $\overline{m}_{t-1} = \frac{1}{N}\sum_{i=1}^N m_{t-1,i}$ *and* $\overline{g}_t = \frac{1}{N}\sum_{i=1}^N g_{t,i}$.

Proof: We have

$$\begin{aligned}
\overline{z}_{t+1} - \overline{z}_t &= \overline{x}_{t+1} + \frac{\beta_1}{1-\beta_1}(\overline{x}_{t+1} - \overline{x}_t) - \overline{x}_t - \frac{\beta_1}{1-\beta_1}(\overline{x}_t - \overline{x}_{t-1}) \\
&= \frac{1}{1-\beta_1}(\overline{x}_{t+1} - \overline{x}_t) - \frac{\beta_1}{1-\beta_1}(\overline{x}_t - \overline{x}_{t-1}) \\
&= \frac{\alpha}{1-\beta_1}\frac{\overline{m}_t}{\sqrt{\hat{v}_t}} - \frac{\alpha\beta_1}{1-\beta_1}\frac{\overline{m}_{t-1}}{\sqrt{\hat{v}_{t-1}}}.
\end{aligned}$$

By the updating rule of Algorithm 3, we have $\overline{m}_t = \beta_1\overline{m}_{t-1} + (1-\beta_1)\overline{g}_t$. Thus,

$$\overline{z}_{t+1} - \overline{z}_t = \frac{\alpha\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_t}} - \frac{1}{\sqrt{\hat{v}_{t-1}}}\right)\odot \overline{m}_{t-1} + \alpha\frac{\overline{g}_t}{\sqrt{\hat{v}_t}},$$

which completes the proof. □

We will use this auxiliary sequence to prove convergence of the algorithm.

By Lipschitz continuity, we have

$$f(\overline{z}_{t+1}) \leq f(\overline{z}_t) + \langle \nabla f(\overline{z}_t), \overline{z}_{t+1} - \overline{z}_t\rangle + \frac{L}{2}\|\overline{z}_{t+1} - \overline{z}_t\|^2$$

and thus

$$\begin{aligned}&-\mathbb{E}[\langle\nabla f(\overline{z}_t),\overline{z}_{t+1}-\overline{z}_t\rangle]\\ \leq&\mathbb{E}[f(\overline{z}_t)]-\mathbb{E}[f(\overline{z}_{t+1})]+\frac{L}{2}\mathbb{E}[\|\overline{z}_{t+1}-\overline{z}_t\|^2]\end{aligned}\tag{4}$$

where the expectation is taken over all the randomness of stochastic gradients until iteration t.

What we need to do next is to upper bound the second-order term on RHS of (4) and characterized the effective descent in the first order term (LHS of (4)). We first characterize the effective descent.

By (3), we can write the first-order term as

$$\begin{aligned}&\langle\nabla f(\overline{z}_t),\overline{z}_{t+1}-\overline{z}_t\rangle\\ =&\alpha\langle\nabla f(z_t),\frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}}-\frac{1}{\sqrt{\hat{v}_t}}\right)\odot\overline{m}_{t-1}\\ &-\alpha\langle\nabla f(z_t),\frac{\overline{g}_t}{\sqrt{\hat{v}_t}}\rangle\end{aligned}\tag{5}$$

Since $\hat{v}_t$ is independent of $\overline{g}_t$ and $\mathbb{E}[g_{t,i}]=\nabla f_i(x_{t,i})$, taking expectation on both sides of (5), we have

$$\begin{aligned}&\mathbb{E}\left[\langle\nabla f(\overline{z}_t),\overline{z}_{t+1}-\overline{z}_t\rangle\right]\\ =&\alpha\mathbb{E}\left[\langle\nabla f(z_t),\frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}}-\frac{1}{\sqrt{\hat{v}_t}}\right)\odot\overline{m}_{t-1}\right]\\ &-\alpha\mathbb{E}\left[\langle\nabla f(z_t),\frac{\overline{\nabla f}(x_t)}{\sqrt{\hat{v}_t}}\rangle\right]\end{aligned}$$

where $\overline{\nabla f}(x_t)=\frac{1}{N}\sum_{i=1}^N\nabla f_i(x_{t,i})$.

In the next, we will try to split out a descent quantity from $\mathbb{E}\left[\langle\nabla f(z_t),\frac{\overline{\nabla f}(x_t)}{\sqrt{\hat{v}_t}}\rangle\right]$. Because $\langle a,b\rangle=\frac{1}{2}(\|a-b\|^2-\|a\|^2-\|b\|^2)$, we can transform this term into

$$\begin{aligned}\langle\nabla f(\overline{z}_t),\frac{\overline{\nabla f}(x_t)}{\sqrt{\hat{v}_t}}\rangle=&\frac{1}{2}\left\|\frac{\nabla f(\overline{z}_t)}{\hat{v}_t^{1/4}}\right\|^2+\frac{1}{2}\left\|\frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2\\ &-\left\|\frac{\nabla f(\overline{z}_t)-\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2\end{aligned}\tag{6}$$

where the first two quantities on RHS of the equality will contribute to the descent of objective in a single optimization step, while the last term is the possible ascent introduced by the bias on the stochastic gradients. We need to bound the last term on RHS of (6) in the next which is given by Lemma 5.4.

Lemma 5.4. *For Algorithm 3, we have*

$$\begin{aligned}&\left\|\frac{\nabla f(\overline{z}_t)-\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2\\ \leq&\frac{L}{\min_j(\hat{v}_t^{1/2})_j}\left(2\frac{\beta_1^2}{(1-\beta_1)^2}+8(k-1)^2\right)\alpha^2 d\frac{G^2}{\epsilon}\end{aligned}\tag{7}$$

Proof: First, we have

$$\begin{aligned}&\left\|\frac{\nabla f(\overline{z}_t)-\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2=\left\|\frac{\frac{1}{N}\sum_{i=1}^N(\nabla f_i(\overline{z}_t)-\nabla f_i(x_{t,i}))}{\hat{v}_t^{1/4}}\right\|^2\\ \leq&\frac{1}{N}\sum_{i=1}^N\left\|\frac{\nabla f_i(\overline{z}_t)-\nabla f_i(x_{t,i})}{\hat{v}_t^{1/4}}\right\|^2\\ \leq&\frac{1}{N}\sum_{i=1}^N 2\left(\left\|\frac{\nabla f_i(\overline{z}_t)-f_i(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2+\left\|\frac{\nabla f_i(\overline{x}_t)-\nabla f_i(x_{t,i})}{\hat{v}_t^{1/4}}\right\|^2\right)\end{aligned}\tag{8}$$

where the last inequality is due to Cauchy-Schwartz.

Using Lipschitz property of ∇f_i, we can further bound the differences of gradients on RHS of (8) by

$$\begin{aligned}&\frac{2}{N}\sum_{i=1}^N\left\|\frac{\nabla f_i(\overline{z}_t)-f_i(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2\\ \leq&\frac{2}{N}\sum_{i=1}^N\frac{\|\nabla f_i(\overline{z}_t)-f_i(\overline{x}_t)\|^2}{\min_j(\hat{v}_t^{1/2})_j}\leq\frac{2}{N}\sum_{i=1}^N\frac{L\|\overline{z}_t-\overline{x}_t\|^2}{\min_j(\hat{v}_t^{1/2})_j}\end{aligned}\tag{9}$$

Similarly,

$$\frac{2}{N}\sum_{i=1}^N\left\|\frac{\nabla f_i(\overline{x}_t)-\nabla f_i(x_{t,i})}{\hat{v}_t^{1/4}}\right\|^2\leq\frac{2}{N}\sum_{i=1}^N\frac{L\|\overline{x}_t-x_{t,i}\|^2}{\min_j(\hat{v}_t^{1/2})_j}\tag{10}$$

The next step will be bounding $\|\overline{z}_t-\overline{x}_t\|^2$ and $\|\overline{x}_t-x_{t,i}\|^2$ using the update rule of x and z. For the difference between $\overline{z}_t$ and $\overline{x}_t$, we have

$$\begin{aligned}\sum_{i=1}^N\|\overline{z}_t-\overline{x}_t\|^2=&\frac{\beta_1^2}{(1-\beta_1)^2}\sum_{i=1}^N\|\overline{x}_t-\overline{x}_{t-1}\|^2\\ =&\frac{\beta_1^2}{(1-\beta_1)^2}\alpha^2N\left\|\frac{\overline{m}_{t-1}}{\sqrt{\hat{v}_{t-1}}}\right\|^2\\ \leq&\frac{\beta_1^2}{(1-\beta_1)^2}\alpha^2Nd\frac{G^2}{\epsilon}\end{aligned}\tag{11}$$

where $\overline{m}_t=\frac{1}{N}\sum_{i=1}^N m_{t,i}$.

For the second term containing the consensus error $\overline{x}_t-x_{t,i}$, we have

$$\sum_{i=1}^N\|\overline{x}_t-x_{t,i}\|^2\leq4N(k-1)^2\alpha^2d\frac{G^2}{\epsilon}\tag{12}$$

by Lemma 5.5.

Lemma 5.5. *For iterates produced by Algorithm 3, we have* $\forall i\in[N]$,

$$\|\overline{x}_t-x_{t,i}\|^2\leq4(k-1)^2\alpha^2d\frac{G^2}{\epsilon}$$

Proof: For the ease of presentation, we need to introduce some notations. Denote $\lfloor t\rfloor_k$ to be the largest multiple of k that is less than t, we have

$$x_{t,i}=\overline{x}_{\lfloor t\rfloor_k+1}-\alpha\sum_{l=\lfloor t\rfloor_k+1}^{t-1}\frac{m_{l,i}}{\sqrt{\hat{v}_l}}$$

since $x'_{t,i}s$ are averaged on steps $ck+1, c\in\mathbb{N}_0$.

Thus, we have

$$\|\overline{x}_t - x_{t,i}\|^2 = \alpha^2 \left\| \sum_{l=\lfloor t \rfloor_k+1}^{t-1} \left(\frac{m_{l,i}}{\sqrt{\hat{v}_l}} - \frac{1}{N}\sum_{o=1}^{N} \frac{m_{l,o}}{\sqrt{\hat{v}_o}} \right) \right\|^2 \leq 4N(k-1)^2\alpha^2 d \frac{G^2}{\epsilon}$$

because $(t-1)-(\lfloor t \rfloor_k+1)+1 \leq k-1$, and $(\hat{v}_o)_j \geq \epsilon, \forall j \in [d], o \in [N]$, and $\|m_{t,o}\|_\infty \leq G, \forall t, o$. □

Combining (8), (9), (10), (11) and (12), we get

$$\left\| \frac{\nabla f(\overline{z}_t) - \overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}} \right\|^2 \leq \frac{L}{\min_j(\hat{v}_t^{1/2})_j} \left(2\frac{\beta_1^2}{(1-\beta_1)^2} + 8(k-1)^2 \right) \alpha^2 d \frac{G^2}{\epsilon}$$

which is the desired bound. □

Thus by (6) and (7), we have

$$\begin{aligned} & -\langle \nabla f(\overline{z}_t), \frac{\overline{\nabla f}(x_t)}{\sqrt{\hat{v}_t}} \rangle \\ \leq & -\frac{1}{2}\left\| \frac{\nabla f(\overline{z}_t)}{\hat{v}_t^{1/4}} \right\|^2 - \frac{1}{2}\left\| \frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}} \right\|^2 \\ & + \alpha^2 \frac{L}{\min_j(\hat{v}_t^{1/2})_j} \left(\frac{\beta_1^2}{(1-\beta_1)^2} + 4(k-1)^2 \right) d\frac{G^2}{\epsilon} \end{aligned} \tag{13}$$

Substitute (13) into (5) and (4), we have

$$\begin{aligned} & \alpha \mathbb{E}\left[\frac{1}{2}\left\| \frac{\nabla f(\overline{z}_t)}{\hat{v}_t^{1/4}} \right\|^2 + \frac{1}{2}\left\| \frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}} \right\|^2 \right] \\ \leq & \mathbb{E}[f(\overline{z}_t)] - \mathbb{E}[f(\overline{z}_{t+1})] + \frac{L}{2}\mathbb{E}[\|\overline{z}_{t+1} - \overline{z}_t\|^2] \\ & + \alpha \mathbb{E}\left[\langle \nabla f(\overline{z}_t), \frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}} - \frac{1}{\sqrt{\hat{v}_t}} \right) \odot \overline{m}_{t-1} \right] \\ & + \alpha^3 \mathbb{E}\left[\frac{L}{\min_j(\hat{v}_t^{1/2})_j} \left(\frac{\beta_1^2}{(1-\beta_1)^2} + 4(k-1)^2 \right) d\frac{G^2}{\epsilon} \right] \end{aligned}$$

Summing over t from 1 to T and divide both sides by $T\alpha$, we get

$$\begin{aligned} & \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\frac{1}{2}\left\| \frac{\nabla f(\overline{z}_t)}{\hat{v}_t^{1/4}} \right\|^2 + \frac{1}{2}\left\| \frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}} \right\|^2 \right] \\ \leq & \frac{1}{T\alpha}(\mathbb{E}[f(\overline{z}_1)] - \mathbb{E}[f(\overline{z}_{T+1})]) + \frac{L}{2}\frac{1}{T\alpha}\underbrace{\sum_{t=1}^{T}\mathbb{E}[\|\overline{z}_{t+1} - \overline{z}_t\|^2]}_{T_1} \\ & + \underbrace{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\langle \nabla f(\overline{z}_t), \frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}} - \frac{1}{\sqrt{\hat{v}_t}} \right) \odot \overline{m}_{t-1} \right\rangle \right]}_{T_2} \\ & + \alpha^2 \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{L}{\min_j(\hat{v}_t^{1/2})_j} \left(\frac{\beta_1^2}{(1-\beta_1)^2} + 4(k-1)^2 \right) d\frac{G^2}{\epsilon} \right] \end{aligned} \tag{14}$$

What remains is to bounded T_1 and T_2. Let's look at T_1 first.

Lemma 5.6. *We have*

$$T_1 \leq 2\alpha^2 \sum_{t=1}^{T} \mathbb{E}\left[\left\| \frac{\overline{\nabla f}(x_t)}{\sqrt{\hat{v}_t}} \right\|^2 + \frac{1}{N}\sigma^2 \left\| \frac{1}{\sqrt{\hat{v}_t}} \right\|^2 \right] + 2\alpha^2 \frac{\beta_1^2}{(1-\beta_1)^2} G^2 \frac{d}{\epsilon}$$

Proof: By (3), we know

$$\begin{aligned} & \|\overline{z}_{t+1} - \overline{z}_t\|^2 \\ = & \alpha^2 \left\| \frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}} - \frac{1}{\sqrt{\hat{v}_t}} \right) \odot \overline{m}_{t-1} - \alpha \frac{\overline{g}_t}{\sqrt{\hat{v}_t}} \right\|^2 \\ \leq & 2\alpha^2 \left(\left\| \frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}} - \frac{1}{\sqrt{\hat{v}_t}} \right) \odot \overline{m}_{t-1} \right\|^2 + \left\| \frac{\overline{g}_t}{\sqrt{\hat{v}_t}} \right\|^2 \right) \end{aligned}$$

In addition, conditioned on all randomness in $\hat{v}_t$ (gradients from iteration 1 until $t-1$), we have

$$\begin{aligned} & \mathbb{E}\left[\left\| \frac{\overline{g}_t}{\sqrt{\hat{v}_t}} \right\|^2 \right] = \mathbb{E}\left[\left\| \frac{\overline{g}_t}{\sqrt{\hat{v}_t}} \right\|^2 \right] = \mathbb{E}\left[\left\| \frac{\overline{g}_t}{\sqrt{\hat{v}_t}} \right\|^2 \right] \\ = & \mathbb{E}\left[\left\| \frac{1}{N}\sum_{i=1}^{N} \frac{g_{t,i}}{\sqrt{\hat{v}_t}} \right\|^2 \right] = \frac{1}{N^2}\mathbb{E}\left[\sum_{j=1}^{N}\sum_{i=1}^{N} \left\langle \frac{g_{t,i}}{\sqrt{\hat{v}_t}}, \frac{g_{t,j}}{\sqrt{\hat{v}_t}} \right\rangle \right] \\ \overset{(a)}{=} & \frac{1}{N^2}\mathbb{E}\left[\sum_{j=1}^{N}\sum_{i=1}^{N} \left\langle \frac{\nabla f_i(x_{t,i}) + \xi_{t,i}}{\sqrt{\hat{v}_t}}, \frac{\nabla f_j(x_{t,j}) + \xi_{t,j}}{\sqrt{\hat{v}_t}} \right\rangle \right] \\ \overset{(b)}{=} & \frac{1}{N^2}\left\| \frac{\sum_{i=1}^{N}\nabla f_i(x_{t,i})}{\sqrt{\hat{v}_t}} \right\|^2 + \frac{1}{N^2}\sum_{i=1}^{N}\mathbb{E}\left[\left\| \frac{\xi_{t,i}}{\sqrt{\hat{v}_t}} \right\|^2 \right] \\ \overset{(c)}{\leq} & \left\| \frac{\overline{\nabla f}(x_t)}{\sqrt{\hat{v}_t}} \right\|^2 + \frac{1}{N}\sigma^2 \left\| \frac{1}{\sqrt{\hat{v}_t}} \right\|^2 \end{aligned}$$

where (a) is a reparameterization of the noises on gradients, (b) is due to the assumption that all elements of $\xi_{t,i}$'s are independent of each other and $\mathbb{E}[(\xi_{t,i})_j] = 0, \forall j \in [d]$, (c) is by the assumption that $E[(\xi_{t,i})_j^2] \leq \sigma$ and the fact that $\hat{v}_t$ and $x_{t,i}$ are both fixed given all gradients before iteration t.

Further, we have

$$\begin{aligned} & \left\| \frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}} - \frac{1}{\sqrt{\hat{v}_t}} \right) \odot \overline{m}_{t-1} \right\|^2 \\ \leq & \frac{\beta_1^2}{(1-\beta_1)^2} G^2 \sum_{j=1}^{d}\left(\frac{1}{\sqrt{(\hat{v}_{t-1})_j}} - \frac{1}{\sqrt{(\hat{v}_t)_j}} \right)^2 \\ \leq & \frac{\beta_1^2}{(1-\beta_1)^2} G^2 \frac{1}{\epsilon^{1/2}} \sum_{j=1}^{d}\left| \frac{1}{\sqrt{(\hat{v}_{t-1})_j}} - \frac{1}{\sqrt{(\hat{v}_t)_j}} \right| \end{aligned}$$

and thus

$$\begin{aligned} & \sum_{t=1}^{T}\left\| \frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}} - \frac{1}{\sqrt{\hat{v}_t}} \right) \odot \overline{m}_{t-1} \right\|^2 \\ \leq & \sum_{t=1}^{T} \frac{\beta_1^2}{(1-\beta_1)^2} G^2 \frac{1}{\epsilon^{1/2}} \sum_{j=1}^{d}\left| \frac{1}{\sqrt{(\hat{v}_{t-1})_j}} - \frac{1}{\sqrt{(\hat{v}_t)_j}} \right| \\ \leq & \frac{\beta_1^2}{(1-\beta_1)^2} G^2 \frac{1}{\epsilon^{1/2}} \sum_{j=1}^{d}\left| \frac{1}{\sqrt{(\hat{v}_0)_j}} \right| = \frac{\beta_1^2}{(1-\beta_1)^2} G^2 \frac{1}{\epsilon} d \end{aligned}$$

where the last inequality is due to non-decreasing property of $\hat{v}_t$.

Combining all above, we get

$$T_1 \leq 2\alpha^2 \sum_{t=1}^{T} \mathbb{E}\left[\left\|\frac{\overline{\nabla f}(x_t)}{\sqrt{\hat{v}_t}}\right\|^2 + \frac{1}{N}\sigma^2\left\|\frac{1}{\sqrt{\hat{v}_t}}\right\|^2\right] + 2\alpha^2\frac{\beta_1^2}{(1-\beta_1)^2}G^2\frac{d}{\epsilon}$$

which completes the proof □.

Now let's look at T_2. We have

$$\begin{aligned} T_2 &= \sum_{t=1}^{T}\mathbb{E}\left[\left\langle \nabla f(\overline{z}_t), \frac{\beta_1}{1-\beta_1}\left(\frac{1}{\sqrt{\hat{v}_{t-1}}} - \frac{1}{\sqrt{\hat{v}_t}}\right)\odot \overline{m}_{t-1}\right\rangle\right] \\ &\leq \mathbb{E}\left[\sum_{t=1}^{T}\frac{\beta_1}{1-\beta_1}G^2\sum_{j=1}^{d}\left|\frac{1}{\sqrt{(\hat{v}_{t-1})_j}} - \frac{1}{\sqrt{(\hat{v}_t)_j}}\right|\right] \\ &\leq \mathbb{E}\left[\frac{\beta_1}{1-\beta_1}G^2\sum_{j=1}^{d}\left|\frac{1}{\sqrt{(\hat{v}_0)_j}}\right|\right] = \frac{\beta_1}{1-\beta_1}G^2 d\frac{1}{\epsilon^{1/2}} \end{aligned}$$

where the second inequality is because $\hat{v}_t$ is non-decreasing.

Substituting the bounds on T_1 and T_2 into (14), we get

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{1}{2}\left\|\frac{\nabla f(\overline{z}_t)}{\hat{v}_t^{1/4}}\right\|^2 + \frac{1}{2}\left\|\frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2\right] \\ &\leq \frac{1}{T\alpha}(\mathbb{E}[f(\overline{z}_1)] - \mathbb{E}[f(\overline{z}_{T+1})]) \\ &\quad + \frac{L}{T}\alpha\sum_{t=1}^{T}\mathbb{E}\left[\left\|\frac{\overline{\nabla f}(x_t)}{\sqrt{\hat{v}_t}}\right\|^2 + \frac{1}{N}\sigma^2\left\|\frac{1}{\sqrt{\hat{v}_t}}\right\|^2\right] \\ &\quad + \frac{1}{T}\frac{\beta_1}{1-\beta_1}G^2 d\frac{1}{\epsilon^{1/2}} + \frac{L}{T}\alpha^2\frac{\beta_1^2}{(1-\beta_1)^2}G^2\frac{d}{\epsilon} \\ &\quad + \alpha^2\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{L}{\min_j(\hat{v}_t^{1/2})_j}\left(\frac{\beta_1^2}{(1-\beta_1)^2} + 4(k-1)^2\right)d\frac{G^2}{\epsilon}\right] \end{aligned} \tag{15}$$

Further, by choosing $\alpha = \min(\frac{\sqrt{N}}{\sqrt{Td}}, \frac{\sqrt{\epsilon}}{4L})$, we have

$$\begin{aligned} L\frac{1}{T}\alpha\sum_{t=1}^{T}\mathbb{E}\left[\left\|\frac{\overline{\nabla f}(x_t)}{\sqrt{\hat{v}_t}}\right\|^2\right] &\leq L\frac{1}{T}\alpha\sum_{t=1}^{T}\mathbb{E}\left[\frac{1}{\sqrt{\epsilon}}\left\|\frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2\right] \\ &\leq \frac{1}{4}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2\right] \end{aligned} \tag{16}$$

Thus, from (15) and (16), we get

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{1}{2}\left\|\frac{\nabla f(\overline{z}_t)}{\hat{v}_t^{1/4}}\right\|^2 + \frac{1}{4}\left\|\frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2\right] \\ &\leq \frac{1}{T\alpha}(\mathbb{E}[f(\overline{z}_1)] - \mathbb{E}[f(\overline{z}_{T+1})]) + L\frac{\sqrt{d}}{\sqrt{TN}}\sigma^2\frac{1}{\epsilon} \\ &\quad + \frac{1}{T}\frac{\beta_1}{1-\beta_1}G^2 d\frac{1}{\epsilon^{1/2}} + \frac{LN}{T^2}\frac{\beta_1^2}{(1-\beta_1)^2}\frac{G^2}{\epsilon} \\ &\quad + \frac{N}{T}L\left(\frac{\beta_1^2}{(1-\beta_1)^2} + 4(k-1)^2\right)\frac{G^2}{\epsilon^{1.5}} \end{aligned}$$

and when $T \geq \frac{16NL^2}{\epsilon d}$, we have

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{1}{2}\left\|\frac{\nabla f(\overline{z}_t)}{\hat{v}_t^{1/4}}\right\|^2 + \frac{1}{4}\left\|\frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2\right] \\ &\leq \frac{\sqrt{d}}{\sqrt{TN}}(\mathbb{E}[f(\overline{z}_1)] - \mathbb{E}[f(\overline{z}_{T+1})]) + L\frac{\sqrt{d}}{\sqrt{TN}}\sigma^2\frac{1}{\epsilon} \\ &\quad + \frac{d}{T}\frac{\beta_1}{1-\beta_1}G^2\frac{1}{\epsilon^{1/2}} + \frac{LN}{T^2}\frac{\beta_1^2}{(1-\beta_1)^2}\frac{G^2}{\epsilon} \\ &\quad + \frac{N}{T}L\left(\frac{\beta_1^2}{(1-\beta_1)^2} + 4(k-1)^2\right)\frac{G^2}{\epsilon^{1.5}} \end{aligned} \tag{17}$$

Now we get the $O(\frac{\sqrt{d}}{\sqrt{TN}})$ convergence rate (when T is sufficiently large) which matches the convergence rate of SGD.

One last thing we need to do is to convert the convergence measure to the norm of gradients of f. We do this by the following Lemma

Lemma 5.7. *For Algorithm 3, we have*

$$\left\|\frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2 \geq \frac{1}{2}\left\|\frac{\nabla f(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2 - 4L(k-1)^2\alpha^2 d\frac{G^2}{\epsilon^{1.5}}$$

Proof: We have

$$\begin{aligned} &\left\|\frac{\overline{\nabla f}(x_t)}{\hat{v}_t^{1/4}}\right\|^2 \\ &\geq \frac{1}{2}\left\|\frac{\nabla f(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2 - \left\|\frac{\overline{\nabla f}(x_t) - \nabla f(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2 \\ &\geq \frac{1}{2}\left\|\frac{\nabla f(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2 - \frac{1}{N}\sum_{i=1}^{N}\left\|\frac{\nabla f_i(x_{t,i}) - \nabla f_i(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2 \\ &\geq \frac{1}{2}\left\|\frac{\nabla f(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2 - 4L(k-1)^2\alpha^2 d\frac{G^2}{\epsilon^{1.5}} \end{aligned}$$

where the first inequality is due to Cauchy-Schwartz, the second inequality is due to Jensen's inequality, and the last inequality is due to Lemma 5.5 and L-smoothness of f_i. □

Then we can transform (17) into

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{1}{8}\left\|\frac{\nabla f(\overline{x}_t)}{\hat{v}_t^{1/4}}\right\|^2\right] \\ &\leq \frac{\sqrt{d}}{\sqrt{TN}}(\mathbb{E}[f(\overline{z}_1)] - \mathbb{E}[f(\overline{z}_{T+1})]) + L\frac{\sqrt{d}}{\sqrt{TN}}\sigma^2\frac{1}{\epsilon} \\ &\quad + \frac{d}{T}\frac{\beta_1}{1-\beta_1}G^2\frac{1}{\epsilon^{1/2}} + \frac{LN}{T^2}\frac{\beta_1^2}{(1-\beta_1)^2}\frac{G^2}{\epsilon} \\ &\quad + \frac{N}{T}L\left(\frac{\beta_1^2}{(1-\beta_1)^2} + 5(k-1)^2\right)\frac{G^2}{\epsilon^{1.5}} \end{aligned}$$

by Lemma 5.7. Multiplying both sides of the above inequality and using the fact that $z_1 = x_1$ complete the overall proof. □

6 EXPERIMENTS

In this section, we compare the performance of local SGD, local AMSGrad, and naive local AMSGrad on a synthetic Gaussian mixture dataset [21] and MNIST [12].

6.1 Experiments on Gaussian mixture dataset

The synthetic dataset is the Gaussian cluster dataset used in Sagun et al. [21]. We use 10 isotropic 100 dimensional Gaussian distributions with different mean and same standard deviation to generate data. The standard deviation of each dimension is 1. The mean of each cluster is generated from an isotropic Gaussian distribution with 0 mean and 1 standard deviation for each dimension. The labels are the corresponding indices of the cluster from which the data are drawn. The model is a neural network with 2 hidden layers with 50 nodes each layer, the activation function is ReLU for both layers. The batch size in training is 256. We use 5 workers with each worker containing data with two labels, this assignment of data corresponds to a non-iid distribution of data on different nodes. The average period k is set to 10. We perform learning rate search on a log scale, increase the learning rate starting from 1e-6 until the algorithm diverges or the performance deteriorates significantly. Specifically, the maximum learning rate is 1 for local SGD and 1e-2 for both local AMSGrad and naive local AMSGrad. We compare the performance of the algorithms with their best learning rate in Figure 2. It can be seen that local SGD and local AMSGrad perform very similar. Naive local AMSGrad is worse than the other two algorithms by a small margin. The performance of different algorithms with different learning rate is shown in Figure 3. It can be seen that all algorithms perform well with suitable learning rate. From this experiments, it seems local AMSGrad do not have clear advantages over other algorithms. In particular, naive local AMSGrad performs not bad albeit it lacks convergence guarantee. We conjecture that this is due to such a simple dataset cannot make the adaptive learning rate on different nodes differ significantly. In the next sets of experiments, we use more complicated real-world datasets to test the performance of different algorithms.

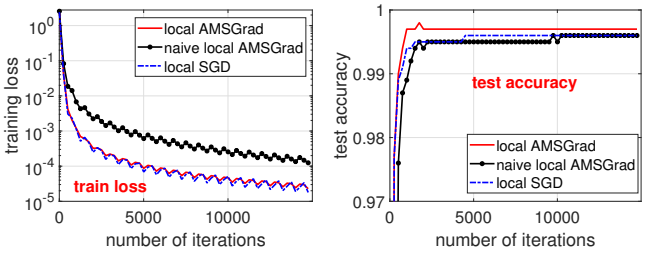

Figure 2: Gaussian mixture dataset: Performance comparison of three different algorithms.

6.2 Experiments on MNIST

In this section, we compare the algorithms on training a convolutional neural network (CNN) on MNIST. Similar to last set of experiments, we perform learning rate search starting from 1e-6 until the algorithm diverges or deteriorates significantly. The average period k is set to 10. We set ϵ to be 1e-4 for both Adam and AMSGrad. The data are distributed on 5 nodes and each node contains data with two labels, there is no overlap on labels between different nodes. We expect such an allocation of data can creates a highly non-iid data distribution, leading to significantly different adaptive learning rates on different nodes. The neural network

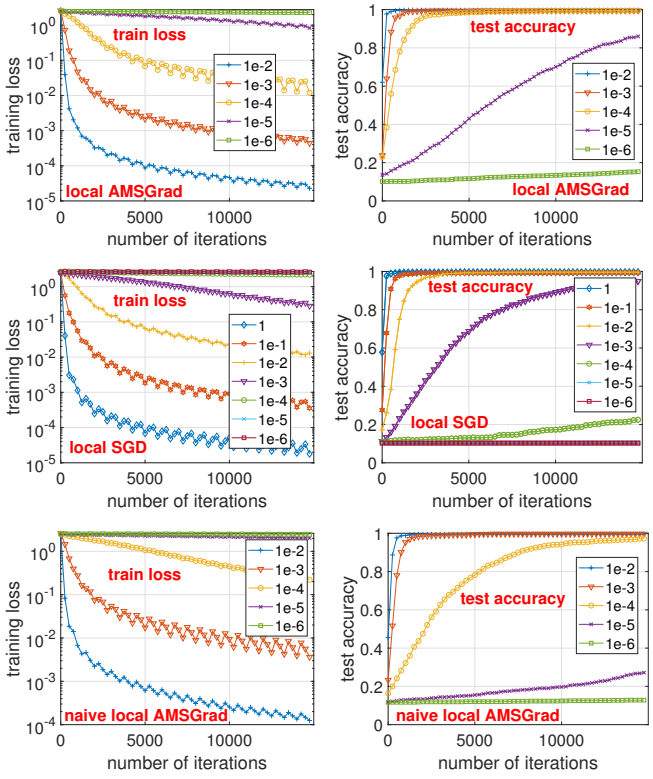

Figure 3: Gaussian mixture dataset: Performance comparison of different stepsizes for three methods.

in the experiments consists of 3 convolution+pooling layers with ReLu activation, followed by a 10 nodes fully connected layer with softmax activation. The first convolution+pooling layer has 20 5x5 filters followed by 2x2 max pooling with stride 2. The second and third convolution+polling layer has 50 filters with other parameters being the same as the first layer. Figure 4 shows the training and testing performance of different algorithms. Specifically, 1e-3 is chosen for local AMSGrad and naive local AMSGrad while 1e-4 is chosen for local SGD. It can be seen that local AMSGrad outperform local SGD by a large margin. The performance of algorithms with different learning rate is shown in Figure 5. It can be seen that naive local AMSGrad has very poor performance with all learning rate choices, local AMSGrad tend to perform better than local SGD on average. The slow convergence of local SGD is also observed in McMahan et al. [17] when the data distribution is non-iid. While the sampling on nodes in McMahan et al. [17] may somehow reduce the influence of non-iid data, the convergence speed of local SGD is significantly impacted by the non-iid distribution in our experiment since we always use all nodes for parameter update.

6.3 Experiments on letter recognition dataset

In this section, we use the letter recognition data set from Frey and Slate [7] to test the performance of different algorithms. We train a fully connected neural network with two hidden layers on the dataset. The first hidden layer has 300 nodes and the second

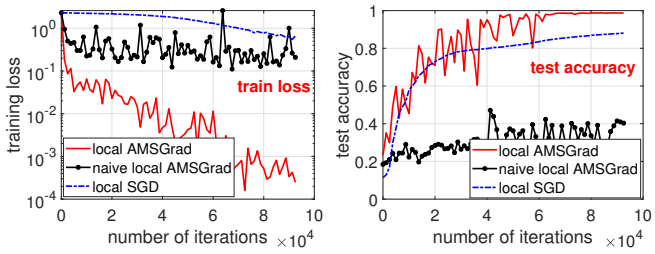

Figure 4: MNIST dataset: Performance comparison of three different algorithms.

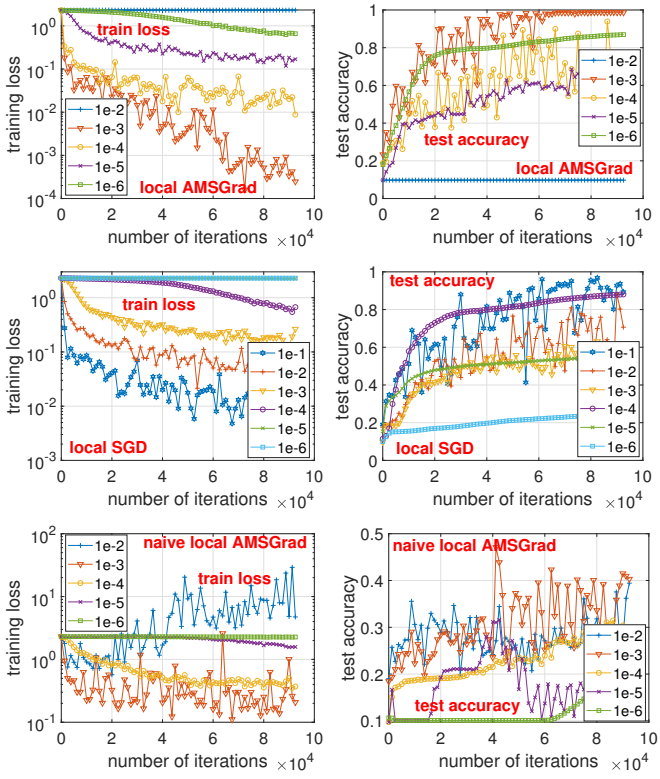

Figure 5: MNIST dataset: Performance comparison of different stepsizes for three methods.

layer has 200 nodes, both of them use ReLU activation function. The learning rate search strategy and other parameter settings are the same as the MNIST experiments. Different from the previous two sets of experiments, the data on the 5 workers are randomly assigned. This corresponds to an i.i.d. data distribution. Thus, all algorithms are expected to work well in this set of experiments. The performance comparison of algorithms with their best learning rate is show in Figure 6. It can be seen that the model trained by all algorithms have over 90% accuracy and local AMSGrad performs best, better than local SGD by about 2%. Note that in this case, naive local AMSGrad is also better than local SGD. Thus, when data distribution is i.i.d., it might be okay to use the naive version of local AMSGrad since it involves less communication. The performance of algorithms with different learning rate is shown in Figure 7, from the figure it can be seen that all algorithms perform quite stable, while the two adaptive gradient methods still outperform local SGD on average.

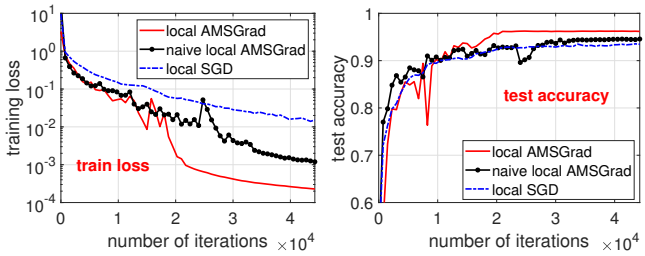

Figure 6: Letter recognition dataset: Performance comparison of three different algorithms.

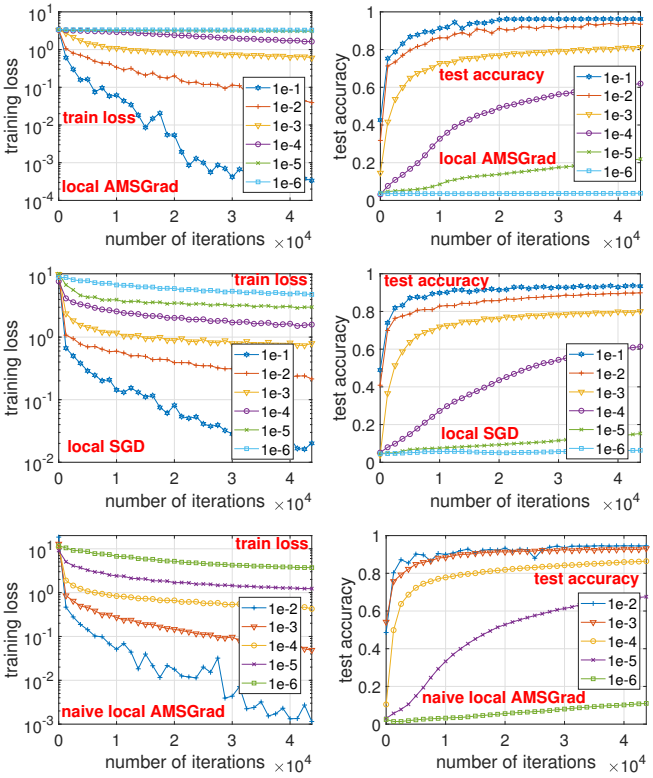

Figure 7: Letter recognition dataset: Performance comparison of different stepsizes for three methods.

7 CONCLUSION

In this paper, we study how to design adaptive gradient methods for federated learning. In particular, we propose the first adaptive gradient methods in the setting of federated learning. Theoretically, the algorithm can match the convergence rate of its centralized version using a number of communication rounds sublinear in T. Experiments show that the proposed algorithm can outperform local SGD and a naive design of adaptive gradient methods.

REFERENCES

[1] Naman Agarwal, Brian Bullins, Xinyi Chen, Elad Hazan, Karan Singh, Cyril Zhang, and Yi Zhang. The case for full-matrix adaptive regularization. *arXiv preprint arXiv:1806.02958*, 2018.

[2] 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*, pages 1709–1720, 2017.

[3] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

[4] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *arXiv preprint arXiv:1808.02941*, 2018.

[5] Zaiyi Chen, Zhuoning Yuan, Jinfeng Yi, Bowen Zhou, Enhong Chen, and Tianbao Yang. Universal stagewise learning for non-convex problems with convergence on averaged solutions. *arXiv preprint arXiv:1808.06296*, 2018.

[6] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[7] Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.

[8] Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Trading redundancy for communication: Speeding up distributed sgd for non-convex optimization. In *International Conference on Machine Learning*, pages 2545–2554, 2019.

[9] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[12] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[13] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pages 583–598, 2014.

[14] Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. *arXiv preprint arXiv:1805.08114*, 2018.

[15] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[16] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

[17] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[18] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[19] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[20] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. *arXiv preprint arXiv:1909.13014*, 2019.

[21] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[22] Matthew Staib, Sashank J Reddi, Satyen Kale, Sanjiv Kumar, and Suvrit Sra. Escaping saddle points with adaptive gradient methods. *arXiv preprint arXiv:1901.09149*, 2019.

[23] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[24] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pages 1299–1309, 2018.

[25] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pages 6677–6686, 2019.

[26] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pages 1509–1519, 2017.

[27] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2019.

[28] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. *arXiv preprint arXiv:1905.03817*, 2019.

[29] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

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

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

[32] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

[33] Fangyu Zou and Li Shen. On the convergence of adagrad with momentum for training deep neural networks. *arXiv preprint arXiv:1808.03408*, 2(3):5, 2018.

[34] Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11127–11135, 2019.