CONVERGENT ADAPTIVE GRADIENT METHODS IN DECENTRALIZED OPTIMIZATION

Anonymous authors
Paper under double-blind review

1 REVIEWER 1

This paper studied the decentralized adaptive gradient methods and provided convergence guarantees. Experiment on MNIST is conducted to show the effectiveness of the proposed approach.

The theoretical result is weak. The linear speedup result is not proved as in (Lian et al. 2017), the benefits of adaptive gradient methods are also not illustrated in the bound in Theorem 2 and Theorem 3.

The learning rate scheme is not practical and does not hold in practice. As illustrated in Theorem 2 and 3, the learning rate α is set to be less than $\epsilon^{0.5}/16L$.

The LHS of Theorem 2 and Theorem 3 are not the standard gradient squared norm but the scaled version. It is unclear what is the bound if the LHS is the standard gradient squared norm as in (Lian et al. 2017). It is important to use the same measure as in the previous literature for fair comparison.

The experiment is weak. Doing distributed training only on a tiny dataset on MNIST is not sufficient. I would like to see results on larger datasets such as CIFAR and ImageNet.

Our Reply.

We thank the reviewer for the comments/remarks on our paper. We wish to provide as many clarifications as needed to remove any doubts.

1. Linear speedup is not proved, the benefits of adaptive gradient methods are not shown in theory.

Regarding the linear speedup as in Lian et.al, we will add in the rebuttal version, a similar bound for our decentralized scheme. Thank you for the suggestion. While the advantage of the adaptive gradient methods does not show in our bound, we must stress that this latter bound showcases similar complexity in terms of number of iterations as centralized adaptive methods. Thus, this unclear advantage is inherited by current bounds in the centralized paradigm. Although we believe that theoretically showing that adaptive methods present a clear edge over non adaptive methods is an interesting and important research problem, we emphasize on the contribution of our paper being on the derivation of a general framework to convert any adaptive gradient optimization methods into its decentralized counterpart. The theoretical results we present aim at statistically prove convergence of this framework.

2. The learning rate less than $\epsilon^{0.5}/16L$ is not practical.

The theory in our paper is more focuses on showing the convergence guarantee in terms of T since even such basic guarantee is not guaranteed by DADAM. The assumption that step size is smaller than $\epsilon^{0.5}/16L$ can be relaxed by adding another $O(\frac{1}{\sqrt{T}})$ error term on RHS of (2). We will add a comment on this stepsize requirement below Theorem 2.

3. Relate LHS of Theorem 2 and 3 to standard gradient squared norm.

The LHS of Theorem 2 and 3 can be translated into standard gradient bound by using worst case L_∞ upper on $\overline{U}_t$, which depends on $\hat{v}_{t,i}$. This can not be specified in Theorem 2 since it is a framework and $\hat{v}_{t,i}$ is not determined. For Theorem 3, we can have $\|\overline{U}_t\|_\infty \leq G_\infty^2$ (due to update rules for $\hat{v}_{t,i}$) and thus $\|\frac{\nabla f(\overline{X}_t)}{\overline{U}_t^{1/4}}\|^2 \geq \frac{1}{G_\infty}\|\nabla f(\overline{X}_t)\|^2$. Then this measure can be translated into standard gradient norm bound. We will add discussion on this below Theorem 3. We want to friendly remind the reviewer that the main focus of this work is not proving matching bounds in (Lian et al. 2017) but to provide a framework for designing convergent adaptive gradient methods while DADAM diverges.

4. The experiment is weak.

We agree that experiments on larger datasets will be a good improvement to our paper, we are adding more experiments in the rebuttal. Yet, our current experiment shows a clear advantage of decentralized AMSGrad over DADAM on heterogeneous dataset. In Figure 1 (b), DADAM performs really bad in both training and testing compared with DGD, caused by its possible divergence. In contrast, decentralized AMSGrad performs even better than D-PSGD, showing that fixing the divergence issue of DADAM is of significance and supporting the importance of designing theoretically convergent algorithms such as the proposed framework.

2 REVIEWER 2

Cons:

In section 3.2, the paper claims AdaGrad and AMSGrad satisfy the condition to guarantee the convergence of Algorithm 2 while Adam does not. It seems not to be an obvious conclusion from the reference Chen et al. (2019). Is there more explanation or proof about why AdaGrad and AMSGrad satisfy the condition? And does it mean Adam still diverges even after using the algorithmic approach proposed in this paper? In Theorem 2, the convergence analysis result of Algorithm 2 is given. However, the convergence of common adaptive gradient methods such as AdaGrad and Adam is still not clear. Therefore, the 'convergent adaptive gradient method' in the title is very misleading. Do most of the adaptive gradient methods have the same theoretical guarantees as AMSGrad? $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}t - 2 + \hat{V}t - 1)\|_{abs}] = o(T)$ is the key condition to ensure the convergence. But when the above equation is $O(\sqrt{T})$, the convergence rate is worse than the centralized counterpart. Is that case possible? In section 3.4, the experiment is divided into homogeneous and heterogeneous data, which is very confusing. What is the reason for doing this and what will happen if we just deal with the dataset normally? The heterogeneous data is treated very intentionally. Is there any discussion about when the treatment of heterogeneous data is important? In the homogeneous data experiment, the performance of DADAM and decentralized AMSGrad are similar. What is the reason that the learning rates on different node tend to be similar? Is that a common case? Maybe the experiment on more dataset is needed to address this concern. Besides, how will such similarity among data impact the theoretical convergence?

Our Reply.

1. More explanation or proof about why AdaGrad and AMSGrad satisfy the condition? Does Adam still diverge after the algorithmic approach?

AdaGrad and AMSGrad can satisfy the condition essentially due to the stability in adaptive learning rate in their update rules. The sequence of adaptive learning rate in vanilla AMSGrad is non-decreasing, thus the overall oscillation of adaptive learning across T iterations is actually bounded by a constant. For AdaGrad, its adaptive learning rate can be viewed as the average of all past squared gradients (coordinate-wise). Due to the average, as t grows, the difference between adjacent adaptive learning rate is getting smaller. This ensures that the overall oscillation of adaptive learning rate in AdaGrad is bounded by $O(\log(T))$.

For Adam, it will still diverge even after the algorithmic approach. This is because the original Adam is divergent, our algorithmic approach can only convert convergent adaptive algorithms to their decentralized counterparts. However, Adam is not a convergent algorithm as proven in [Reddi et al., 2018]. As indicated by Theorem 2, the last term (oscillation of adaptive learning rate) in (2) should be o(T) to ensure convergence. However, the oscillation of adaptive learning rate is $O(T)$ for vanilla Adam in worst case, which is unbounded.

2. Convergence of common adaptive gradient methods such as AdaGrad and Adam is not clear in Theorem 2, thus the title is misleading. Do most of the adaptive gradient methods have the same theoretical guarantees as AMSGrad?

As noticed by the reviewer, Theorem 2 highlights a condition to ensure convergence of adaptive gradient methods converted by our framework, i.e. $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}] = o(T)$. Such a condition also appears in [Chen et al., 2019] and is satisfied by AMSGrad and AdaGrad, but not Adam. Thus, AdaGrad will converge but Adam will not. We will add more detailed discussion on this condition for AMSGrad, AdaGrad, and Adam below Theorem 2.

3. When $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}]$ is larger than $O(\sqrt{T})$, the convergence rate is worse than the centralized counterpart.

In short, such a situation is possible, yet, to the best of our knowledge, the term $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}]$ for many convergent adaptive algorithms is smaller than $O(\sqrt{T})$. Specifically, it is proven in literature that both AdaGrad and AMSGrad satisfy $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}] = o(\sqrt{T})$. Also, the convergence rate of centralized counterparts for these algorithms also depends on this term. Thus, our bound will not be worse than the centralized counterparts for these algorithms. We will add a comment on this below Theorem 2.

However, we agree with the reviewer that it is possible to design convergent algorithms with $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}]$ being large or unbounded, even $O(T)$. For example, when the adaptive learning rate is a positive random number generated independently at each iteration, we can have $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}] = O(T)$. Yet, because the adaptive learning rate does not depend on gradients, the convergence bias caused by correlation between adaptive learning rate and stochastic gradient is 0 and the algorithm can converge.

4. Reason for dividing homogeneous data and heterogeneous data.

We consider these two settings separately because both of them can happen in real world applications and the performance of different algorithms could be significantly different on these settings. Following are real world examples of homogeneous and heterogeneous data distributions.

1). Homogeneous data: The dataset is shuffled and uniformly randomly distributed on different nodes. Or all the nodes have access to the same dataset. These settings are common for computer clusters.

2). Heterogeneous data: Each node will collect their own dataset, the generating data distributions of different nodes are different. A data shuffling is prohibited because the nodes are reluctant to share data. Such settings are common in a cooperative training situations, one notable setting is federated learning.

5. Why DADAM and decentralized AMSGrad are similar in the homogeneous data experiment? Is that a common case? How will similarity among data impact the theoretical convergence?

These two algorithms are similar because the difference between DADAM and decentralized AMSGrad is the adaptive learning rate. DADAM uses individual adaptive learning rate on different nodes while decentralized AMSGrad uses approximate average of adaptive learning rate in DADAM as the true adaptive learning rate. The individual adaptive learning rates will be similar to the average of them since for homogeneous data distribution, the gradient distribution on different nodes are similar, so are the adaptive learning rates. The convergence bias of DADAM is caused by the difference of adaptive learning rates on different nodes and tends to be small for homogeneous data. The impact of distribution similarity on different nodes on the convergence of DADAM is a very interesting research question but we found a quantitative analysis on it to be non-trivial. We will leave this for future work.

3 Reviewer 3

However, I have some minor comments to improve the manuscript.

The experimental evaluation of the work is quite limited. I understand the space limit but it would have been nice to see more experiments instead of showcasing Algorithm 2 with an extra example in Algorithm 3. It is important to see the convergence behavior of the method (on the training data) with respect to the DGD on various datasets/networks in practice, rather than observing how the testing accuracy behaves. Note that your method does not guarantee any specific generalization behavior and therefore I believe it is more suited to report the experiments only in terms of training performance when you are out of space.

What are the drawbacks of this method? I can see more memory requirements for the agents due to the new variable $\tilde{u}$ for instance. Do you have any quantified evaluation in this respect? I suspect it can be significant especially if the trained model is large and the agents have limited memory/computational resources

In Section 3.2, the author says "Algorithm 2 can become different adaptive gradient methods by specifying r_t as different functions. E.g., when we choose ..., Algorithm 2 becomes a decentralized version of AdaGrad." This sentence is not accurate as algorithms like AdaGrad and Adadelta do not use momentum on the past gradients. They only use the squared values of the past gradients. I believe your method, as I mentioned above, is a general framework for momentum-based techniques including ADAM, AdaMax, NADAM, etc, which brings me to the next question.

Is it possible to generalized your method for an adaptive gradient descent algorithm that does not use the momentum of the gradients? For example, take AdaGrad with a fixed learning rate of η instead of m_t. How does your convergence behavior change?

Our Reply.

We thank you for the valuable comments on our submission.

1. Numerical experiments plots

We agree with the reviewer that experiments on various datasets/networks will be a good improvement to our paper, we will try to add more experiments in a future version. What we want to highlight in the experiments is that in Figure 1 (b), DADAM performs bad in terms of both training and testing performance, while decentralized AMSGrad performs as good as D-PSGD (even better). This shows the divergence issue for DADAM can have significant impact on model performance in practice instead of being a negligible defect, supporting the importance of designing theoretically convergent algorithms.

2. What are the drawbacks of this method?

As noticed by the reviewer, one drawback of our framework is the additional memory requirement for $\tilde{u}_{t,i}$. In our framework, across iterations, each node have to save a few quantities including $m_{t,i}$, $x_{t,i}$, memories for $\hat{v}_{t,i}$, and $\tilde{u}_{t,i}$. Take decentralized AMSGrad (Algorithm 3) for an example, we need d floats for $m_{t,i}$, $2d$ floats for $\hat{v}_{t,i}$ (because both $v_{t,i}$ in line 6 and $\hat{v}_{t,i}$ in line 7 of Algorithm 3 requires d floats), d floats for $x_{t,i}$, and additional d floats for $\tilde{u}_{t,i}$. Overall, we need to store $5d$ floats while centralized AMSGrad requires storing $4d$ floats (without $\tilde{u}_{t,i}$). Thus, decentralized AMSGrad requires 25% more storage compared with centralized AMSGrad which is acceptable in certain situations. However, compared with only d floats storage requirement in D-PSGD, decentralized AMSGrad indeed introduced significantly more memory. We also want to remark that the memory requirement can be smaller if the memory for adaptive learning rate $\hat{v}_{t,i}$ is smaller for other possible adaptive gradient methods. Suppose k instead of d floats are used for $\hat{v}_{t,i}$ (e.g. per-layer adaptive learning rate, k being the number of layers). One may also use k floats for $\tilde{u}_{t,i}$, leading to a memory requirement of $2d + 2k$ floats.

Another drawback of our method is the doubled amount of communication compared with DADAM and D-PSGD since we need consensus operation for $\tilde{u}_{t,i}$ in addition to $x_{t,i}$.

3. Is it possible to generalize the method to adaptive gradient methods without momentum such as AdaGrad?

It is possible to analyze the convergence of adaptive gradient without momentum using our framework. Actually a decentralized version of AdaGrad falls into our framework by setting $\hat{v}_{t,i} = \frac{1}{t}\sum_{k=1}^{t} g_{k,i}^2$ and $\beta_1 = 0$. Then Theorem 2 applies and then we can focus on bounding the term $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}]$ in (2) for this algorithm. Finally, some algebraic manipulations can show $\mathbb{E}[\sum_{t=1}^{T} \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs}] = O(Nd\log(T))$ for this algorithm, which is worse than decentralized AMSGrad by a factor of $O(\log(T))$ on this term. Substituting this bound into (2) we can see that this decentralized version of AdaGrad also converges with a rate of $O(\frac{\sqrt{d}}{\sqrt{T}})$

4 REVIEWER 4

This paper discusses the problem of adaptive acceleration in the decentralized setting. The premise is that decentralized versions, while successful in the simple SGD setting, do not extend well in the acceleration settings, e.g. Adam and Adagrad. They first present a counterexample where a simple decentralized scheme, applying Adam, converges to a nonstationary point. (Suggestion: I would maybe add a cleaner, more flushed out version of this proof in the appendix, maybe with illustrations.)

Overall, everything the paper presented seemed reasonable. The motivation and counterexample in DADAM case are solid, and the following theorems seem to suggest gradient error norm at rate $O(1/\sqrt{T})$ which is reasonable in nonconvex optimization. The intuition in the adjusted merging scheme is also reasonable, and makes sense that it would work better than vanilla merging schemes. However, I did not have a chance to carefully check the proofs, which are clearly the main contribution of the paper.

One thing I would suggest is a more thorough set of numerical experiments. The two examples shown, in fact all the methods converge, and while the proposed method converges faster, it isn't really verifying the paper's main point, which is that the standard distributed methods diverge and the proposed method converges. Showing this on a standard machine learning task would improve motivation.

Our Reply.

We thank the reviewer for his/her interest in our paper. Below we address your concerns about our contribution.

While we acknowledge that the numerical experiments can be improved by adding runs on larger datasets, we emphasize that the main contribution of our paper is (1) the derivation of a general framework for converting adaptive methods into their decentralized variant and (2) to establish their theoretical convergence guarantees. We precise that existing decentralized variant of adaptive method, namely DADAM, does not converge in certain settings, hence the necessity to come up with convergent solutions. The intermediary divergent example for DADAM is for illustrative purpose and is not the main argument of this paper. Meanwhile, the proposal convergent algorithms, their non asymptotic analysis and how they behave in numerical runs are what we want to push forward.

5 Reviewer 5

This paper consider the decentralized adaptive algorithms. At the first glance, I am really happy to that the adaptive methods are used for the decentralized optimization. However, after I read the main document, I do not think paper actually analyzes the decentralized adaptive algorithms.

In line 9 of Algorithm 1, the denominator is $\sqrt{\hat{v}_{t,i}}$

However, in Algorithms 2 and 3, it is changed as $\sqrt{u_{t,i}}$. In the proofs, the authors proved the convergence based on $u_{t,i} \geq \epsilon$. This is actually the DSGD. The proofs can be quite simple. And the restriction $\alpha = O(\sqrt{\epsilon})$ can be easily removed. This paper does not present any insights for the decentralized adaptive methods. It only depends on $u_{t,i}$. The numerical results show that the proposed method is similar as DSGD. As mentioned before, it is actually DSGD but with slightly modification.

Our Reply.

We thank the reviewer for the comments on our contribution.

However, we respectfully disagree with the remark regarding the fact that our analysis covers "a slightly modified DSGD".

Essentially, Algorithm 2 is not DSGD but rather a decentralized framework of some adaptive gradient methods (depending on the choice of the function r_t in line 6). The change of notation from $\hat{v}_{t,i}$ to $u_{t,i}$ is simply because in our decentralized version of adaptive methods, unlike DADAM of Nazari et. al., the algorithm (Alg. 2) uses the average consensus of $\hat{v}_{t,i}$ to help convergence (recall that DADAM fails to converge in some settings). Hence, $\hat{v}_{t,i}$ is not simply "changed to" $u_{t,i}$ but converted into an auxiliary quantity noted $u_{t,i}$ that is the result of several operations, see line 8 and 11, on the original $\hat{v}_{t,i}$.

Our analysis and algorithms are indeed with respect to decentralized variants of adaptive methods. We will change the notation of $u_{t,i}$ to avoid any confusion.