Fast Training Methods for Stochastic Compositional Optimization Problems

Anonymous Author(s)
Affiliation
Address
email

Abstract

The stochastic compositional optimization problem covers a wide range of machine learning models, such as sparse additive models and model-agnostic meta-learning. Thus, it is necessary to develop efficient methods for its optimization. Existing works for the stochastic compositional optimization problem only focus on the single machine scenario, which is far from satisfactory when data are distributed on different devices. To address this problem, we proposed novel decentralized stochastic compositional gradient descent methods to efficiently train the large-scale stochastic compositional optimization problem. To the best of our knowledge, our work is the first one facilitating decentralized training for this kind of problem. Furthermore, we provide the convergence analysis for our methods, which shows that the convergence rate of our methods can achieve linear speedup with respect to the number of devices. At last, we apply our decentralized training methods to the model-agnostic meta-learn task, and the experimental results demonstrated the superior performance of our methods.

1 Introduction

The stochastic compositional optimization problem [14] plays an important role in the machine learning field, since it covers numerous applications, such as policy evaluation [13], sparse additive models [14], and model-agnostic meta-learning [5, 1]. Specifically, the stochastic compositional optimization problem is defined as follows:

$$\min_{x\in\mathbb{R}^d} \mathbb{E}_\zeta\left[f\left(\mathbb{E}_\xi[g(x;\xi)];\zeta\right)\right], \tag{1}$$

where the outer-level function $f:\mathbb{R}^{d'}\to\mathbb{R}$ is a smooth and nonconvex function, the inner-level function $g:\mathbb{R}^d\to\mathbb{R}^{d'}$ is a smooth function. It is obvious that the loss function consists of two stochastic functions f and g, making it different from the non-compositional problem.

The compositional structure in Eq. (1) leads to more challenges in optimization compared with the non-compositional problem, since the stochastic gradient of the loss function in Eq. (1) is not an unbiased estimation of the full gradient. Thus, efficient training of the stochastic compositional optimization problem has attracted increasing attention in recent years. For example, [14] proposed the stochastic compositional gradient descent (SCGD) method to deal with the two-level stochasticity in Eq. (1). But it has a worse convergence rate than the standard stochastic gradient descent method. To improve it, a series of variance-reduced methods have been proposed. For example, [19] developed the composite randomized incremental gradient method based on the SAGA [3] technique. [21, 18] further improved the convergence rate based on the SPIDER [4] method.

Although aforementioned methods have achieved much progress in optimizing Eq. (1), all of them only focus on the single-machine scenario. In fact, data are usually distributed on different devices

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

in many real-world applications. It is necessary to study the distributed training method for the stochastic compositional optimization problem. In the past few years, numerous distributed training methods have been proposed, such as asynchronous SGD [10], decentralized SGD [7], local SGD [8]. But they are not applicable for the stochastic compositional problem. Especially, their convergence results do not hold for Eq. (1) because those distributed training methods require the stochastic gradient is an unbiased estimation of the full gradient. As far as we know, there are no existing works studying distributed training methods for Eq. (1) with a theoretical guarantee.

To fill the aforementioned gap, we developed two novel decentralized training methods for the stochastic compositional optimization problem. In particular, we proposed a *gossip-based* decentralized stochastic compositional gradient descent (GP-SCGD) method and a *gradient-tracking-based* decentralized stochastic compositional gradient descent (GT-SCGD) method. With our methods, different devices can collaboratively optimize Eq. (1). However, establishing the convergence rate of our proposed methods is challenging. Existing convergence analysis techniques for decentralized SGD methods are not applicable because the *stochastic compositional gradient* is different from the *stochastic gradient*. To address this challenge, we proposed new convergence analysis techniques for the decentralized SCGD method, where we show how to bound the gradient variance and the consensus error under the decentralized setting. In particular, both GP-SCGD and GT-SCGD can achieve the convergence rate $O(\frac{1}{K\epsilon^2})$ to achieve ϵ-accuracy solution, where K is the number of devices. It indicates that our methods can achieve the linear speedup with respect to the number of devices, which is consistent with decentralized SGD method. To the best of our knowledge, our work is the first one to establish the convergence rate of the decentralized SCGD and achieving the linear speedup. Finally, we apply our proposed methods to the model-agnostic meta-learning problem. The empirical results confirm the effectiveness of our methods. In the following, we summarize the contributions of our work.

- We proposed two novel decentralized stochastic compositional gradient descent methods: GP-SCGD and GT-SCGD. This is the first work studying the decentralized training of stochastic compositional optimization problems.
- We studied the convergence rate of our decentralized stochastic compositional gradient descent methods with novel theoretical analysis. This is the first work establishing the convergence rate $O(\frac{1}{K\epsilon^2})$ for decentralized SCGD methods and showing the linear speedup with respect to the number of devices.
- The extensive empirical evaluation on the model-agnostic meta-learning task confirms the effectiveness of our proposed methods.

2 Related Works

2.1 Stochastic Compositional Optimization Problem

The stochastic compositional optimization problem is common in the machine learning field. However, it is more difficult to optimize than the standard non-compositional problem. The reason is that its stochastic gradient is not an unbiased estimation of the full gradient, which is shown as follows:

$$\mathbb{E}_{\xi,\zeta}[\nabla g(x;\xi)^T \nabla_g f(g(x;\xi);\zeta)] \neq \nabla g(x)^T \nabla_g f(g(x)) \ . \tag{2}$$

To train the stochastic compositional problem efficiently, numerous methods have been proposed in the past few years. Specifically, [14] developed the stochastic compositional gradient descent (SCGD) method, which is defined as follows:

$$\begin{aligned} u_t &= (1-\beta_{t-1})u_{t-1} + \beta_{t-1} g(x_t; B_{\xi,t}) \ , \\ x_{t+1} &= x_t - \eta_t \nabla g(x_t; B_{\xi,t})^T \nabla_g f(u_t; A_{\zeta,t}) \ , \end{aligned} \tag{3}$$

where $B_{\xi,t}$ and $A_{\zeta,t}$ denote the random samples, $0 < \beta_t < 1$ is a hyperparameter, and η_t is the learning rate. Here, u_t serves as the estimator of the inner function $g(x_t)$, and then the gradient $\nabla_g f(g(x_t))$ is estimated by $\nabla_g f(u_t)$. As a result, this stochastic compositional gradient $\nabla g(x_t; B_{\xi,t})^T \nabla_g f(u_t; A_{\zeta,t})$ has a smaller variance so that it can accelerate the convergence speed. However, it still has a worse theoretical convergence rate than SGD. Furthermore, to improve the convergence speed of SCGD for nonconvex problems, the advanced variance reduction techniques have been incorporated into SCGD. For example, [19] incorporated the variance reduction technique

SAGA [9] to SCGD and obtained a better convergence rate. Later, [21] employed another variance reduction technique SPIDER [4] for SCGD, while [17] accelerated SCGD with the STORM [2] variance reduction technique. However, all of these methods only studied the convergence rate for the single machine case, ignoring the distributed setting.

2.2 Decentralized Training Methods

Recently, with the emergence of large-scale data, numerous distributed training methods [10, 7, 8, 12, 6] have been proposed. Among of them, the decentralized training method, where different devices conduct the peer-to-peer communication, has attracted a lot of attention. In particular, there are two kinds of communication strategies in a decentralized training system. They are the *gossip* strategy and the *gradient tracking* strategy. Based on the gossip strategy, [7] studied the convergence rate of the decentralized SGD method and disclosed how the topology of the communication graph affects the convergence rate. Additionally, numerous works aim to reduce the communication cost of the gossip-based decentralized SGD by compressing gradients or skipping the communication step. Meanwhile, based on the gradient-tracking strategy, [11] developed the distributed stochastic gradient tracking method and established its convergence rate. Recently, some works [12, 16] combined the variance reduction strategy and the gradient-tracking strategy to further accelerate the decentralized SGD method. However, all of these methods are not applicable for SCGD method. Thus, in this paper, we will develop efficient decentralized SCGD methods for optimizing the stochastic compositional optimization problem.

3 Decentralized Stochastic Compositional Optimization Methods

3.1 Problem Definition

Formally, we consider the distributed stochastic compositional optimization problem, which is defined as follows:

$$\min_{x\in\mathbb{R}^d} \frac{1}{K}\sum_{k=1}^{K} \mathbb{E}_{\zeta}\left[f^{(k)}\left(\mathbb{E}_{\xi}[g^{(k)}(x;\xi)];\zeta\right)\right], \tag{4}$$

where $x \in \mathbb{R}^d$ denotes the model parameter, K is the total number of devices, $f^{(k)} : \mathbb{R}^{d'} \to \mathbb{R}$ and $g^{(k)} : \mathbb{R}^d \to \mathbb{R}^{d'}$ are smooth functions on the k-th device, and accordingly $\mathbb{E}_{\zeta}\left[f^{(k)}\left(\mathbb{E}_{\xi}[g^{(k)}(x;\xi)];\zeta\right)\right]$ is the loss function on the k-th device. In other words, each device has its own data and all devices collaboratively learn the model parameter x by utilizing their local data.

To efficiently train this distributed stochastic compositional optimization problem, we consider the decentralized training method. In particular, each device connects with its neighbors, composing a communication network. All devices conduct the peer-to-peer communication based on this communication network. Typically, we represent the communication network with $\mathcal{G} = (V, W)$ where $V = \{v_1, v_2, \cdots, v_K\}$ denotes all devices and $W = [w_{ij}] \in \mathbb{R}^{K\times K}$ is the adjacency matrix which indicates whether two devices are connected or not. Following [6, 7], the adjacency matrix W is assumed to satisfy the following assumption.

Assumption 1. *The adjacency matrix W has the following properties:*

- *Nonnegative:* $w_{ij} \geq 0$, $\forall i, j$.
- *Symmetric:* $w_{ij} = w_{ji}$, $\forall i, j$.
- *Doubly Stochastic:* $\sum_{i=1}^{K} w_{ij} = 1$, $\sum_{j=1}^{K} w_{ij} = 1$.

Assumption 1 indicates that the eigenvalues of the adjacency matrix can be sorted as $|\lambda_n| \leq \cdots \leq |\lambda_2| < |\lambda_1| = 1$. In our convergence analysis, we denote $\lambda = |\lambda_2|$ for simplicity. In light of the aforementioned decentralized training setting, we proposed two descentralized stochastic compositional gradient descent methods. The details are described in the following subsection.

3.2 Gossip-based Decentralized Stochastic Compositional Gradient Descent Method

In Algorithm 1, we developed the *gossip*-based descentralized stochastic compositional gradient descent (GP-DSCGD) method. Following the standard SCGD method [14], each device estimates the inner-level function $g^{(k)}(x_t^{(k)})$ with $u_t^{(k)}$ as follows:

$$u_t^{(k)} = (1-\gamma\beta_{t-1})u_{t-1}^{(k)} + \gamma\beta_{t-1}g^{(k)}(x_t^{(k)};\mathcal{B}_t^{(k)}) \ , \tag{5}$$

where $\gamma > 0$ and $\beta_t > 0$ are two hyperparameters, $\gamma\beta_t < 1$, k is the index of devices and t is the index of iterations. Different from the standard SCGD method in Eq. (3), there is an additional hyperparameter γ when computing u_t, which can help controlling the estimation variance. We will show it in our theoretical analysis in Appendix. Based on this estimation, each device computes the stochastic compositional gradient $z_t^{(k)}$ as shown in Line 9. Then, each worker uses the gossip communication strategy to update the local model parameter as follows:

$$\tilde{x}_{t+1}^{(k)} = \sum_{j\in\mathcal{N}_{v_k}} w_{kj}x_t^{(j)} - \eta z_t^{(k)} \ , \tag{6}$$

where $\eta > 0$ denotes the learning rate, $\mathcal{N}_{v_k} = \{j|w_{kj} > 0\}$ is the neighboring devices of the k-th device. Here, the first term on the right-hand side incurs the communication to average the model parameter of neighboring devices, and the second term indicates to update the model parameter with the local gradient $z_t^{(k)}$. Then, instead of using $\tilde{x}_{t+1}^{(k)}$ as the new model parameter, we compute the model parameter of the $t+1$-th iteration as follows:

$$x_{t+1}^{(k)} = x_t^{(k)} + \beta_t(\tilde{x}_{t+1}^{(k)} - x_t^{(k)}). \tag{7}$$

In fact, this strategy couples the update of $u_t^{(k)}$ and $x_t^{(k)}$ by the sharing hyperparameter β_t, which can benefit controlling the estimation variance of $u_t^{(k)}$. All devices repeat the aforementioned steps until the iterates converge.

Algorithm 1 Gossip-based Decentralized Stochastic Compositional Gradient Descent (GP-DSCGD)

Input: $x_0^{(k)} = x_0$, $\beta_t > 0$, $\gamma > 0$, $\eta > 0$.

1: **for** $t = 0, \cdots, T-1$, each device k **do**
2: Sample a subset of samples $\mathcal{B}_t^{(k)}$ to compute:
3: **if** $t = 0$ **then**
4: $u_t^{(k)} = g^{(k)}(x_t^{(k)};\mathcal{B}_t^{(k)})$
5: **else**
6: $u_t^{(k)} = (1-\gamma\beta_{t-1})u_{t-1}^{(k)} + \gamma\beta_{t-1}g^{(k)}(x_t^{(k)};\mathcal{B}_t^{(k)})$
7: **end if**
8: $v_t^{(k)} = \nabla g^{(k)}(x_t^{(k)};\mathcal{B}_t^{(k)})$
9: Sample a subset of samples $\mathcal{A}_t^{(k)}$ to compute:
$z_t^{(k)} = (v_t^{(k)})^T\nabla f^{(k)}(u_t^{(k)};\mathcal{A}_t^{(k)})$
10: $\tilde{x}_{t+1}^{(k)} = \sum_{j\in\mathcal{N}_{v_k}} w_{kj}x_t^{(j)} - \eta z_t^{(k)}$
11: $x_{t+1}^{(k)} = x_t^{(k)} + \beta_t(\tilde{x}_{t+1}^{(k)} - x_t^{(k)})$
12: **end for**

3.3 Gradient-Tracking-based Decentralized Stochastic Compositional Gradient Descent Method

Besides the gossip-based communication strategy, the gradient tracking strategy is also widely used in the decentralized training method. Therefore, for completeness, we further proposed the gradient-tracking-based descentralized stochastic compositional gradient descent (GT-DSCGD) method in Algorithm 2. Same with Algorithm 1, we compute $u_t^{(k)}$ and $z_t^{(k)}$ as shown in Lines 2-9. Different from Algorithm 1, GT-DSCGD introduces an auxiliary variable $s_t^{(k)}$ for each device, which is used to track the global stochastic compositional gradient $\frac{1}{K}\sum_{k=1}^K z_t^{(k)}$ [12, 11]. Specifically, $s_t^{(k)}$ is

updated as follows:

$$s_t^{(k)} = \sum_{j \in \mathcal{N}_{v_k}} w_{kj} s_{t-1}^{(j)} + z_t^{(k)} - z_{t-1}^{(k)} \ . \tag{8}$$

It is easy to verify that $\frac{1}{K}\sum_{k=1}^K s_t^{(k)} = \frac{1}{K}\sum_{k=1}^K z_t^{(k)}$. Then, each device uses $s_t^{(k)}$ instead of $z_t^{(k)}$ to update the model parameter as follows:

$$\tilde{x}_{t+1}^{(k)} = \sum_{j \in \mathcal{N}_{v_k}} w_{kj} x_t^{(j)} - \eta s_t^{(k)} \ . \tag{9}$$

To sum up, the difference between our two methods is that GT-DSCGD has an additional step to track the gradient with $s_t^{(k)}$ and accordingly updates the model parameter with $s_t^{(k)}$.

Algorithm 2 Gradient-Tracking-based Decentralized Stochastic Compositional Gradient Descent (GT-DSCGD)

Input: $x_0^{(k)} = x_0$, $z_{-1}^{(k)} = 0$, $s_{-1}^{(k)} = 0$, $\beta_t > 0$, $\gamma > 0$, $\eta > 0$.

for $t = 0, \cdots, T-1$, each device k **do**
  Sample a subset of samples $\mathcal{B}_t^{(k)}$ to compute:
  if $t = 0$ **then**
    $u_t^{(k)} = g^{(k)}(x_t^{(k)}; \mathcal{B}_t^{(k)})$
  else
    $u_t^{(k)} = (1-\gamma\beta_{t-1})u_{t-1}^{(k)} + \gamma\beta_{t-1} g^{(k)}(x_t^{(k)}; \mathcal{B}_t^{(k)})$
  end if
  $v_t^{(k)} = \nabla g^{(k)}(x_t^{(k)}; \mathcal{B}_t^{(k)})$
  Sample a subset of samples $\mathcal{A}_t^{(k)}$ to compute:
  $z_t^{(k)} = (v_t^{(k)})^T \nabla f^{(k)}(u_t^{(k)}; \mathcal{A}_t^{(k)})$
  $s_t^{(k)} = \sum_{j \in \mathcal{N}_{v_k}} w_{kj} s_{t-1}^{(j)} + z_t^{(k)} - z_{t-1}^{(k)}$ // gradient tracking
  $\tilde{x}_{t+1}^{(k)} = \sum_{j \in \mathcal{N}_k} w_{kj} x_t^{(j)} - \eta s_t^{(k)}$
  $x_{t+1}^{(k)} = x_t^{(k)} + \beta_t(\tilde{x}_{t+1}^{(k)} - x_t^{(k)})$
end for

4 Convergence Analysis

4.1 Assumption

To establish the convergence rates of our proposed two methods, we first introduce some standard assumptions, which are also used in existing stochastic compositional gradient descent methods [14, 15, 20, 18].

Assumption 2. *(Smoothness) For $\forall y_1, y_2 \in \mathbb{R}^{d'}$ and $\forall x_1, x_2 \in \mathbb{R}^d$, there exist two constant values $L_f > 0$ and $L_g > 0$ such that*

$$\|\nabla f^{(k)}(y_1) - \nabla f^{(k)}(y_2)\| \le L_f \|y_1 - y_2\|, \ \|\nabla g^{(k)}(x_1) - \nabla g^{(k)}(x_2)\| \le L_g \|x_1 - x_2\| \ . \tag{10}$$

Assumption 3. *(Bounded gradient) For $\forall x \in \mathbb{R}^d$ and $\forall y \in \mathbb{R}^{d'}$, there exist two constant values $C_g > 0$ and $C_f > 0$ such that*

$$\mathbb{E}[\|\nabla g^{(k)}(x;\xi)\|^2] \le C_g^2, \ \mathbb{E}[\|\nabla f^{(k)}(y;\zeta)\|^2] \le C_f^2 \ . \tag{11}$$

Assumption 4. *(Bounded variance) For $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^{d'}$, there exist three constant values $\sigma_f > 0$, $\sigma_g > 0$, $\sigma_{g'} > 0$, such that*

$$\begin{gathered}\mathbb{E}[\|\nabla f^{(k)}(y;\zeta) - \nabla f^{(k)}(y)\|^2] \le \sigma_f^2, \ \mathbb{E}[\|\nabla g^{(k)}(x;\xi) - \nabla g^{(k)}(x)\|^2] \le \sigma_{g'}^2, \\ \mathbb{E}[\|g^{(k)}(x;\xi) - g^{(k)}(x)\|^2] \le \sigma_g^2 \ .\end{gathered} \tag{12}$$

To study the convergence rates of our two methods, we denote $F(x) = \frac{1}{K}\sum_{k=1}^K F^{(k)}(x) = \frac{1}{K}\sum_{k=1}^K f^{(k)}(g^{(k)}(x))$. Then, $F(x)$ is L_F-smooth with $L_F = C_g^2 L_f + C_f L_g$. In addition, we denote $\bar{x}_t = \frac{1}{K}\sum_{k=1}^K x_t^{(k)}$ and x_* to be the optimal solution.

4.2 Convergence Rate

In light of the aforementioned definitions and assumptions, we establish convergence rates of our proposed Algorithm 1 and Algorithm 2, respectively.

Theorem 1. *Assume Assumptions 1-4 hold, for Algorithm 1, by setting* $|\mathcal{A}_t^k| = |\mathcal{B}_t^k| = B$, $\gamma > 0$, $\beta \leq \min\{\frac{1}{8\gamma}, \frac{1}{2\eta L_F}, 1\}$, $\eta \leq \min\{\eta_1, \eta_2, \eta_3\}$ *where*

$$\eta_1 = \frac{\gamma}{3\sqrt{6}C_g^2 L_f}, \eta_2 = \frac{1-\lambda^2}{2} \Big/ \Big(\frac{27C_g^4 L_f^2 + \gamma^2 L_F^2}{\gamma^2} + \frac{\gamma(24C_f^2 L_g^2 + 27C_g^4 L_f^2)}{2C_g^2 L_f^2} \Big),$$
$$\eta_3 = \frac{\sqrt{b^2+4ac}-b}{2a}, a = \frac{(24C_f^2 L_g^2 + 3C_g^4 L_f^2)}{8(1-\lambda^2)C_g^2 L_f^2} + \frac{3C_g^2}{1-\lambda^2} + \frac{27C_g^4 L_f^2}{2\gamma^2}, b = \frac{2}{1-\lambda^2}, c = \frac{\gamma}{4C_g^2 L_f^2}, \tag{13}$$

we can get

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\bar{x}_t)\|^2] \leq \frac{2\mathbb{E}[F(\bar{x}_0) - F(x_*)]}{\eta\beta T} + \frac{2(3C_g^2\sigma_f^2 + 3C_f^2\sigma_{g'}^2 + C_g^2 L_f^2 \sigma_g^2 + 6\gamma\sigma_g^2)}{B} + \frac{48C_f^2\sigma_{g'}^2 + 16C_g^2\sigma_f^2 + 27C_g^2 L_f^2\sigma_g^2}{16\beta C_g^2 L_f^2 B} + \frac{6C_g^2 L_f^2 \sigma_g^2}{\beta\gamma B} + \frac{12\sigma^2}{\beta B}. \tag{14}$$

Corollary 1. *Assume Assumptions 1-4 hold, for Algorithm 1, by setting* $\eta = O(\frac{\sqrt{K}}{\sqrt{T}})$, $B = O(\sqrt{KT})$, β *and* γ *to be two positive constant values, we can get*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\bar{x}_t)\|^2] \leq \frac{2\mathbb{E}[F(\bar{x}_0) - F(x_*)]}{\beta\sqrt{KT}} + \frac{2(3C_g^2\sigma_f^2 + 3C_f^2\sigma_{g'}^2 + C_g^2 L_f^2 \sigma_g^2 + 6\gamma\sigma_g^2)}{\sqrt{KT}} + \frac{48C_f^2\sigma_{g'}^2 + 16C_g^2\sigma_f^2 + 27C_g^2 L_f^2\sigma_g^2}{16\beta C_g^2 L_f^2 \sqrt{KT}} + \frac{6C_g^2 L_f^2 \sigma_g^2}{\beta\gamma\sqrt{KT}} + \frac{12\sigma^2}{\beta\sqrt{KT}}. \tag{15}$$

Remark 1. *For sufficiently large* T, *Corollary 1 indicates that GP-DSCGD can achieve linear speedup with respect to the number of devices, which is consistent with the decentralized SGD method. Meanwhile, to achieve the* ϵ*-accuracy solution such that* $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\bar{x}_t)\|^2] \leq \epsilon$, *the number of iterations and the communication complexity is* $O\Big(\frac{1}{K\epsilon^2}\Big)$.

Remark 2. *When* T *is sufficiently large,* $\frac{1}{2\eta L_F}$ *could be greater than* 1 *so that* β *and* γ *do not affect the order of the convergence rate in Corollary 1.*

Theorem 2. *Assume Assumptions 1-4 hold, for Algorithm 2, by setting* $|\mathcal{A}_t^k| = |\mathcal{B}_t^k| = B$, $\gamma > 0$, $\beta \leq \min\{\frac{1}{8\gamma}, \frac{1}{2\eta L_F}, 1\}$, *and* $\eta \leq \min\{\eta_1, \eta_2, \eta_3\}$, *where*

$$\eta_1 = \frac{4\gamma(1-\lambda^2)}{8\gamma^2 L_F^2 + 289C_g^4 L_f^2 + 16C_f^2 L_g^2}, \eta_2 = \frac{\gamma}{2\sqrt{19C_g^4 L_f^2 + C_f^2 L_g^2}},$$
$$\eta_3 = \frac{\sqrt{b^2+4ac}-b}{2a}, a = \frac{27C_g^4 L_f^2}{4\gamma^2} + \frac{3C_f^2 L_g^2}{8\gamma^2} + \frac{3C_g^4 L_f^2}{128\gamma^2}, b = \frac{6}{1-\lambda^2}, c = (1-\lambda)^2, \tag{16}$$

we can get

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\bar{x}_t)\|^2] \leq \frac{2\mathbb{E}[F(\bar{x}_0) - F(x_*)]}{\eta\beta T} + \frac{6C_g^2\sigma_f^2 + 6C_f^2\sigma_{g'}^2 + 3C_g^2 L_f^2\sigma_g^2}{B} + \frac{24\eta^2(C_f^2\sigma_{g'}^2 + C_g^2\sigma_f^2)(19C_g^4 L_f^2 + C_f^2 L_g^2)}{\gamma^2 B} + \frac{48\eta^2(C_f^2\sigma_{g'}^2 + C_g^2\sigma_f^2)(19C_g^4 L_f^2 + C_f^2 L_g^2)}{\gamma^2(1-\lambda^2)B} + \frac{48\eta(C_f^2\sigma_{g'}^2 + C_g^2\sigma_f^2)}{\gamma(1-\lambda^2)B} + \frac{96\eta(C_f^2\sigma_{g'}^2 + C_g^2\sigma_f^2)}{\gamma(1-\lambda^2)^2 B} + \frac{8C_g^2 L_f^2\sigma_g^2 + 96(1-\lambda)C_g^2 L_f^2\sigma^2}{\gamma\beta B}. \tag{17}$$

Corollary 2. *Assume Assumptions 1-4 hold, for Algorithm 2, by setting $\eta = O(\frac{\sqrt{K}}{\sqrt{T}})$, $B = O(\sqrt{KT})$, β and γ to be two positive constant values, we can get*

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\bar{x}_t)\|^2] \leq & \frac{2\mathbb{E}[F(\bar{x}_0)-F(x_*)]}{\beta\sqrt{KT}} + \frac{48(C_f^2\sigma_{g'}^2 + C_g^2\sigma_f^2)}{\gamma(1-\lambda^2)T} + \frac{96(C_f^2\sigma_{g'}^2 + C_g^2\sigma_f^2)}{\gamma(1-\lambda^2)^2 T} \\ & + \frac{24(C_f^2\sigma_{g'}^2 + C_g^2\sigma_f^2)(19C_g^4L_f^2 + C_f^2L_g^2)}{\gamma^2K^{1/2}T^{3/2}} + \frac{48(C_f^2\sigma_{g'}^2 + C_g^2\sigma_f^2)(19C_g^4L_f^2 + C_f^2L_g^2)}{\gamma^2(1-\lambda^2)K^{1/2}T^{3/2}} \\ & + \frac{6C_g^2\sigma_f^2 + 6C_f^2\sigma_{g'}^2 + 3C_g^2L_f^2\sigma_g^2}{\sqrt{KT}} + \frac{8C_g^2L_f^2\sigma_g^2 + 96(1-\lambda)C_g^2L_f^2\sigma^2}{\gamma\beta\sqrt{KT}} .\end{aligned} \tag{18}$$

Remark 3. *For sufficiently large T, Corollary 2 also indicates that GT-DSCGD can achieve linear speedup with respect to the number of devices, which is consistent with the decentralized SGD method. Similar to GP-DSCGD, to achieve the ϵ-accuracy solution such that $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\bar{x}_t)\|^2] \leq \epsilon$, the number of iterations and the communication complexity of GT-DSCGD is $O\left(\frac{1}{K\epsilon^2}\right)$.*

4.3 Main Proof

In this subsection, we present the main idea of establishing the convergence rate of Algorithm 1 and Algorithm 2. More details are deferred to Appendix.

Proof of Theorem 1 To establish the convergence rate of Algorithm 1, we proposed a novel potential function as follows:

$$\begin{aligned}P_t = & \mathbb{E}[F(\bar{x}_t)] + \frac{3\eta C_g^2L_f^2}{\gamma}\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|u_t^{(k)} - g^{(k)}(x_t^{(k)})\|^2] + \frac{\eta}{K}\sum_{k=1}^{K}\mathbb{E}[\|u_t^{(k)} - \bar{u}_t\|^2] \\ & + \frac{\gamma\eta\beta}{4C_g^2L_f^2}\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|z_t^{(k)} - \bar{z}_t\|^2] + \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|x_t^{(k)} - \bar{x}_t\|^2] .\end{aligned} \tag{19}$$

Then, the task boils down to bound each term in this potential function. In particular, we obtain these bounds in Lemmas 3, 4, 5, 6, respectively in Appendix. With these lemmas, by setting the hyperparameter as shown in Theorem 1, we can get

$$\begin{aligned}P_{t+1} - P_t \leq & -\frac{\eta\beta}{2}\mathbb{E}[\|\nabla F(\bar{x}_t)\|^2] + \frac{\eta\beta(3C_g^2\sigma_f^2 + 3C_f^2\sigma_{g'}^2 + C_g^2L_f^2\sigma_g^2 + 6\gamma\sigma_g^2)}{B} \\ & + \frac{24C_f^2\sigma_{g'}^2 + 8C_g^2\sigma_f^2 + 3C_g^2L_f^2\sigma_g^2}{B}\frac{\gamma\eta\beta}{4C_g^2L_f^2} .\end{aligned} \tag{20}$$

By summing t, we can complete the proof.

Proof of Theorem 2 Similar to the convergence analysis of Algorithm 1, we also proposed a novel potential function to facilitate the theoretical analysis. Here, different from the proof of Theorem 1, we cannot directly use $\mathbb{E}[\|s_t^{(k)} - \bar{s}_t\|^2]$ in the potential function, since in our preliminary analysis we found $\mathbb{E}[\|s_t^{(k)} - \bar{s}_t\|^2]$ introduces an additive term in the final convergence bound, which cannot be bounded. Thus, to address this problem, we introduce an additional variable $h_t^{(k)}$, which is defined as follows:

$$h_{-1}^{(k)} = 0 \, , h_t^{(k)} = \sum_{j\in\mathcal{N}_{v_k}} w_{kj}h_{t-1}^{(j)} + q_t^{(k)} - q_{t-1}^{(k)} \, , \tag{21}$$

where $q_t^{(k)} = \nabla g^{(k)}(x_t^{(k)})^T\nabla_g f^{(k)}(u_t^{(k)})$. Here, the difference between $h_t^{(k)}$ and $s_t^{(k)}$ is that $h_t^{(k)}$ uses the full gradient of each function while $s_t^{(k)}$ uses the stochastic gradient. Note that $h_t^{(k)}$ is just used for theoretical analysis. It does NOT need to be computed in practice. Based on this new

variable, we define the following potential function:

$$P_t = \mathbb{E}[F(\bar{x}_t)] + \frac{4\eta C_g^2 L_f^2}{\gamma}\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|u_t^{(k)} - g^{(k)}(x_t^{(k)})\|^2] + \frac{\eta(1-\lambda)}{\gamma}\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|h_t^{(k)} - \bar{h}_t\|^2] + \frac{1}{\gamma}\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|x_t^{(k)} - \bar{x}_t\|^2] \,. \tag{22}$$

Similar to the proof of Theorem 1, the task boils down to bound each term in the potential function, respectively. The detailed bounds can be found in Lemmas 9, 13, 11. Then, we can also get the bound for $P_{t+1} - P_t$ and then we can complete the proof by summing t over all iterations like the proof of Theorem 1.

5 Experiment

5.1 Model-Agnostic Meta-Learning

Model-Agnostic Meta-Learning (MAML) [5] is to learn a meta-initialization model parameter for a set of new tasks. Specifically, it is to optimize the following model:

$$\min_{x\in\mathbb{R}^d} \mathbb{E}_{i\sim p, D^i_{\text{test}}}\, l_i\left(\mathbb{E}_{D^i_{\text{train}}}\left(x - \alpha\nabla l_i\left(x, D^i_{\text{train}}\right)\right), D^i_{\text{test}}\right) \tag{23}$$

where p is the task distribution, α is the learning rate, l_i is the loss function for the i-th task, D^i_{train} is the training set of the i-th task and D^i_{test} is the testing set of the i-th task. Note that we omit the superscript k for simplication. MAML can be reformulated as a stochastic compositional optimization problem as follows:

$$\begin{aligned} f(y;\zeta) &= l_i\left(y, D^i_{\text{test}}\right), \text{ where } \zeta = \left(i, D^i_{\text{test}}\right) \\ g(x;\xi) &= x - \alpha\nabla l_i\left(x, D^i_{\text{train}}\right), \text{ where } \xi = D^i_{\text{train}} \,. \end{aligned} \tag{24}$$

Then, we can use the stochastic compostional gradient to optimize MAML. In particular, we can use u_t to track $g(x_t)$ as follows:

$$u_t = (1-\gamma\eta_{t-1})u_{t-1} + \gamma\eta_{t-1}g(x_t; B_{\xi,t}) = (1-\gamma\eta_{t-1})u_{t-1} + \gamma\eta_{t-1}(x_{t-1} - \alpha\nabla l_i\left(x_{t-1}, D^i_{\text{train}}\right)) \,. \tag{25}$$

In our experiment, we will use GP-DSCGD and GT-DSCGD to solve the distributed MAML problem, where tasks are distributed on different devices.

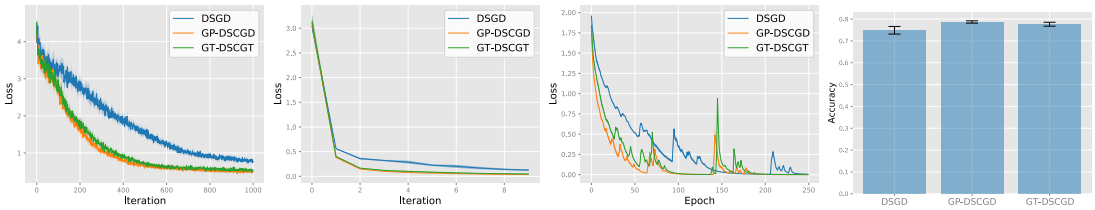
5.2 Regression Task

Following [5], we verify the performance of our methods with the regression problem. In particular, each task in MAML is a regression task, which maps the input to a sine wave. Different sine waves have different amplitudes and phases. Specifically, to generate a set of sine waves, the amplitude is uniformly drawn from $[0.1, 5.0]$ and the phase is uniformly drawn from $[0, \pi]$. Additionally, the input a is randomly drawn from $[-5.0, 5.0]$. Then, we use a parameterized neural network $f(x; a)$ to approximate the sine wave. In our experiment, $f(x; a)$ has two hidden layers and the dimensionality of different layers is $[1, 40, 40, 1]$. In each hidden layer, we use the ReLU function as the activation function. There are numerous regression tasks, the goal of MAML is to learn a good initialization for the model parameter x such that it can be quickly adapted to new regression tasks and have good performance.

In our experiment, we use four GPUs where each GPU is viewed as a device. These devices compose a ring graph. On each device, the number of tasks in each meta-batch is set to 200. The number of samples in each task for training is set to 10. As for the testing set, the number of tasks is set to 500 and the number of samples in each task is also set to 10. In addition, the number of iterations for adaptation in the training phase is set 1 while it is set to 10 in the testing phase. To verify the performance of our methods, we compare them with the gossip-based decentralized SGD (DSGD) method. Specifically, the decentralized SGD method directly uses $u_t = g(x_t)$. In other words, DSGD directly uses $x_{t-1} - \alpha\nabla l_i\left(x_{t-1}, D^i_{\text{train}}\right)$ for adaptation rather than that in Eq. (25). In our experiment,

α is set to 0.01. For adaptation, the learning rate η of DSGD is set to $\eta = 0.001$. Sine Adam has better performance in MAML [5], we actually use the decentralized Adam in our experiment as the baseline method. Accordingly, for fair comparison, we also use the adaptive learning rate for our two methods based on the stochastic compositional gradient $z_t^{(k)}$ and $s_t^{(k)}$. In particular, we set $\gamma = 3.0$, $\eta = 0.03$, $\beta = 0.33$. The reason for this setting is that $x_{t+1}^{(k)} \approx x_t^{(k)} - \beta\eta z_t^{(k)}$. Therefore, the effective learning rate $\beta\eta$ of our methods is approximately equal to that of the baseline method, which is a fair comparison.

In Figure 1(a), we show the average and standard deviation of the training loss across different devices. It can be seen that our two methods converge much faster than DSGD method. The reason is that our methods use u_t to track the $g(x_t)$ so that the estimation variance for $g(x_t)$ is smaller than DSGD. Meanwhile, it can be seen that our two methods achieve similar performance. Moreover, in Figure 1(b), we plot the loss function value when using the learned model parameter as the intilization for new tasks and then conducting 10 gradient descent steps. It can be seen that our two methods can achieve better performance in adaptation to the new task.



(a) Sine: meta-training (b) Sine: meta-test (c) Omniglo: meta-training (d) Omniglot: meta-test

Figure 1: Meta-training and meta-test for the regression and classification task.

5.3 Classification Task

Besides the regression task, we also verify the performance of our distributed training methods with the classification task. Here, we use the Omniglot image dataset. It has 1623 characters and each character has 20 images. Under the MAML setting, recognizing each character corresponds to a task so that there are 1623 tasks totally. In our experiment, we use 1200 tasks as the training set and the rest tasks as the testing set. To recognize the character, we use the same convolutional neural network as [5]. To get the meta-initialization for the model parameter, we use the 5-way 1-shot setting. As for the distributed training, we still use four devices. The meta batch-size on each device is 8. The number of iterations for adaptation in the training phase is set to 1 while it is set to 3 in the testing phase. Additionally, we set the inner learning rate $\alpha = 0.4$, the learning rate of DSGD $\eta = 0.001$. As for our methods, we keep the same learning rates as the regression task.

From Figure 1(c), we also plot the loss function value for the classification task. We can still find that our two methods converge much faster than DSGD, which further confirms the effectiveness of our proposed methods. Additionally, we show the classification accuracy after adaptation for new tasks in Figure 1(d). It is obvious that our two methods can achieve higher accuracy and smaller variance compared with DSCGD, which further confirms the effectiveness of our methods.

6 Conclusion

In this paper, we studied the distributed training methods for the stochastic compositional optimization problem. In particular, we proposed two novel decentralized stochastic compositional gradient descent method based on the gossip communication mechanism and the gradient tracking communication mechanism. As far as we know, our work is the first one facilitating distributed training for large-scale stochastic compositional optimization problem. Meanwhile, we established the convergence rate of our proposed methods with novel theoretical analysis. The theoretical results indicate that our two methods can achieve linear speedup with respect to the number of devices. This is the first work disclosing this favorable result. At last, the extensive experimental results on the MAML task domenstrate the superior performance of our methods.

References

[1] T. Chen, Y. Sun, and W. Yin. Solving stochastic compositional optimization is nearly as easy as solving stochastic optimization. *arXiv preprint arXiv:2008.10847*, 2020.

[2] A. Cutkosky and F. Orabona. Momentum-based variance reduction in non-convex sgd. In *Advances in Neural Information Processing Systems*, pages 15236–15245, 2019.

[3] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *arXiv preprint arXiv:1407.0202*, 2014.

[4] C. Fang, C. J. Li, Z. Lin, and T. Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 689–699, 2018.

[5] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[6] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi. Decentralized deep learning with arbitrary communication compression. *arXiv preprint arXiv:1907.09356*, 2019.

[7] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1705.09056*, 2017.

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

[9] A. Mokhtari, H. Hassani, and A. Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *Journal of Machine Learning Research*, 21(105):1–49, 2020.

[10] F. Niu, B. Recht, C. Ré, and S. J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *arXiv preprint arXiv:1106.5730*, 2011.

[11] S. Pu and A. Nedić. Distributed stochastic gradient tracking methods. *Mathematical Programming*, pages 1–49, 2020.

[12] H. Sun, S. Lu, and M. Hong. Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking. In *International Conference on Machine Learning*, pages 9217–9228. PMLR, 2020.

[13] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[14] M. Wang, E. X. Fang, and H. Liu. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Mathematical Programming*, 161(1-2):419–449, 2017.

[15] M. Wang, J. Liu, and E. X. Fang. Accelerating stochastic composition optimization. *The Journal of Machine Learning Research*, 18(1):3721–3743, 2017.

[16] R. Xin, U. A. Khan, and S. Kar. A near-optimal stochastic gradient method for decentralized non-convex finite-sum optimization. *arXiv preprint arXiv:2008.07428*, 2020.

[17] J. Yang and W. Hu. Stochastic recursive momentum method for non-convex compositional optimization. *arXiv preprint arXiv:2006.01688*, 2020.

[18] H. Yuan, X. Lian, and J. Liu. Stochastic recursive variance reduction for efficient smooth non-convex compositional optimization. *arXiv preprint arXiv:1912.13515*, 2019.

[19] J. Zhang and L. Xiao. A composite randomized incremental gradient method. In *International Conference on Machine Learning*, pages 7454–7462, 2019.

[20] J. Zhang and L. Xiao. Multi-level composite stochastic optimization via nested variance reduction. *arXiv preprint arXiv:1908.11468*, 2019.

[21] J. Zhang and L. Xiao. A stochastic composite gradient method with incremental variance reduction. In *Advances in Neural Information Processing Systems*, pages 9078–9088, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]