Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Variance reduction in decentralized training over heterogeneous data

Master's Thesis

Yue Liu

liuyue@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Dr. Sebastian U. Stich, Tao Lin
Prof. Dr. Roger Wattenhofer, Prof. Dr. Martin Jaggi

April 7, 2021

Abstract

Large-scale machine learning (ML) applications benefit from decentralized learning since it could execute parallel training and only needs to communicate with neighbors. However, comparing to exact averaging in centralized training, decentralized could only provide inexact averaging, resulting in model difference among workers, or to say model variance. When data partition is heterogeneous, the variance would be even larger and results in more severe client drift with standard Local SGD. In this work, we propose a new decentralized variance reduction method, decentralized SCAFFOLD which uses control variates to correct for the client drift in its local updates. Benefiting from reducing variance in every local steps, we show that decentralized SCAFFOLD is 2-3 times faster in terms of communcation rounds comparing to decentralized Local SGD when heterogeneous data. Further, we show that interpolating a few steps doing SCAFFOLD correction in decentralized Local SGD training could also achieve acceleration and remedy client drift, based on which we build a well-performing mixture decentralized training system with both low computation and low communication complexity.

Acknowledgements

Firstly, I feel so grateful that Dr. Sebastian Stich offers this chance and topic that is related to my research interest in distributed learning in the first place. His expertise in this field and patience guided me overcoming numerous obstacles during the last six months and broadened my horizons in the topic of optimization in machine learning. Besides, I also thank a lot to Tao Lin who offered lots of valuable helps for my thesis. Finally, I own thanks to Prof. Martin and Prof. Roger Wattenhofer for their supervison over the project. Besides, I also would like to thanks my friend Xijin Chen and flatmate Yan Wu, who are always patient enough to listen to my continuous anxiety and complaints during these year of studies in Switzerland. In particular, it's lovely to have their companion at this period of 'self-quarantine', or working from home.

Yue Liu, at Zurich, April 7, 2021

Contents

Abstract **1**

Acknowledgements **2**

1 Introduction **5**

1.1 Contributions 7

1.2 Outline 8

2 Related work **9**

2.1 Non-i.i.d data in decentralized training 9

2.2 Client drift and model variance 9

2.3 Centralized variance reduction methods 10

2.4 Batch Normalization 11

2.4.1 Global Batch Normalization: 12

2.4.2 Individual Batch Normalization: 13

2.4.3 Batch Normalization and Client Variance 15

3 Set up **17**

3.1 Problem description 17

3.1.1 Definitions and Notations 17

3.2 Main assumptions 18

3.2.1 Assumption on objective f 18

3.2.2 Assumption on the noise 19

3.2.3 Assumption on the heterogeneity 19

4 Decentralized variance-reduced training **21**

4.1 Decentralized variance reductioon 21

4.1.1 Variance-reduced algorithm framework 21

4.1.2 Intuition: How it reduces variance? 22

4.1.3 Observation of variance out of toy example 24
4.1.4 Decentralized SVRG 25
4.1.5 Decentralized SCAFFOLD 26
4.1.6 Convergence analysis of decentralized SCAFFOLD 28
4.2 Decentralized SGD/SCAFFOLD training system 29

5 Benchmark Experiment 32
5.1 Detailed Experimental Setup 33
5.2 Variance reduction in i.i.d setting 34
5.3 Normalization Techniques Comparison 36
5.4 Overall comparison 37
5.4.1 Generalization performance and communication rounds . 37
5.4.2 Variance and effect of local steps 40
5.4.3 Computation performance 41
5.4.4 Failure of SVRG 42
5.5 Decentralized SGD/SCAFFOLD training system 44
5.5.1 Local steps activation 45
5.5.2 Outer Loops activation 46
5.5.3 Activating both Outer Loops and Local steps 47

6 Conclusion 49

A Appendix: Convergence result A-1
A.1 Useful properties A-1
A.2 Overview of decentralized SCAFFOLD A-2
A.2.1 Additional definition A-3
A.3 Convergence of decentralized SCAFFOLD for nonconvex function A-3

CHAPTER 1

Introduction

Highly parameterized deep neural network shows significant improvement over machine learning task, but it also results in dramatic increase in the size, complexity, and computational power of the training systems. In this case, in order to accelerate and to make use of more computing power, whole training procedure is scaled out to multiple machines via data-parallel. Training samples are partitioned across multiple decentralized edge devices separately and also the model training are distributively executed across them without exchanging local samples. The target of distributed training is as follow,

$$f^\star := \min_{x \in \mathbb{R}^d} [f(x) = \frac{1}{N} \sum_i f_i(x)],$$

where N is the number of edge computing nodes or workers and $f_i = \mathbf{E}_{\xi \sim \epsilon D_i} F_i(x; \xi)$ (D_i : local data distribution).

Large mini-batch parallel SGD (1; 2) is adopted to distributed setting. Workers compute local gradients of the loss function on those different partition simultaneously, and then calculate an exact averaged gradient using either the ALLREDUCE communication primitive (3) or via central controller called parameter server (4). But the collection of all workers for parameter sever would experience severe communication bottleneck and center node failure (5). To this end, client sampling out of large amount of workers is proposed to solve bottleneck from network bandwidth (6; 7; 8), or to train model through fully decentralized fashion (9; 10; 11; 12) could also get rid of the center node for more efficient large scale training, where every worker only need to communicate with neighbors.

Variance among workers when heterogeneous data Unlike the variance in common sense, the variance considered here refers to deviation between models. The performance of both centralized or decentralized training via just SGD would suffer from variance among workers when heterogeneous partition. For centralized training, Local SGD reduces communication cost by performing K local updates before centralized aggregation $x = \frac{1}{N} \sum_i y_{i,K}$, where $y_{i,k}$ is the corresponding local model at time $k \in [0, K]$. While it has shown success in certain applications, its performance on heterogeneous data would be actually deteriorated. When data partition is heterogeneous, performing too many local steps would make local models develop in different directions. It introduces 'client-variance' in the updates across the different clients (6; 13). Thus variance is the deviation among workers during the local steps. For decentralized training, in addition to local steps, since no centralized aggregation is allowed, every worker could only communicate with neighbors. Thus each work has its own local estimation of global model estimator x_i instead of the globally acknowledged model x in centralized case. Variance could also be the deviation among workers w.r.t final model x_i, which results in a more severe and common problem in decentralized case. That's, as long as gossip averaging over graph except complete graph, inexact averaging would always brings in variance among workers, regardless of being non-i.i.d or not.

$$\frac{1}{N}\sum_i ||y_{i,k} - \bar{y}_k||^2 \text{ and } \frac{1}{N}\sum_i ||x_i - \bar{x}||^2,$$

where $y_{i,.k}$: local model at local step k, and x_i: aggregated global model.

Batch normalization layer and variance Batch normalization layer (14) is a popular technique used by most of modern nerual network. While it is effective in practice, its dependence on minibatch mean and variance is known to be problematic in parallel training. Most work discussing decentralized training avoids normalization (10).When data partition is heterogeneous, every local worker would be normalized by different local distributions layer by layer, thus resulting in growingly larger deviation between local models(, which is larger variance) (15). Thus it suggests the significance of reducing variance when doing decentralized training of models with normalization layer.

Why variance hurts? Both targets are able to achieve the optimum at the globally averaged model. However, if performing too many local steps $K \gg 1$ when underlying data partition is heterogeneous, then averaged model would be stuck in local optimum of form similar to $\frac{1}{N}\sum_i x_i^\star$, where $x_i^\star := \arg\min_{x\in\mathbb{R}^d} f_i(x)$. And it has been detected and confirmed as client drift (6; 16; 17; 18; 15). Thus it shows that the larger the variance, the more severe the drift could be.

Variance reduction method Since model variance has negative impact on decentralized training, variance reduction plays an important role. However, in single machine, the standard Stochastic variance reduction (SVR) is considering reducing the variance brought by stochastic gradient, or noise. It consists of a collection of techniques SVRG (13), SAGA (19), SARAH (20), etc. It employs extra control variates c, $\tilde{c}$ to correct stochastic gradient (SGD) so as to approximate the full batch gradient and to reduce impact from stochasticity. It iteratively updates model y with corrected gradient, and it updates those control variates by calculating full batch gradient w.r.t snapshot point x. If $c \approx \nabla F(y;\xi)$, $\tilde{c} \approx \nabla f(y)$,

$$y = y - \eta(\nabla F(y;\xi_i) - c + \tilde{c}) \approx y - \eta\nabla f(y). \tag{1.1}$$

But this approximation via correction could also be applied to help reduce the model variance in distributed learning. In the distributed setting, D-SVRG (21; 22; 23) in master/slave model and Network-SVRG(24) combined with gradient tracking in decentralized mode reduce the noise of the local stochastic gradient of each node so as to accelerate convergence. But they both have hard constraint over model dissimilarity(or variance). SCAFFOLD is also SVRG-like designed particularly to remedy the client drift (6) and supports client sampling, but it's only designed for the centralized setting and would suffer from central node failure. In the decentralized case, GT-SVRG and GT-SAGA (25) uses correction combined with gradient tracking to reduce both stochastic noise and model variance, but it needs communication every model updates which requires too much communication overhead.

In this work, we mainly consider the decentralized SCAFFOLD that could reduce communication cost by performing K local steps and reduce model variance in decentralized setting. Intuitively, similar to (1.1), we estimates direction of each client $c_i \approx \nabla F_i(y_i; \xi_i)$, and of global direction $\tilde{c}_i \approx \frac{1}{N}\sum_i \nabla F_i(y_i; \xi_i)$. Then for each model update, we correct the gradient with direction difference $-c_i + \tilde{c}_i$,

$$y_i = y_i - \eta(\nabla F_i(y_i; \xi_i) - c + \tilde{c}) \approx y_i - \eta \frac{1}{N} \sum_j \nabla F_j(y_j; \xi_j),$$

which aprroximates the ideally bulk synchronous parallel (26), the most communication-heavy approach. Thus it mitigates the client drift brought by local steps and decentralized communication on heterogeneous data. We use this viewpoint to show that decentralized SCAFFOLD is relatively unaffected by heterogeneity and arbitrary decentralized communication pattern. Comparing to other existing method in the decentralized training, e.g, DSGD (9), decentralized SCAFFOLD is able to reduce the variance among workers and accelerates in terms of communication rounds for models, regardless of whether having normalization layer or which normalization technique used. And even doing local steps with iterative online choice either pure SGD or modified SGD used withn decetralized SCAFFOLD, very few steps of correction within decentralized SCAFFOLD could be already enough to reduce variance, remedy client drift and accelerate at the same time.

1.1 Contributions

The goal is to train deep neural network on top of heterogeneous data in decentralized fashion with both low communication and computation overhead. In this work, we focus on a common type of non-i.i.d data, widely used in prior work (27; 28; 29): skewed distribution of data labels across locations/devices. Our study covers typical DNN models, training datasets, degrees of label skewness, decentralized learning algorithms and communication topologies.
Contributions We summarize our main results below.

- We propose the decentralized SCAFFOLD from its originally centralized version and give the theoretical analysis over its convergence rate on heterogeneous data.
- We empirically evaluate the performance of decentralized SCAFFOLD, and demonstrate that this method is robust against the non-i.i.d statistical heterogeneity and also outperforms decentralized Local SGD in both communication efficiency and generalization performance.
- We validate the effectiveness of decentralized SCAFFOLD combined with different normalization techniques, and shows that model quality are less dependent of normalization techniques but more related to algorithm when heterogeneous data. The performance of decentralized SCAFFOLD is consistently outperforms decentralized SGD with all normalization techniques.
- We propose a decentralized training system with mixture SGD and SCAFFOLD steps. The combination with small fraction of SCAFFOLD for SGD could further improve decentralized training of DNN models with low computation, low communication overhead and the best generalization performance on arbitrary communication network.

1.2 Outline

The remainder of this thesis is structured as follows. We introduces the preliminaries and background in Section 2. We describe the problem set up and main assumption in Section 3, and present our decentralized variance reduction methods in Section 4. Section 5 present the experimental results, followed by conclusion in Section 6.

Chapter 2

Related work

2.1 Non-i.i.d data in decentralized training

Decentralized Learning aims to train a local inference model for every workers from locally distributed data $\{D_1, ..., D_m\}$ but to test on global test data D. Those datasets are usually generated from different source or endpoint, and could be very different from each other. Non-i.i.d has always been a key challenge in distributed learning. And when local distribution D_i deviates from global true distribution D, the local objective of each worker is inconsistent with the global optima. Considering the different distributional shift, there are two different non-i.i.dness types, label distribution skew(18; 30) and feature distribution skew(18; 31). In label distribution skew, the label distributions $P(a|i)$ (a : label, i : worker index) vary across workers. Such a case is common in practice. For example, when user's activity has specific preference, the data collected from different users would also show the bias over choices. In feature distribution skew, the feature distributions $P(x_i)$ (x : model) vary across workers although the knowledge $P(a|x_i)$ is same. For example, when doing whether forecasting, different place is influenced by different landscape. In this work, I mainly consider the label distribution skew type of heterogeneity.

2.2 Client drift and model variance

Heterogeneous data is a common situation and a typical challenge in general distributed training. Client drift (6; 16; 17; 18; 15) is known to be detected in Local SGD when heterogeneous data partition. In centralized training, when data partition is non-iid, functions for each worker $\{f_i\}$ is distinct, then the direction of gradient calculated from $\{f_i\}$ would be very different and local updates would experience drift from the server model. Contrary to finding the global optimum $x^\star$, Local SGD would end up with $\frac{1}{N}\sum_i x_i^\star$. The amount of this client drift is depicted and exactly determined by the gradient dissimilarity parameter G, B in Assumption 3.7.

In addition, in decentralized training, there is no server to maintain a common and global model. Every worker keeps their own model $\{x_i\}$. The deviation between $\{x_i\}$ and $\bar{x} = \frac{1}{N}\sum_i x_i$ is another source of drift. When i.i.d setting, in centralized training, the deviation of local models y_i resulted from noise of stochastic gradient via local steps would be compromised by the exact averaging later in model aggregation of server. But in decentralized training, every worker could only communication with their neighbors through gossip averaging. This kind of inexact averaging cannot alleviate the noise but result in deviation among workers in global model x_i and for the start of next round of local steps, every worker already has very different starting point. Then the slight deviation accumulates and results in drift even when i.i.d setting in Figure 5.2a and Figure 5.3a, and could be depicted by significant model variance $\frac{1}{N}\sum_i^N ||x_i - \bar{x}||^2$. The ultimate goal of decentralized training is to find optimum point on $\bar{x}^\star$, and meantime, every worker is optimum $\bar{x}^\star$. That's zero-variance among workers.

2.3 Centralized variance reduction methods

D-SVRG(21)

D-SVRG is the combination of SVRG(13) and DANE(32). Instead of solving the DANE subproblem exactly, it uses SVRG as the local solver to produce an approximated solution. It could be able to converge faster(21) than Local SGD in terms of communication rounds.

Algorithm 1 D-SVRG

```
1: **Initialize:** $x_i^0 = x_j^0,\ \forall i \in [n],\ \tilde{c}_i^0 = \vec{0} = c_i^0$
2: **for all** client $i \in \{1, ..., n\}$ parallel **do**
3:   **for outer loops:** $r \leftarrow 1$ to $R$ **do**
4:     $y_{i,0}^r = x_i^r, v_{i,0}^r = \tilde{c}_i^r$
5:     **for inner loops:** $k \leftarrow 0$ to $K-1$ **do**
6:       Sample $\xi_{i,k}$ from $D_i$ and compute $\nabla F(y_{i,k}^r; \xi_{i,k})$
7:       $v_{i,k}^r = \nabla F_i(y_{i,k}^r; \xi_{i,k}) - \nabla F_i(x_i^r; \xi_{i,k}) + \tilde{c}_i^r$
8:       $y_{i,k+1}^r = y_{i,k}^r - \eta_c v_{i,k}^r$
9:     **end for**
10:    $x^{r+1} = \frac{1}{N}\sum_i^N y_{i,K}^r$   ▷ update global model
11:    $c_i^{r+1} = \nabla f_i(x^{r+1})$   ▷ local full-batch gradient as global control variate
12:    $\tilde{c}^{r+1} = \frac{1}{N}\sum_i^N c_i^{r+1}$   ▷ update global control variate
13:  **end for**
14: **end for**
```

However, in its theorectical analysis, it puts hard constraints over the heterogeneity and also the optimizer SVRG itself suffers from ineffectiveness in deep neural network training(33). Thus, even though it fits in the general distributed variance reduction methods, it still needs more investigation on how it performs in distributed training of DNN models.

SCAFFOLD(6)

SCAFFOLD models on non-i.i.d as introducing variance among the workers and applies the variance reduction technique. It estimates control variates for the server c and workers c_i respective for estimator of direction of server model and local model. Then, the drift of local training is approximated by the difference between these two update directions, $c_i - \tilde{c}$. Thus, during local model update, SCAFFOLD corrects direction with the opposite direction of estimated drift. Here we only consider the option I where it reuses the previously computed gradients so as to save the extra computation of full-batch gradient. Compared with Local SGD(34), intuitively, SCAFFOLD doubles the communication model size(x and $\tilde{c}$) per communication round and also doubles the computation due to the extra modification $c_i - \tilde{c}$ on SGD.

Algorithm 2 SCAFFOLD

```
1: Initialize: $x_i^0 = x_j^0,\ \forall i \in [n],\ \tilde{c}_i^0 = \vec{0} = c_i^0$
2: for all client $i \in \{1, ..., n\}$ parallel do
3:   for outer loops: $r \leftarrow 0$ to $R-1$ do
4:     $y_{i,0}^r = x_i^r$
5:     for inner loops: $k \leftarrow 0$ to $K-1$ do
6:       Sample $\xi_{i,k}$ from $D_i$ and compute $\nabla F(y_{i,k}^r; \xi_{i,k})$
7:       $y_{i,k+1}^r = y_{i,k}^r - \eta_c(\nabla F_i(y_{i,k}^r; \xi_{i,k}) - c_i^r + \tilde{c}_i^r)$
8:     end for
9:     $x^{r+1} = x^r + \eta_s \frac{1}{N} \sum_i^N (y_{i,K}^r - x^r)$    ▷ global model
10:    $c_i^{r+1} = c_i^r - \tilde{c}^r + \frac{1}{K\eta_c}(x^r - y_{i,K}^r)$    ▷ local control variate with option I
11:    $\tilde{c}^{r+1} = \tilde{c}^r + \frac{1}{N} \sum_i^N (c_i^{r+1} - c_i^r)$    ▷ local control variate
12:  end for
13: end for
```

2.4 Batch Normalization

It is well-known that the latest deep learning models usually adopt Batch Normalization (BN) (14) to facilitate and stabilize optimization. A BN layer normalized input data z

$$\hat{z} = \gamma \frac{z - \hat{\mu}}{\hat{\sigma}^2} + \beta$$

via the batch statistcs $\hat{\mu} = \mu_B, \hat{\sigma}^2 = \sigma_B^2$, and for later evaluation, it keeps running estimates $\bar{\mu}$ and $\bar{\sigma}^2$ through accumulated averaging of each batch statistics. $m \in [0, 1]$ is the parameter for controlling momentum.

$$\bar{\mu} = m\bar{\mu} + (1-m)\hat{\mu},\ \bar{\sigma}^2 = m\bar{\sigma}^2 + (1-m)\hat{\sigma}^2$$

However, classical distributed algorithm and most recent works, including both the centralized and the decentralized, avoid normalization. Because normalization technique, even though works pretty well in practice, puts great challenge to theoretical studies. Methods could work in general non-convex functions as theorectial analysis suggested, but situation for models with normalization could be different. And even in practice, since Batch Normalization needs to calculate batch statistics for every hidden layers and keeps running estimates for later evaluation, the major concern is that its performance is particular vulnerable to heterogeneous data distribution(14).Thus how to reasonably normalize under distributed setting and in particular when underlying distribution for every worker is different, is still unclear.

Unlike the previous case in single machine, in distributed setting, since data and computation are located in decentralized devices, each worker would maintain their own estimator of those parameters in BN layer. In addition, if performing local steps $k \in \{0, ..., K\}$ and at communication rounds $r \in \{1, ..., R\}$, the notation would be

$$\hat{\mu}_{i,k}^r,\ (\hat{\sigma}^2)_{i,k}^r,\ \bar{\mu}_{i,k}^r,\ (\bar{\sigma}^2)_{i,k}^r,\ \gamma_{i,k}^r, \beta_{i,k}^r.$$

Considering how those parameters would communication among workers throughout training and evaluation, there classifies BN into two different category: Global Batch Normalization and Individual Batch Normalization.

2.4.1 Global Batch Normalization:

As names suggests, doing global normalization is to make sure to normalize with same batch statistics. And only then we could guarantee that data is normalized by the same and true underlying distribution.

1. **Synchronized Batch Normalization(SyncBN)(14; 35)**The naive thoughts for distributed setting to mimic the behavior in single machine is to globally synchronize batch statistics for forward and backward operation to guarantee that every worker normalize based the same distribution, only then there is no concern over heterogeneity across workers. For forward operation, it calculates batch statistics via

$$\hat{\mu} = \frac{1}{N}\sum_j \mu_{B_j},\ \hat{\sigma}^2 = \frac{1}{N}\sum_j \sigma_{B_j}^2,\ \hat{z}_i = \gamma_i^r \frac{z_i - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \epsilon}} + \beta_i^r,$$

and for backward operation, it calculates gradient w.r.t input,

$$\frac{\partial f}{\partial z_{ik}} = \frac{1}{\sqrt{\hat{\sigma}^2+\epsilon}}(\frac{\partial f}{\partial \hat{z}_{ik}} - \frac{1}{N}\sum_j \frac{1}{N}\sum_s \frac{\partial f}{\partial \hat{z}_{js}} - \hat{z}_{i,k}\frac{1}{N}\sum_j \frac{1}{N}\sum_s \frac{\partial f}{\partial \hat{z}_{js}}\hat{z}_{js})).$$

Then for running estimates, since for every update, the used batch statistics are the same for all workers, then the running estimates it keeps through classical momentum are the same. In particular, aggregate the affine parameters along with every communication as part of model,

$$\gamma_{i,0}^{r+1} = \sum_j w_{ij}\gamma_{j,K}^r,\ \beta_i^{r+1,0} = \sum_j w_{ij}\beta_{j,K}^r.$$

However, such fully synchronized Batch Normalization could only happen for centralized training, and be against the intrinsic property of federated learning, where it communicates much less. SyncBN needs to communicate for every forward and backward operation, which puts significantly heavy communication overhead, and it has servere concern over privacy.

2.4.2 Individual Batch Normalization:

However, in the truly distributed setting, it's impossible to perform global normalization, but to do it independently for each worker.

$$\hat{\mu}_{i,k},\ \hat{\sigma}_{i,k}^2,\ \hat{z}_{i,k}^r = \gamma_{i,k}^r\frac{z_i^r - \hat{\mu}_{i,k}}{\sqrt{\hat{\sigma}_{i,k}^2+\epsilon}} + \beta_{i,k}^r,$$

Considering how to operate on components needed for normalization(running estimates and affine parameters), there are three different and somewhat hierarchy methods, from completely synchronizing all components to completely asynchronizing(keeping both locally). And it's already investiated that doing batch normalization individually would harmonize local data distribution in the field of domain adoptation(36; 37), and only then model aggregation could be beneficial.

1. **Local Batch Normalization(LBN)(14)** More natural and naive way for training deep learning model with Batch Normalization in federated learning fashion is to do the Batch Normalization locally, and to average the running estimates later alone with model aggregation.

$$\bar{\mu}_i^{r+1} = \sum_j w_{ij}\bar{\mu}_{j,K}^r,\ (\bar{\sigma}_i^2)^{r+1} = \sum_j w_{ij}(\bar{\sigma}_{j,K}^2)^r,$$

$$\gamma_i^{r+1} = \sum_j w_{ij}\gamma_{j,K}^r,\ \beta_i^{r+1} = \sum_j w_{ij}\beta_{j,K}^r,$$

where $\hat{\mu}_i,\ \hat{\sigma}_i^2$ used for normalization during training are the unbiased estimators w.r.t local distribution D_i. And for the linear affine parameters $\gamma_i^r,\ \beta_i^r$ and running estimates $\bar{\mu}_i^r,\ (\bar{\sigma}^2)^r$ will be averaged through gossip as part of model when model aggregation. Thus doing local Batch Normalization is to leave the deviation brought by normalized by different local distribution for later modification to gradient or model aggregation within federated learning.

2. **SiloBN(38)** SiloBN is adopted from domain adoptation(36). For training, it's the same as LBN, using local batch statistics to normalize and treating affine parameter as part trained model waiting for later model aggregation. But it also keeps running estimates locally, unlike that in LBN.

$$\bar{\mu}_i^{r+1} = \bar{\mu}_i^r,\ (\bar{\sigma}_{i,K}^2)^{r+1} = (\bar{\sigma}_{i,K}^2)^r,$$

$$\gamma_i^{r+1} = \sum_j w_{ij}\gamma_{j,K}^r,\ \beta_i^{r+1} = \sum_j w_{ij}\beta_{j,K}^r.$$

The idea behind it is straightforward and provides different perspective of standardized distribution. It treats batch statistics and affine parameters differently, where the former encodes local dataset information, the latter can be employed to transfer information across different workers. Instead of making sure every worker uses the same batch statistics, SiloBN standardizes data to uniform zero-mean and unit variance using local batch statistics, and the affine parameter here helps to expand data distribution to more standardized β_i^r-mean and $(\gamma_i^r)^2$-variance, the deviation of which could be tackled by algorithm. Keeping BN statistics local permits the decentralized training of a model robust to the heterogeneity, as being normalized by local statistics ensures that the intermediate activations of all workers are centered to a similar value(approximate zero-mean and unit-variance).

3. **FedBN(39)** FedBN is somewhat more extreme and completely asychronized w.r.t Batch Normalization layer. It not only noramlizes with local batch statistics but also never communicates affine parameters.

$$\bar{\mu}_{i,0}^{r+1} = \bar{\mu}_{i,K}^r,\ (\bar{\sigma}_{i,0}^2)^{r+1} = (\bar{\sigma}_{i,K}^2)^r,$$

$$\gamma_{i,0}^{r+1} = \gamma_{i,K}^r,\ \beta_{i,0}^{r+1} = \beta_{i,K}^r.$$

It's shown in (39) that once having two nice models w_1, w_2 and their corresponding γ_1, γ_2 respectively, the performance of $\bar{w} = (w_1 + w_2)/2, \bar{\gamma} = (\gamma_1 + \gamma_2)/2$ has the largest generalization error, but $\bar{w}$ with either γ_1 or γ_2 still could perform well as w_1, γ_1 or w_2, γ_2. It suggests that keeping even affine parameters local could help not deteriorate the performance when model aggregation.

Batch Normalization overview				
	technique	**batch statistics** $\hat{\mu}_i$	**running statistics** $(\bar{\mu}_{i,0}^{r+1}, (\bar{\sigma}_{i,0}^2)^{r+1})$	**affine parameters** $(\gamma_{i,0}^{r+1}, \beta_{i,0}^{r+1})$
Global BN	SyncBN	$\hat{\mu} \equiv \frac{1}{N}\sum_j \mu_{B_j}$	$\frac{1}{N}\sum_j(\bar{\mu}_{j,K}^r, (\bar{\sigma}_{j,K}^2)^r)$	$\sum_j w_{ij}(\gamma_{j,K}^r, \beta_{j,K}^r)$
Individual BN	LBN	μ_{B_i}	$\sum_j w_{ij}(\bar{\mu}_{j,K}^r, (\bar{\sigma}_{j,K}^2)^r)$	
	SiloBN		$(\bar{\mu}_{i,K}^r, (\bar{\sigma}_{i,K}^2)^r)$	
	FedBN			$(\gamma_{i,K}^r, \beta_{i,K}^r)$

Table 2.1: **How the operation changes for each component within Batch Normalizaion techniques**

Since SyncBN is just an ideal fully synchronized normalization technique where every worker could be able to be normalized by the same distribution for every step. It had conformed to initial conjecture over solving the non-i.i.d challenge for model with normalization layer, however, later it shows that SyncBN with Local SGD doesn't outperform that much other instance-based normalization method, e.g, GN, in terms of generalization performance in Figure 5.4b or of consensus distance in Figure 5.4a, while it's already known those instance-based normalization suffers from some degree of quality loss comparing to BN(40; 18).

Thus it suggests that if using BN, the guarantee with exactly the same distribution is not the key factor counting for failure of decentralized Local SGD on heterogeneous data. The naive thought of modifying BN so as to be normalized by approximately global distribution counterintuitively seems not to be promising. And later in detailed experimental section, it will show that doing normalization locally, and treating the output of locally normalized data as nothing different and just an internal part of local objective f_i is still workable. Those all the internal impacts combined on f_i resulting from heterogenous data could be as a whole solved by suitable external optimization technique.

2.4.3 Batch Normalization and Client Variance

The heterogeneous data affects BN in two different levels. First from 1) on same model but different local data then to 2) different model and different local data. Since standard BN are particularly vulnerable to skewed label partitions, when under data partition is heterogeneous, distributed training of deep neural network would suffer from severe quality loss(15). However, it's also investigated in more extreme non-i.i.d cases, e.g domain adoptation, that doing BN completely locally actually mitigates domain shift and harmonizes different local distribution w.r.t local model(39; 41). Thus model trained on heterogeneous data could be able to end up with alignment with each other comparing to model without(Figure 1 in (39)). It suggests that if guaranteeing the same model on all workers, then heterogeneity could be directly mitigated by BN, since the model updated on non-i.i.d data are still harmonized with each other w.r.t parameter space. Then it further suggests that the closer the models are, the better the aggregated model could be. Smaller client variance would take advantage of this alignment brought by individual BN (doing BN completely locally) to end up with better global model in distributed training.

CHAPTER 3

Set up

In this chapter, we introduce the standard set up for problem formulation, basic notation and the main assumption.

3.1 Problem description

Suppose we have a dataset $\mathbf{Z}$. In distributed setting, data is partitioned and distributed on n different workers, the partition of dataset is defined as $\mathbf{Z} = \bigcup_i^N \mathbf{Z}_i$, where $\mathbf{Z}_i \bigcap \mathbf{Z}_j = \varnothing, \forall i \neq j$. The data partition $\mathbf{Z}_i$ is stored on worker i, $1 \leq i \leq N$. No further assumption on statistical distribution of each node D_i is made. Consider the following distributed minimization problem

$$\min_{x \in \mathbf{R}^d} f(x) := \frac{1}{N} \sum_i^N f_i(x), \text{ where } f_i(x) = \mathbb{E}_{\xi_i \sim D_i} F_i(x; \xi_i) \tag{3.1}$$

$f_i(x)$ is defined to be the expected local loss for worker i w.r.t local distributed D_i, and $F_i(x;\xi)$ is the stochastic loss. Also, denote $\bigtriangledown f_i(x)$ as the expected gradient, and $\bigtriangledown F_i(x,\xi)$ as the stochastic gradient.

If finite sample, instead of considering expectation loss, the empirical loss would be evaluated. B is the maximum number of sample per node.

$$\min_{x \in \mathbf{R}^d} f(x) := \frac{1}{N} \sum_i^N \frac{1}{B} \sum_z F_i(x; z). \tag{3.2}$$

3.1.1 Definitions and Notations

Throughout this thesis, the following notations and definitions are used:

Table 3.1: summary of notation

$N,\ i$	total number, index of worker
$R,\ r$	number, index of communication rounds
$K,\ k$	number, index number of inner loops
$x_i^r,\ y_{i,k}^r \in \mathbb{R}^d$	global model, local model, of worker i at communication round r, and local steps k.
$\tilde{c}_i^r,\ c_i^r$	global, local control variate, of worker i at communication round r
$W,\ w_{ij}$	adjacency matrix, entries of topology at time t in decentralized setting
$\eta_c,\ \eta_s$	client step size, model step size

We use $x_i^r \in \mathbb{R}^d$ denotes the model estimates for worker i at time step r, and further define the average over all workers

$$\bar{x}^r := \frac{1}{N}\sum_i^N x_i^r \in \mathbb{R}^d. \tag{3.3}$$

And the equivalent matrix and vector notation is defined as following.

W denotes the adjacency matrix of decentralized network topology.

$$\begin{aligned} &W_\infty := \frac{11^T}{N} \in \mathbb{R}^{N\times N} \\ &X^r := (x_1^r, ..., x_N^r) \in \mathbb{R}^{d\times N} \\ &\nabla F(X;\xi) := (\nabla F_1(x_1;\xi_1), ..., \nabla F_N(x_N;\xi_N)) \in \mathbb{R}^{d\times N} \\ &\nabla F(X) := (\nabla f_1(x_1), ..., \nabla f_N(x_N)) \in \mathbb{R}^{d\times N} \end{aligned} \tag{3.4}$$

Then easily there are some useful terms calculated in matrix format,

$$\begin{aligned} &\bar{X}^r := (\bar{x}^r, ..., \bar{x}^r) \equiv X^r\frac{11^T}{N} \equiv X^r W_\infty \in \mathbb{R}^{d\times N} \\ &\nabla F(\bar{X}) := (\nabla f_1(\bar{x}), ..., \nabla f_N(\bar{x})) \in \mathbb{R}^{d\times N} \\ &\overline{\nabla F(X)} := \frac{1}{N}\sum_i^N f_i(x_i) \in \mathbb{R}^d \\ &f(\bar{x})1^T = (f(\bar{x}), ..., f(\bar{x})) \in \mathbb{R}^{d\times N} \end{aligned} \tag{3.5}$$

And in particular, there is also $\mathbb{E}_\xi \nabla F(X;\xi) = \nabla F(X)$.

3.2 Main assumptions

3.2.1 Assumption on objective f

Assumption 3.1. (L-smoothness) Each function $F_i(x;\xi) : \mathbb{R}^d \to \mathbb{R},\ \xi \sim D_i$ is differentiable and there exists a constant $L \geq 0$ such that for $\forall x, y \in \mathbb{R}^d$,

$$||\nabla F_i(x;\xi) - \nabla F_i(y;\xi)|| \leq L||x - y||. \tag{3.6}$$

Assumption 3.2. (L-smoothness) Each function $f_i(x) : \mathbb{R}^d \to \mathbb{R},\ i \in [n]$ is differentiable and there exists a constant $L \geq 0$ such that for $\forall x, y \in \mathbb{R}^d$,

$$||\nabla f_i(x) - \nabla f_i(y)|| \leq L||x - y|| \tag{3.7}$$

Remark 3.3. Sometimes it's enough to just assume the smoothness of function $f_i(x)$. Assumption 3.2 is more general than Assumption 3.1, and Assumption 3.1 can imply Assumption 3.2 for convex $f_i(x)$(9).

3.2.2 Assumption on the noise

Assumption 3.4. (uniform bounded noise) For every worker $\forall i \in [N]$, the variance w.r.t each f_i is uniform.

$$\mathbb{E}_{D_i}||\nabla F_i(x;\xi) - \nabla f_i(x)||_2^2 \leq \sigma^2 \tag{3.8}$$

Assumption 3.5. (bounded gradient variance) There exists constant $M, \hat{\sigma}^2$ such that $\forall x_1, ..., x_n \in \mathbb{R}^d$

$$\Psi \leq \hat{\sigma}^2 + \frac{M}{n}\sum_i^n ||\nabla f(x_i)||_2^2, \tag{3.9}$$

where $\Psi := \frac{1}{n}\sum_i^n \mathbb{E}_{\xi\sim D_i}||\nabla F_i(x;\xi) - \nabla f_i(x)||_2^2$.

Remark 3.6. When uniform bounded on noise,

$$\frac{1}{N}\sum_i \mathbb{E}_{D_i}||\nabla F_i(x;\xi) - \nabla f_i(x)||_2^2 \leq \sigma^2,$$

which is the special case for bounded gradient variance when $M = 0,\ \hat{\sigma}^2 = \sigma^2$. Thus it's easy to check that Assumption 3.5 is weaker than assuming uniform bound on the noise σ_i^2 in Assumption 3.4.

3.2.3 Assumption on the heterogeneity

Assumption 3.7. ((G,B)-bounded dissimilarity) We assume there exists constant $B \geq 1,\ G$ such that $\forall x \in \mathbb{R}^d$,

$$\frac{1}{N}\sum_i^N ||\nabla f_i(x)||_2^2 \leq G^2 + B^2||\nabla f(x)||_2^2 \tag{3.10}$$

Remark 3.8. Another commonly used assumption on dissimilarity is to assume uniform bound on gradient, $||\nabla f_i(x)||^2 \leq G^2$(34; 42). When bounded gradient, then

$$\frac{1}{N}\sum_{i}^{N} ||\nabla f_i(x)||^2 \leq G^2,$$

which clearly satisfied Assumption 3.7.

Chapter 4

Decentralized variance-reduced training

In this section, we mainly discuss the common problem encountered within heterogeneous decentralized learning tasks and propose the variance-reduced algorithm, decentralized SVRG and decentralized SCAFFOLD, combined with corresponding normalization technique for models with normalization layer.

4.1 Decentralized variance reductioon

In this section, we first describe the general decentralized variance-reduced framework, and then discuss how it has the potential to solve the problem of large variance among workers and client drift.

4.1.1 Variance-reduced algorithm framework

Here now present the general decentralized variance-reduced framework. It's based on the original SVRG(13) in the single machine, classifiying iterations into two phases, one for updating the variance reduction components, one for model update with modified stochastic gradient. And within the framework, it has three main steps:

- local updates to the local model, $y_i = y_i - \eta_c(\nabla F_i(y_i;\xi) - c_i + \tilde{c}_i)$
- local updates to the client control variate, $c_i = \textbf{local\_control\_variate}(\cdot)$
- and aggregating the updates. $(x_i, \tilde{c}_i) = (\textbf{global\_model}(\cdot), \textbf{global\_control\_variate}(\cdot))$

Along with the general decentralized variance-reduced framework, since no global communication is allowed, those parameters of model and control variates are having two sets of estimators, one for local estimation of local value $y_i = \textbf{local\_model}(\cdot)$, $c_i = \textbf{local\_control\_variate}(\cdot)$) and the other for the local estimation of global

Algorithm 3 General decentralized variance-reduced algorithm framework

```
1: **parameters:**
2: $R$: number of communication rounds;
3: $K$: number of local steps;
4: $\eta_c$: learning rate for local model update;
5: $\eta_s$: learning rate for global model update;
6: $W$: adjacency matrix of decentralized topology.
7: **Initialize:** $x_i^0 = x_j^0,\ \forall i, j \in [n];\ \tilde{c}_i^0 = \vec{0} = c_i^0,\ \forall i \in [n]$
8: **for all** client $i \in \{1, ..., n\}$ parallel **do**
9:     **for outer loops:** $r \leftarrow 0$ to $R-1$ **do**
10:        $y_{i,0}^r = x_i^r$
11:        **for inner loops:** $k \leftarrow 0$ to $K-1$ **do**
12:            $y_{i,k+1}^r = y_{i,k}^r - \eta_c(\nabla F_i(y_{i,k}^r; \xi_{i,k}) - c_i^r + \tilde{c}_i^r)$
13:        **end for**
14:        $x_i^{r+1} = \textbf{global\_model}(\{y_{1,K}, .., y_{n,K}\}; \eta_s, W)$
15:        $c_i^{r+1} = \textbf{local\_control\_variate}(\cdot)$
16:        $\tilde{c}_i^{r+1} = \textbf{global\_control\_variate}(\cdot; W)$
17:    **end for**
18: **end for**
```

value $x_i = \textbf{global\_model}(\cdot)$, $\tilde{c}_i = \textbf{global\_control\_variate}(\cdot)$), via decentralized communication with neighbors and gossip averaging of each others' local value. And different methods have various ways of estimations. And if $c_i \equiv 0$, then the algorithms is equivalent to decentralized SGD.

4.1.2 Intuition: How it reduces variance?

Variance here refer to the model deviation(local model: y_i and global model x_i) among workers instead of the noise in stochastic oracle resulted from SGD. If communication cost is not a concern, the ideal model update at worker i would be

$$y_{i,k+1} = y_{i,k} - \eta_c \frac{1}{N} \sum_i \nabla F_i(y_{i,k}; \xi_{i,k}). \tag{4.1}$$

Only then each work is completely synchronized every local step. And $\frac{1}{N}\sum_i \nabla F_i(y_{i,k}; \xi_{i,k})$ provides perfect unbiased estimator of gradient of $f = \sum_i f_i$. And the performance is equivalent to ideal centralized Local SGD at i.i.d setting. However, it needs communication with every worker at every update step. In decentralized training only allows communication with neighbors resulting in variance among workers $||x_i - \bar{x}||^2 \geq 0$, and only allows communication every K steps.
Then consider the case in decentralized variance reduction. Since both c_i, $\tilde{c}_i$ are constant throughout all local steps given one communication round, if

$$c_i \approx \nabla F_j(y_{j,k}; \xi_{j,k})\ \forall k \in [K],$$

$$\tilde{c}_i \approx \frac{1}{N}\sum_j \nabla F_j(y_{j,k};\xi_{j,k}) \ \forall k \in [K],$$

for each worker and any local steps within the communication round,

- the gradient update made would be

$$\begin{aligned}\nabla F_i(y_{i,k};\xi_{i,k}) - c_i + \tilde{c}_i &\approx \nabla F_i(y_{i,k};\xi_{i,k}) - \nabla F_i(y_{i,k};\xi_{i,k}) + \frac{1}{N}\sum_j \nabla F_j(y_{j,k};\xi_{j,k}) \\ &= \frac{1}{N}\sum_j \nabla F_j(y_{j,k};\xi_{j,k}),\end{aligned} \tag{4.2}$$

  which means that variance reduction method has the ability to mimic the synchronization in (4.1) even in decentralized training.

- At mean time, define $\bar{y}_k := \frac{1}{N}\sum_i y_{i,k}$, and if $\frac{1}{N}\sum_i c_i = \frac{1}{N}\sum_i \tilde{c}_i$,

$$\bar{y}_k = \bar{y}_{k-1} - \eta_c \frac{1}{N}\sum_i \nabla F_i(y_{i,k-1};\xi_{i,k-1})$$

  Consider the variance among workers $\frac{1}{N}\sum_i ||y_{i,k} - \bar{y}_k||^2$. Define $S_i := \nabla F_i(y_{i,k-1};\xi_{i,k-1}) - \frac{1}{N}\sum_j \nabla F_j(y_{j,k-1};\xi_{j,k-1})$. For simplicity, if assuming at one time $y_{i,k-1} = \bar{y}_{k-1}$, then we only comparing the distance after one local step,

$$\begin{aligned}&\frac{1}{N}\sum_i ||y_{i,k} - \bar{y}_k||^2 \\ &= \frac{1}{N}\sum_i ||y_{i,k-1} - \bar{y}_{k-1} - \eta_c(\nabla F_i(y_{i,k-1};\xi_{i,k}) - c_i + \tilde{c}_i - \frac{1}{N}\sum_j \nabla F_j(y_{j,k-1};\xi_{j,k-1}))||^2 \\ &= \frac{\eta_c^2}{N}\sum_i ||\nabla F_i(y_{i,k-1};\xi_{i,k}) - c_i + \tilde{c}_i - \frac{1}{N}\sum_j \nabla F_j(y_{j,k-1};\xi_{j,k-1}))||^2 \\ &= \frac{\eta_c^2}{N}\sum_i ||S_i - c_i + \tilde{c}_i||^2\end{aligned}$$

  Comparing to the variance of SGD without correction of control variates

$$\begin{aligned}&\frac{1}{N}\sum_i ||y_{i,k} - \bar{y}_k||^2 \\ &= \frac{1}{N}\sum_i ||y_{i,k-1} - \bar{y}_{k-1} - \eta_c(\nabla F_i(y_{i,k-1};\xi_{i,k}) - \frac{1}{N}\sum_j \nabla F_j(y_{j,k-1};\xi_{j,k-1}))||^2 \\ &= \frac{\eta_c^2}{N}\sum_i ||\nabla F_i(y_{i,k-1};\xi_{i,k}) - \frac{1}{N}\sum_j \nabla F_j(y_{j,k-1};\xi_{j,k-1}))||^2 \\ &= \frac{\eta_c^2}{N}\sum_i |S_i||^2\end{aligned}$$

Since $c_i \approx \nabla F_i(y_{i,k-1}; \xi_{i,k-1})$ and $\tilde{c}_i \approx \frac{1}{N}\sum_j \nabla F_j(y_{j,k-1}; \xi_{j,k-1})$, $S_i \approx c_i - \tilde{c}_i$. That's then

$$\sum_i ||S_i - c_i + \tilde{c}_i||^2 - \sum_i ||S_i||^2 \approx -\sum_i ||c_i - \tilde{c}_i||^2 \leq 0.$$

It means that the distance achieved after local with correction is smaller than that without, which suggests the ability of variance-reduction method to reduce variance among workers. And also, there is another toy example of how consensus distance between decentralized SGD and decentralized SCAFFOLD.

4.1.3 Observation of variance out of toy example

So as to illustrate how approximation of c_i, $\nabla F_i(y_{i,k}; \xi_{i,k})$, and $\tilde{c}_i$, $\frac{1}{N}\sum_j \nabla F_j(y_{j,k}; \xi_{j,k})$ influences variance, define normalized deviation as $\Delta_1 := \frac{1}{N}\sum_i^N \frac{||c_i - \nabla F_i(y_{i,k};\xi_{i,k})||^2}{||\nabla F_i(y_{i,k};\xi_{i,k})||^2}$ and $\Delta_2 := \frac{1}{N}\sum_i^N \frac{||\tilde{c}_i - \frac{1}{N}\sum_j \nabla F_j(y_{j,k};\xi_{j,k})||^2}{||\frac{1}{N}\sum_j \nabla F_j(y_{j,k};\xi_{j,k})||^2}$. And following consider a toy example on how variance is reduced. Here consider using ResNet-8 on Cifar10 with 8 workers, and local steps $K = 245$, which is equivalent to 5 epochs of local datasets. Instead of the absolute but normalized deviation is considered to illustrate how close or how far the deviation in standard scale.

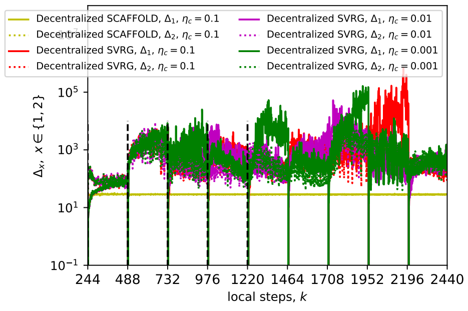


Figure 4.1: **The normalized averaged deviation between $c_i, \nabla F_i(y_{i,k}; \xi_{i,k})$ and $\tilde{c}_i, \frac{1}{N}\sum_j \nabla F_j(y_{j,k}; \xi_{j,k})$**: from previously intuitive analysis, within the reduction framework, the smaller the Δ_1 and Δ_2 is, the smaller the variance w.r.t every local update is.

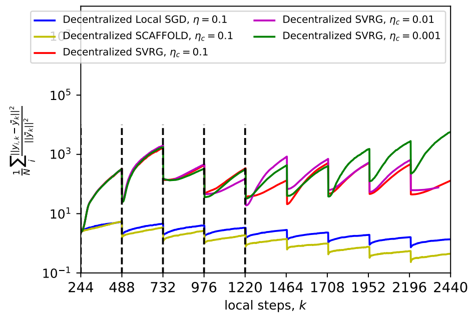


Figure 4.2: **The normalized averaged variance among workers for every local steps**: $\frac{1}{N}\frac{\sum_i ||y_{i,k} - \bar{y}_k||^2}{||\bar{y}_k||^2}$ using different methods and different learning rates. The smaller the value in y-axis, the smaller the variance. Comparing to Decentralized Local SGD, decentralized SCAFFOLD is able to reduce model variance among workers, decentralized variance reduction methods indeed help to reduce the variance.

Shown in Figure 4.1, decentralized SCAFFOLD could be able provide much better estimator to approximate $\nabla F_i(y_{i,k}; \xi_{i,k})$ and $\frac{1}{N}\sum_i \nabla F_i(y_{i,k}; \xi_{i,k})$, thus resulting smaller normalized client variance. Also the failure of D-SVRG is due to worse approximation, thus instead of correction to reduce variance but deviating direction and introducing more variance. And this phenomenon is independent of local model learning rate η_c when given same global model learning rate η_s.

Then in the following section introduces two different decentralized adaptations from their centralized origins within this variance-reduced framework. One is called decentralized SVRG from D-SVRG(21) and the other is decentralized SCAFFOLD from SCAFFOLD(6).

4.1.4 Decentralized SVRG

Decentralized SVRG is the direct adoptation from the centralized version D-SVRG(21). The naive thoughts behind is that replacing exact averaging with gossip averaging

$$x_i = \sum_j w_{ij} y_{j,K}, \quad c_i = \nabla f_i(x_i), \quad \tilde{c}_i = \sum_j w_{ij} c_j, \ w_{ij}$$

then only the neighbors $\{j : w_{ij} \neq 0\}$(including i itself) involved. If underlying topology is complete graph, then gossip averaging is exactly the same as exact averaging, and $x_i = x_j, \ \tilde{c}_i = \tilde{c}_j, \ \forall i \neq j$, thus decentralized SVRG is equivalent to centralized D-SVRG.

Within decentralize SVRG, almost the same as that of original SVRG in single machine, it calculates local full-batch gradient every outer loop, and it uses stochastic gradient w.r.t global model $\nabla F_i(x_i^r; \xi_{i,k})$ when modifying local model update, $v_{i,k}^r = \nabla F_i(y_{i,k}^r; \xi_{i,k}) - \nabla F_i(x_i^r; \xi_{i,k}) + \tilde{c}_i^r$. It needs an large amount of extra computation for this periodically calculated full-batch gradient comparing to decentralized Local SGD. In addition, for every global parameter update, since $\tilde{c}_i$ needs the global estimation of full-batch gradient w.r.t newly obtained model x_i, the operation of communication of $y_{i,K} \to x_i$ and $c_i = \nabla f_i(x_i) \to \tilde{c}_i$ is sequential. Then two times communication is needed for one outer loop.

4.1.5 Decentralized SCAFFOLD

SCAFFOLD(6) is a centralized SVRG-like method. Unlike D-SVRG, it provides option that needs no full-batch gradient calculation. It uses the precomputed stochastic gradient throughout local steps of last outer loop as the replacement of recalculated full-batch gradient at every outer loop in general variance-reduced framework

$$c_i^r = \frac{1}{K} \sum_k \nabla F_i(y_{i,k}^{r-1}; \xi_{i,k}).$$

The most promising property of SCAFFOLD comparing to D-SVRG to be adopted in decentralized setting is that it could support client sampling. That's, for every global model update and global control variate estimation, it's still workable with information from smaller part of workers. And this situation is similar to that in decentralized setting where every estimation of global parameter is through its neighbors, a fraction of all workers involved in system. The only difference is that in the centralized, no matter how sampling, there is only one sampling pattern at a time and only one global model maintained by server. But in the decentralized, the neighbors of each worker could be different, and every worker itself maintain different version of global model. Then in the decentralized training, do model update and estimate control variates as follow.

$$x_i^{r+1} = \sum_j^N w_{ij}(x_i^r + \eta_s(y_{i,K}^r - x_i^r))$$

$$c_i^{r+1} = c_i^r - \tilde{c}_i^r + \frac{1}{K\eta_c}(x_i^r - y_{i,K}^r)$$

$$\tilde{c}_i^{r+1} = \sum_j^N w_{ij}(\tilde{c}_i^r + c_i^{r+1} - c_i^r)$$

When underlying graph is complete, decentralized SCAFFOLD could be equivalent to the centralized. In addition, if initializing $c = \frac{1}{N}c_i$,

$$(\bar{\tilde{c}}^+,\ \bar{c}^+) = \frac{1}{N}\sum_i(\tilde{c}_i^r,\ c_i^r),\ \bar{\tilde{c}}^+ = \frac{1}{N}\sum_i \bar{c}^+$$

If annotating in matrix format, then

$$\begin{aligned}
\tilde{C}^{r+1} &= W(\tilde{C}^r + C^{r+1} - C^r) \\
&= W(W(\tilde{C}^{r-1} + C^r - C^{r-1})) + C^{r+1} - C^r) \\
&= W^{r+1}\tilde{C}^0 + \sum_{t=1}^{r} W^t(C^{r+1-t} - C^{r-t}) \\
&= WC^r + \underbrace{\sum_{t=1}^{r-1}(-W^t + W^{t+1})C^{r-t}}_{\Delta C^r} \quad \triangleright[W^{r+1}\tilde{C}^0 = W^r C^0]
\end{aligned}$$

The correction term ΔC^r within $\tilde{C}^{r+1} = WC^r + \Delta C^r$ of global control variate should be more stable comparing to decentralized SVRG($\tilde{C}^{r+1} = WC^r$), since it's made up of accumulated previous local control variates $C^{r-t}, t \in [1, r-1]$, working as momentum and the larger the number of communication rounds r, the smaller impact $-W^t + W^{t+1}$ for those C^t when $t << r$.

Note that there is a trade-off space between SCAFFOLD and SVRG regarding to the number of batches used to calculate the local control variate c_i. If define a pass over local data as an epoch, and B is the number of batches for mini-batch or sample points z of worker i w.r.t an epoch. Considering the different value of K, there will be following three different situation.

B : maximum number of samples per worker, or number of mini batches.
in decentralized SVRG, $c_i^+ = \frac{1}{|B|}\sum_z \nabla F(x_i^r; z) = \nabla f_i(x_i^r)$
in decentralized SCAFFOLD, $c_i^+ = \frac{1}{K}\sum_k \nabla F(y_{i,k}^{r-1}; \xi_{i,k})$.

- When $K = |B|$, the number of batches involved in the global control variate estimator c_i is the same for SVRG and SCAFFOLD.
- When $K > |B|$, SCAFFOLD is using more batches to calculate than SVRG.
- And when $K < |B|$, less.

Thus it means that SVRG could not be possible to use more batches than $|B_i|$ as that in SCAFFOLD. But since those gradient $\nabla F(y_{i,k}; \xi_{i,k})$ within SCAFFOLD is w.r.t stale model $y_{i,k}^{r-1}$ unlike x_i^r in SVRG, it rises a trade-off space to decide either using more batches with stale model or using newly updated model with less batches.

In addition, for SCAFFOLD, it has the same number of gradient computation as SGD before every gossip averaging. But since SVRG would recalculate the full-batch gradient w.r.t newly averaged model, it always needs another an epoch of extra computation for local full-batch gradient $c_i^{r+1} = \nabla f_i(x_i^{r+1})$ and one more communication for $\tilde{c}_i^{r+1} = \sum_j w_{ij} c_j^{r+1}$.

4.1.6 Convergence analysis of decentralized SCAFFOLD

Theorem 4.1. *For any L-smooth non-convex functions f, the output of decentralized SCAFFOLD has expected error smaller than ϵ for $\eta_s \geq 1$ and some values of $\eta_c \leq \frac{\rho^2}{500LB^2}$, R satisfying*

$$R = O\Big(\frac{\sigma^2}{\rho^2 K \epsilon^2} + \frac{\sigma^2}{KN\epsilon^2} + \frac{LG^2}{\rho^2 \epsilon^2} + \frac{B^2}{\rho^2 \epsilon}\Big) \cdot LF_0,$$

where $F_0 := f(x^0) - f(x^\star)$.

Table 4.1: Comparison of convergence rates for general non-convex functions.

	Methods	Communication rounds R	Assumptions
centralized	SCAFFOLD(6)	$O(\frac{\sigma^2}{KS\epsilon^2} + \frac{1}{\epsilon}(\frac{N}{S})^{\frac{2}{3}}) \cdot LF_0$	Ass. 3.4
decentralized	DSGD(9)	$O(\frac{\hat{\sigma}^2}{KN\epsilon^2} + \frac{G\sqrt{M+1}}{\rho\epsilon^{3/2}} + \frac{\hat{\sigma}^2}{\sqrt{\rho K}\epsilon^{3/2}} + \frac{\sqrt{(B^2+1)(M+1)}}{\rho\epsilon}) \cdot LF_0$	Ass. 3.5, 3.7
	Ours (Thm. 4.1)	$O(\frac{\sigma^2}{KN\epsilon^2} + \frac{\sigma^2}{\rho^2 K\epsilon^2} + \frac{G^2}{\rho^2\epsilon^2} + \frac{B^2}{\rho^2\epsilon}) \cdot LF_0$	Ass. 3.4, 3.7

Disscusion: Following is the discussion over each key components that could influences convergence performance.

- Both noise $\sigma^2 > 0$ and dissimilarity from heterogeneity $G^2 > 0$ dominates. We observer that the dominating term has linear speed up in number of workers N and number of local steps K. When $\sigma^2 = 0$, then the only the dissimilarity G resulted from heterogeneity dominates. When $\sigma^2 = 0$ and data partition is i.i.d, which is $G^2 = 0$, $B^2 = 0$, the convergence for decentralized SCAFFOLD is independent of topology, local steps, and number of workers. And the result is of the same order as that for decentralized Local SGD if $M = 0$.

- mixing rate ρ: when $\rho = 1$ at complete graph, decentralized SCAFFOLD is equivalent to the centralized. Since SCAFFOLD doesn't need any assumption on workers dissimilarity, thus the additional $O(\frac{G^2}{\epsilon^2} + \frac{B^2}{\epsilon})$ means decentralized SCAFFOLD needs more communication. But in special case where $B = 1, \ G = 0$, the convergence analysis is the same as that for centralized SCAFFOLD.

- noise $\sigma^2, \ \hat{\sigma}^2$: In the decentralized case comparing to SCAFFOLD, the underlying topology also influences how noise affects convergence. In particular, through the analysis in appendix, $O(\frac{\sigma^2}{\rho^2 K \epsilon^2})$ is resulted from the client direction approximation $\frac{1}{N}\sum_i ||c_i - \nabla f_i(\bar{x})||^2$ with reusing computed gradients. The smaller the mixing rate is, the noise has more power to break the convergence, which makes it not ideal for the optimization in the high-noise regime.

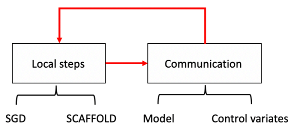
- dissimilarity $G^2, \ B^2$: Instead of the G^2 or B^2 individually, but $O(\frac{G^2}{\rho^2})$ and $O(\frac{B^2}{\rho^2})$ combined affects the communication performance w.r.t gradient similarity. It also suggests that the less connectivity of topology affects convergence via severe gradient dissimilarity. It is similar to case in decentralized Local SGD when $M = 0$(Ass. 3.5 decays to Ass. 3.4), where dissimilarity affects speed in term depending on topology in form of $O(\frac{G}{\rho})$ and $O(\frac{B}{\rho})$

- local steps K: All methods show linear speed up w.r.t local steps, and consider the noise term $O(\frac{\sigma^2}{\rho^2 K \epsilon^2})$ in decentralized SCAFFOLD. It shows that the doing even more local steps could help to remedy with same number of communication rounds when worse topology (smaller ρ).

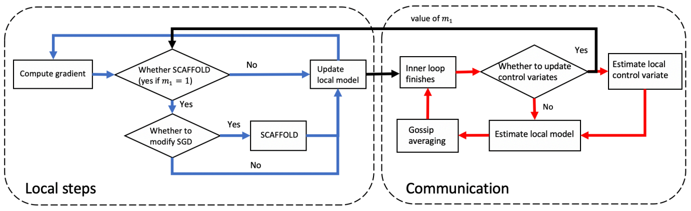
- number of workers N: N only affects the stochastic term for all methods in term of $O(\frac{\sigma^2}{KN\epsilon^2})$.

4.2 Decentralized SGD/SCAFFOLD training system

Comparing to decentralized Local SGD, decentralized SCAFFOLD uses the extra control variates to decrease the variance among workers, thus alleviate the client drift and accelerate. However, the modification to gradient brings extra computation and estimation of control variates brings extra communication burden(larger communication model). That's decentralized SCAFFOLD could do better in generlization performance and communication at the expense of both computation and communication. Then consider the typical decentralized training pipeline depicted in Figure 4.3a, which makes up with two components, Local steps and Communication.



(a) **system sketch**



(b) **training pipeline**

Figure 4.3: **Flow chart of decentralized SGD/SCAFFOLD training system**

In the following, "SCAFFOLD" is short to doing gradient modification with control variates in decentralized SCAFFOLD, and "SGD" short for local model update with sole SGD in decentralized Local SGD. We consider a deterministic activation pattern for each model update where doing either SGD or SCAFFOLD at current local step. $\beta_x,\ x \in \{1,2\}$ depicts the percentage of SCAFFOLD activation in different scenarios, and $L_x,\ x \in \{1,2\}$ the corresponding interval. Detailed explanation is in the following. We could tune the hyperparameters β_x and L_x to control the activation pattern for gradient $v_{i,k}^r$ when doing local model update

$$y_{i,k+1}^r = y_{i,k}^r - \eta_c v_{i,k}^r$$

.

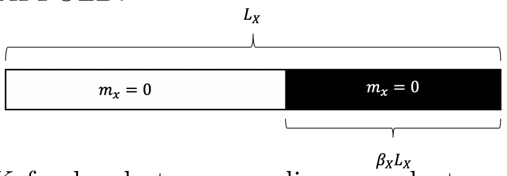
- **Outer Loops**. It's allowed to choose whether to activate SCAFFOLD within next outer loop, only then updating control variates $\{c_i^r,\ \tilde{c}_i^r\}$ at current communication. Assuming choice m_1, and it has only two choice within feasible set $m_1 \in \{0,1\}$. Use $m_1 = 1$ depicts event $\{$Update $c_i^r,\ \tilde{c}_i^r\}$, allowing SCAFFOLD for following local steps. $m_1 = 0$ is event $\{$Do nothing$\}$, doing naive SGD for following local steps. Then we define the percentage of event $\{m_1 = 1\}$ to be β_1. Then it clearly shows that the choice of $\{$Do nothing$\}$ makes just half-sized of information to communicate so as to be communication efficient comparing to standard decentralized SCAFFOLD system (from $(x_i, \tilde{c}_i)$ to (x_i)). And in addition, it could save extra computation of c_i^r and K times gradient modification in local steps ($v_{i,k}$ from $\nabla F_i(y_{i,k};\xi) - c_i + \tilde{c}_i$ to $\nabla F_i(y_{i,k};\xi)$)

$$\begin{aligned}
&m_1 \in \{0,1\},\ P(m_1) = 1\\
&m_1 = 1,\ \{\text{Update } c_i^r,\ \tilde{c}_i^r\}:\ c_i^{r+1},\ (\tilde{c}_i^{r+1},\ x_i^{r+1}) = \text{gossip}(\tilde{c}_i^r + c_i^{r+1} - c_i^r,\ y_{i,K}^r)\\
&m_1 = 0,\ \{\text{Do nothing}\}:\ (x_i^{r+1}) = \text{gossip}(y_{i,K}^r)
\end{aligned}$$

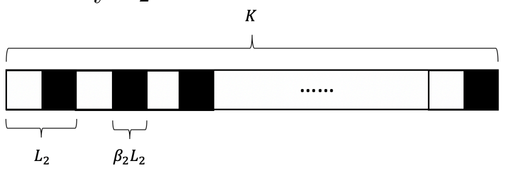
- **Local steps**. Only when $m_1 = 1$, then local steps is allowed to consider between SGD and SCAFFOLD. $v_{i,k}^r$ is the gradient used to update local model, $y^r_{i,k+1} = y^r_{i,k} - \eta_c v^r_{i,k}$, and it's allowed to decide whether to modify stochastic gradient, $\nabla F_i(y^r_{i,k}; \xi_{i,k})$. Given choice m_1, and similar to that in Local steps, $m_2 \in \{0, 1\}$. Use $m_2 = 1$ depicts the event {SCAFFOLD}, which is modifying gradient with control variates, then $m_2 = 0$ is the event {SGD}, which is using direct stochastic gradient computed. We define the percentage of m_2 to be β_2. Comparing to previous discussed standard decentralized SCAFFOLD system, the mixture choices over SGD and SCAFFOLD could be more computation efficient.

$$\begin{aligned}
&m_2 \in \{0, 1\},\ P(m_2 = 1|m_1 = 1) = \beta_1 \\
&m_2 = 1,\ \{\text{SCAFFOLD}\}:\ v^r_{i,k} = \nabla F_i(y^r_{i,k}; \xi_{i,k}) - c^r_i + \tilde{c}^r_i \\
&m_2 = 0,\ \{\text{SGD}\}:\ v^r_{i,k} = \nabla F_i(y^r_{i,k}; \xi_{i,k})
\end{aligned}$$

- **Implementation.** For the following experimental exploration, the activation is deterministic. Here describes the patterns. Following figures shows the overview of how it implements. L_x, $x \in \{1, 2\}$ indicates the length of specific block. L_x is the parameter to control the frequency. For example, when local steps sampling, if $L_2 = K$, then $\beta_2 K$ local steps would do gradient modification. For local steps sampling, the maximum value for L_2 is K, and for communication rounds sampling, L_1 could be to the maximum length of how many communication rounds planned for training, $L_1 \leq R$. The white block represents the event $m_x = 0$, and black for $m_x = 1$. Only the last $\beta_x L_x$ fraction of specific length does computation or communication needed for SCAFFOLD.



When $L_2 \leq K$ for local steps sampling, evenly truncate local steps into smaller block of length L_2, then within the smaller block, only the last $\beta_2 L_2$ does computation needed for SCAFFOLD. For example, if $\beta_2 = 0.5$, doing SCAFFOLD either every two rounds when $L_2 = 2$ or consecutive last five rounds every ten rounds $L_2 = 10$ both conform to the rules. And such frequency is defined by L_2.



- **Special cases.** Note that when $\beta = \beta_1\beta_2$ is the overall percentage of local model $y_{i,k}$ update using SCAFFOLD modification. Training procedure decays to standard decentralized Local SGD when $\beta = 0$, and to standard decentralized SCAFFOLD when $\beta = 1$,

Chapter 5

Benchmark Experiment

In this section, we conduct extensive experiment so as to investigate the effectiveness of decentralized SCAFFOLD when training model with and without normalization layer over heterogeneous data. And the benchmark metrics include both global model variance $\frac{1}{N}\sum_i ||x_i - \bar{x}||^2$ and top-1 test accuracy of globally averaged model $\bar{x}$.

1. Firstly, we show that global model variance in decentralized training would negatively influence model quality even in i.i.d setting, and decentralized SCAFFOLD is able to correct the drift. And this effect is consistent for models no matter with or without normalization.

2. Then we in detailed investigate the impact of different normalization techniques under heterogeneous data partition. It shows that the even though the performance of decentralized Local SGD changes along with different normalization techniques, decentralized SCAFFOLD could be able to show quite consistent performance, and always works much better than Local SGD.

3. Then through the overall comparison among decentralized Local SGD, decentralized SCAFFOLD and decentralized SVRG over both i.i.d and non-i.i.d data, different number of local steps, and models with or without normalization, decentralized SCAFFOLD could consistently outperforms the rest in terms of speed and generalization performance. It's almost 2-3 times faster than decentralized Local SGD when the number of local steps is large.

4. At last, we build a decentralized SCAFFOLD training system, where it could iteratively choose between SGD and SCAFFOLD so as to in addition decrease the computation overhead. It shows that the system has the great potential of using only few steps of SCAFFOLD to remedy the client drift and to accelerate on top of SGD, which embraces both the least computation and the least communication overhead.

5.1 Detailed Experimental Setup

Task We empirically study the decentralized training behavior on image classification task for CIFAR-10(43) via ResNet-8(44) and Lenet(45), and only consider ring topology. For later discussion over the impact of normalization layer, ResNet-8 is the typical example for modern neural network using normalization technique and Lenet for general non-convex function without normalization.

Local training scheme

Table 5.1: **Hyperparameters overview**

Hyper-parameters in decentralized SCAFFOLD, SVRG and Local SGD				
algorithm	**learning rate**		**workers**	**local steps**
decentralized SCAFFOLD	$\eta_s: \ x_i = x_i - \eta_s(y_{i,K} - x_i)$	$\eta_c: \ y_{i,k} = y_{i,k-1} - \eta_c v_{i,k-1}$	N	K
decentralized SVRG				
decentralized Local SGD	$\eta: \ y_{i,k} = y_{i,k-1} - \eta v_{i,k-1}$			

For the sake of simplicity, In particular, we only tune the learning rate on SGD on single machine over range $\{1e-1, \ 1e-2, \ 1e-3, \ 1e-4, \ 1e-5\}$ for both ResNet-8 and LeNet. And the best top-1 accuracy obtained on single machine after 1000 epochs, is 85.45% for ResNet-8 and 74.21% for LeNet respectively.

1. All learning rates involved are constant without decay. Both ResNet-8 and LeNet are using the best learning rate $\eta_c = 0.1$ tuned on single machine case and the learning rate for model update is fixed $\eta_s = 1$ for decentralized SCAFFOLD and decentralized SVRG, and $\eta = 0.1$ for decentralized Local SGD.

2. No Nesterov momentum acceleration and no weight decay.

3. Batch size for each worker is set to be 128. That's $|B_i| = 49$ batches/epoch if 8 workers, or $|B_i| = 25$ batches/epoch if 16 workers during training. Dataset is evenly partitioned by N workers, which guarantees every worker shares the same number of batches per epoch.

4. Unless mentioned otherwise, the number of workers is set to $N = 8$.

5. Generalization performance is evaluated on global averaged model $\bar{x} = \frac{1}{N}\sum_i x_i$, using top-1 accuracy. Beside, variance between workers and global average is another important index to consider the individual quality of each worker.

Heterogeneous data simulation The heterogeneity considered here talks about the different label distribution $P(z|i)$ (z: label; i: worker index), among workers in train set, which is also the setting considered by most prior work in this domain. Such distribution-based label imbalance could be simulated by allocating only propotion of the samples of each label according to Dirichlet distribution. Then the concerntratin parameter $\alpha > 0$ for Dirichlet distribution controls the degree of heterogeneity: $\alpha = 1000$ approximates the homogeneous local data distribution, and the smaller α is, the more skewed the distribution $P(z|i)$, $\forall i \in [n]$ is. An example of such partitioning is shown in Figure 5.1. And the skewed label distribution is common whenever data are generated from different users or device. Some bias within users' preference will easily result in the skewness in label.

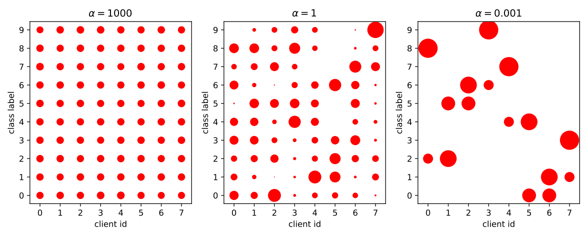


Figure 5.1: **Illustration of # of samples per class allocated to 8 workers** (indicated by dot size), for different α values of Dirichlet distribution.

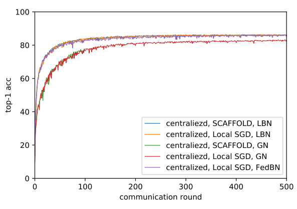
For later detailed implementation, we consider $\alpha = 1000$ for representative of the i.i.d and only could be able to consider the smallest degree $\alpha = 1$ as representative of non-i.i.d case since the case $\alpha = 0.001$ is too extreme to be considered in general federated learning, where each client have only few fraction of class label. Probably, personalized federated learning(46) is more suitable, which is out of scope of this work. For evaluation, the performance of each algorithm is evaluated by globally averaged model $\bar{x} = \frac{1}{N}\sum_i^N x_i$ on single test set.

5.2 Variance reduction in i.i.d setting

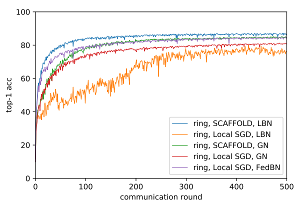
In Figure 5.2 and Figure 5.3 show the result of distributed training of ResNet-8 and Lenet in the centralized and decentralized. It includes both the generalization performance and global model variance. Since ResNet-8 has normalization layer, in addition it considers both LBN and GN. The degree of non-i.i.dness is set to be $\alpha = 1000$, which is actually i.i.d. $N = 16$ workers, and Local steps $K = 250$ which is equivalent to 10 epochs are used. Decentralized training here uses only ring topology. For simplicity, constant and the same learning rate is used for both centralized and decentralized training. And the specific value is explained in section 5.1.

Client drift also happens for i.i.d cases in decentralized training. When number of local steps is large, ResNet-8 in Figure 5.2b and LeNet in Figure 5.3a converge to suboptimum comparing to the centralized cases on ring topology. It results from the inexact communication of gossip averaging for every model update, comparing to truly exact communication in the centralized case. Then even though i.i.d data partition in the first place, during model training procedure, worker deviates greatly from each other when local steps is large, and this deviation could not be perfectly compensated by later model aggregating, resulting in global model variance $\frac{1}{N}\sum_i ||x_i - \bar{x}||^2 > 0$. Then throughout step by step accumulation, local models trained by SGD starts to drift, and more severe for model with normalization.

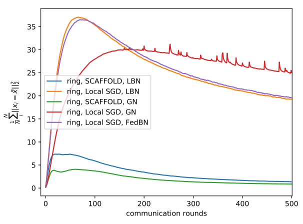
Variance reduction could reduce model variance to alleviate client drift. From Figure 5.2c, non-negligible gap between Local SGD and SCAFFOLD shows that the success of SCAFFOLD against client drift from its ability to control variance for every model update, which is also the variance among workers x_i w.r.t $\bar{x}$. And for SCAFFOLD, using GN or LBN doesn't make much difference; for Local SGD, the variance for LBN is better than GN even though LBN suffers more from client drift in terms of accuracy.



(a) **Centralized Resnet-8**

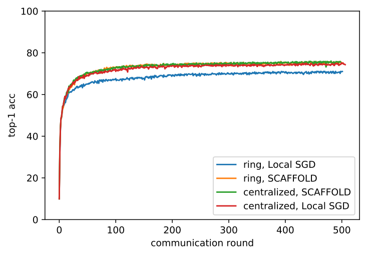


(b) **ResNet-8 on ring**

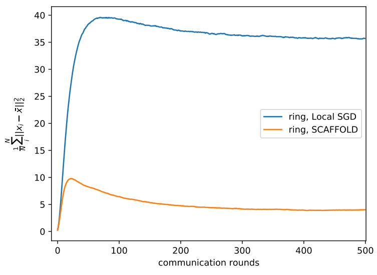


(c) **Model variance**

Figure 5.2: **Distributed learning of Resnet-8 in the both centralized and decentralized**. GN here uses 2 groups/channel for normalization.



(a) **Centralized and decentralized LeNet**



(b) **Global model variance**

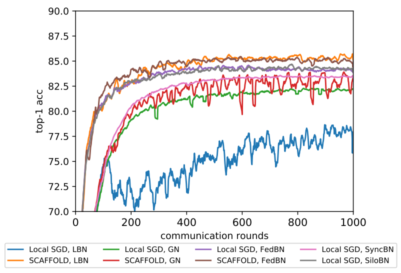
Figure 5.3: **Distributed learning of Lenet in the both centralized and decentralized**

5.3 Normalization Techniques Comparison

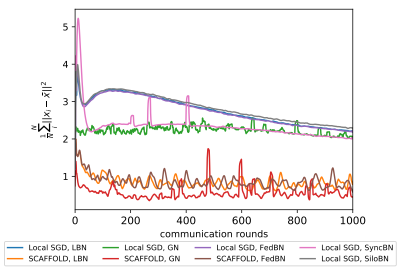
In this section mainly discusses the effect of different normalization techniques so as to investigate how those models with normalization layer are affected. In the following experiments, decentralized training is performed over ring topology and $N = 8$ workers is used. The degree of non-i.i.dness is $\alpha = 1$. And local steps is $K = 245$, which is equivalent to 5 epochs.

For the normalization techniques, LBN, SyncBN as synchronized BN, FedBN as asynchronized BN, and GN as instance-based normalication. In FedBN, throughout training procedure, affine parameters $(\gamma_{BN})_i, (\beta_{BN})_i$ and running estimates are kept local without any aggregating. But for evaluation, since I evaluate on the globally averaged model $\bar{x}$, I test with globally averaging affine parameters $\bar{\gamma}_{BN} = \frac{1}{N}\sum_i(\gamma_{BN})_i$, $\bar{\beta}_{BN} = \frac{1}{N}\sum_i(\beta_{BN})_i$, and globally averaged running estimates $\bar{\mu} = \frac{1}{N}\sum_i \bar{\mu}_i$, $\bar{\sigma}^2 = \frac{1}{N}\sum_i \bar{\sigma}_i^2$. Beside the generalization performance of $\bar{x}$, variance between workers and global average is another important index to consider the quality of each worker.

Normalized by heterogenous distributions doesn't have negative impact. From Figure 5.4a, consider the artificial scenario in SyncBN where every worker would have global communication for batch statistics and gradient of input every forward and backward calculation. It makes sure that every worker is normalized by the same underlying distribution. Similarly to SyncBN, GN is instance-based, then they both have the ability to decorrelate from heterogeneous normalization. On the contrary, LBN is only partially synchronized. It's normalized with local batch statistics, but only averages the affine parameters and running estimates. Even further, FedBN is completely asynchronized w.r.t normalization. It not only is normalizated with local batch statistics, but also keeps affine parameters and running estimates local. For both LBN and FedBN, their normalization is highly correlated to heterogeneous local distribution. But their performance are very similar given a particular optimizer no mater how normalizing. Since the test top-1 accuracy is measured by the performance of $\bar{x}$, the larger the converged variance means the larger the deviation for x_i from $\bar{x}$. Even the artificial global model $\bar{x}$ reaches the optimum, model maintained by each worker is still not good enough at this time. If using more workers or larger local steps, the variance would also increase accordingly and it makes the training procedure delicate. And among them all, FedBN could even be benefit from the completely local operation and outperforms the rest in terms of accuracy, and decentralized SCAFFOLD could consistently outperforms decentralized Local SGD in terms of both accuracy and variance.



(a) **Test top-1 accuracy**



(b) **Global model variance**

Figure 5.4: **Decetralized learning of ResNet-8** with different normalization techniques

5.4 Overall comparison

In this section, I evaluates the performance of decentralized SCAFFOLD, decentralized Local SGD and decentralized SVRG in terms of generalization performance, communication rounds, variance, and computation performance.

5.4.1 Generalization performance and communication rounds

Table 5.2 and Table 5.3 summarize the results for various degrees of non-i.i.d data, local training steps and models on CIFAR-10.

Decentralized SCAFFOLD has even better generalization performance. Considering the standard situation in single machine case, the highest accuracy reached by SGD given the best learning is 84%. Decentralized Local SGD suffers a bit from client drift that it couldn't reach 84% and the larger the number of steps is, the more it suffers from quality loss. But decentralized SCAFFOLD not only could be able to be against client drift, but also reach a better optimum.

It's possible and reasonable, since it's been detected that decentralized setting provides more variance in the exploration phase and thus ends up with better variance reduction method, SVRG, could only work when number of local steps K is small, and when K is larger, such as $K \geq 1$ epoch, both communication and accuracy performance are hurt and suffer from larger variance shown in Figure 5.5a, instead of reducing variance. Contrary to SCAFFOLD benefiting from larger number of local steps, SVRG would either crashes for models without normalization or be hard to converge for models with normalization. However, since only constant learning rate is used here for SVRG, tuning learning rate could help training procedure more stable and reach a better quality when large K(e.g, $K \geq 5$ epoch).

Decentralized SCAFFOLD could accelerate in either heterogeneity or homogeneity when large number of local steps. Like what discussed in section 4.3.2 over the choice of K, which regards $K = 1$ epoch as the critical point. The performance of SCAFFOLD greatly depends on the number of K. Not only because if K is too large, the discrepancy among workers is too large to be compensated by later model aggregating, but also K also decides how many batches will be used to estimate control variates, $c_i = \frac{1}{K}\sum_k \nabla F(y_{i,k}; \xi_{i,k})$. When $K > 1$ epoch, SCAFFOLD is using more than an epoch of batches to estimates the control variates. Thus there arises a trade-off space over choice of K. However, also because the correction of control variates, comparing to Local SGD, SCAFFOLD has stronger ability to tolerate much larger K, thus being able to take the advantage of large local steps and achieve acceleration in terms of accuracy in communication round in Table 5.3 regardless of partition.

If centralized, the performance of SCAFFOLD is equivalent to its original centralized version and decentralized Local SGD to FedAvg(47). Their performance has been extensively explored by (18), and on the contrary, the consistently significant improvement in decentralized SCAFFOLD over decentralized Local SGD is not always the case.

The accuracy recorded in Table 5.2 is the highest accuracy reached within 1000 communication rounds. For SVRG, twice communication is needed for each outer loop, one for model gossip, the other for gradient gossip. But for Local SGD and SCAFFOLD, only once is needed. Then 1000 communication rounds is equivalent to 1000 outer loops of training for Local SGD and SCAFFOLD, and 500 outer loops for SVRG. And "/" in table means this experiment isn't implemented; "nan" means, under current parameter and training scheme, the training procedure crashes and loss diverges to NAN.

Table 5.2: **Test top-1 accuracy of different decentralized methods in different scenarios** (i.e. different non i.i.d degrees, # of local steps, and models.) on CIFAR-10 after 1000 communication rounds for 8 workers. Then 1000 communication rounds is equivalent to 1000 outer loops of training for Local SGD and SCAFFOLD, and 500 outer loops for SVRG. ResNet-8 uses LBN for normalization layer. 1 epoch here is equivalent to 49 steps. The accuracy recorded in Table 5.2 is the highest accuracy reached within 1000 communication rounds. And "/" in table means this experiment isn't implemented; "nan" means, under current parameter and training scheme, the training procedure crashes and loss diverges to NAN. And the best top-1 accuracy obtained on single machine after 1000 epochs, is 85.45% for ResNet-8 and 74.21% for LeNet respectively.

The test top-1 accuracy of different reached before 1000 communication rounds.					
Models	partitioning	local steps	Local SGD	SCAFFOLD	SVRG
ResNet-8	$\alpha = 1$	1	**52.81**	51.92	51.12
		0.5 epoch	81.2	**81.82**	79.45
		1 epoch	83.41	**83.88**	72.30
		5 epochs	83.09	**85.59**	18.28
		20 epochs	79	**83.75**	/
	$\alpha = 1000$	1 epoch	83	**85.31**	84.89
		20 epochs	83	**86**	/
LeNet	$\alpha = 1$	1 epoch	**71.06**	70.41	nan
		20 epochs	71.12	**73.5**	/

Table 5.3: **Number of communication rounds** first reaching 70% test top-1 accuracy for ResNet-8, and 60% for LeNet. ResNet-8 uses LBN for normalization layer. For SVRG, communication rounds counts separated model gossip and gradient gossip differently, which means 300 communication rounds recorded is equivalent to 150 outer loops of training. ∞ means that it could not reach target accuracy given current scenario. And "/" in table means this experiment isn't implemented. 1 epoch here is equivalent to 49 steps.

The number of communciation rounds to reach target performance T					
Models	partitioning	local steps	Local SGD	SCAFFOLD	SVRG
ResNet-8 $T = 70\%$	$\alpha = 1$	1	4434	4189	**4091**
		0.5 epoch	**186**	300	2880
		1 epoch	**116**	135	3349
		5 epochs	62	**30**	∞
		20 epochs	57	**19**	/
	$\alpha = 1000$	1 epoch	73	73	234
		20 epochs	30	**10**	/
LeNet, $T = 60\%$	$\alpha = 1$	1 epoch	**68**	86	nan
		20 epochs	13	**6**	/

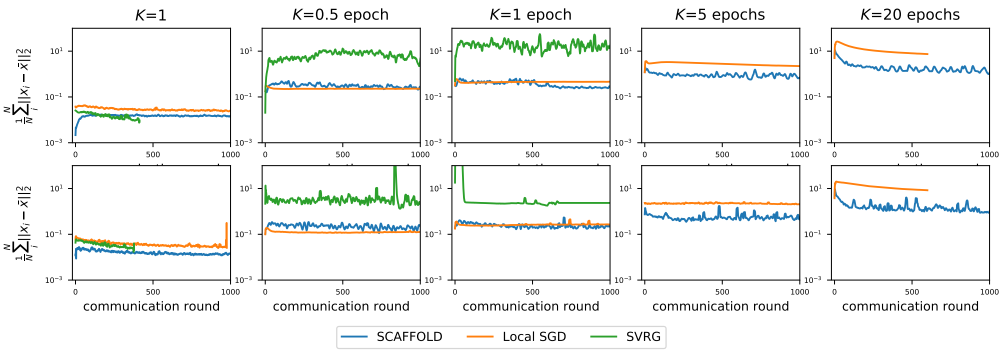
5.4.2 Variance and effect of local steps

In this section, after the discussion over generalization performance and communication rounds, here evaluates the variance of global model for each methods. Considering the SVRG performance in Table 5.2 and Table 5.3, cases where $K = 5$ epochs and $K = 20$ epochs are not considered particularly for SVRG, since their training procedures would break due to too many local steps.

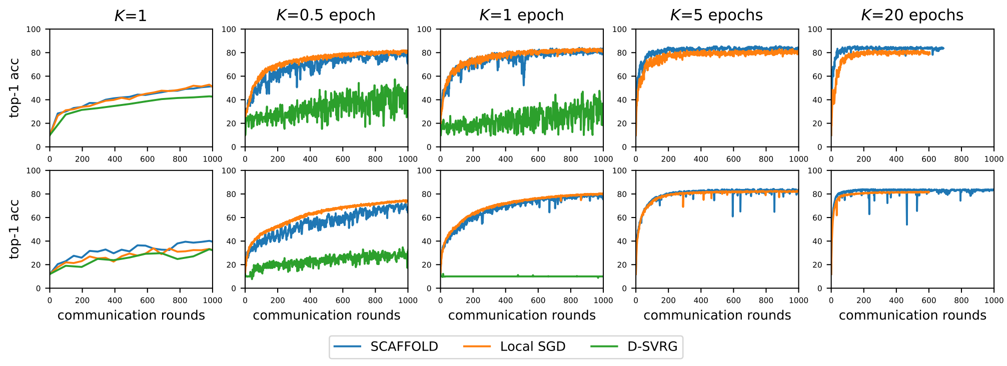
Performance of model variance among workers worsens along with number of local steps. From Figure 5.5a, the variance for all methods increases when using more local steps. And the impact of local steps is more significant for Local SGD than that for SCAFFOLD. And comparing to Figure 5.5b, where using more local steps brings significant acceleration, meanwhile the number of communication rounds needed for variance to converge also increases. Thus it also indicates the trade-off space over local steps about acceleration in global model training or in convergence of smaller variance.

SVRG fails to reduce but to introduce more variance. It has been proposed that the performance of SVRG even on single machine on models with normalization layer doesn't help in acceleration as it's proven in other non-convex functions(33). Rather than reducing variance, it introduces variance to modified gradient v_k^r due to normalization. The situation is alike in decentralized setting.

From Figure 5.5a, in general, with increasing number of local steps K, the converged variance for all methods are increasing accordingly. But SVRG suffers more. The relative gap in variance between SVRG and the rest is also increased along with K, which suggest larger variance among workers, thus making SVRG converge much slower, and the more unstable $\bar{x}$, suggested by the vibration in generalization performance. And detailed discussion over failure of SVRG is in later section 5.4.4.



(a) **Global model variance**



(b) **Communication performance**

Figure 5.5: **Different decentralized variance reduction methods on ResNet-8** over ring given 1000 communication rounds when using different number of local steps in terms of (a) global model variance and (b) communication rounds. **First row:** Resnet-8 with LBN; **second row:** ResNet-8 with GN, and num of groups/channel $= 2$. Here 8 workers are being used, and data partition is non-i.i.d, $\alpha = 1$.

5.4.3 Computation performance

Decentralized SCAFFOLD performs consistently across different choice of local steps w.r.t computation. From Figure 5.6, no matter using GN or LBN, the curve for SCAFFOLD shows consistency when using different local steps, even though when K is so large, e.g, $K = 20$ epoch, then the performance of SCAFFOLD and Local SGD is deteriorated a bit. However, SCAFFOLD even shows stronger ability than Local SGD against this source of deviation. Local SGD is more vulnerable to local steps.

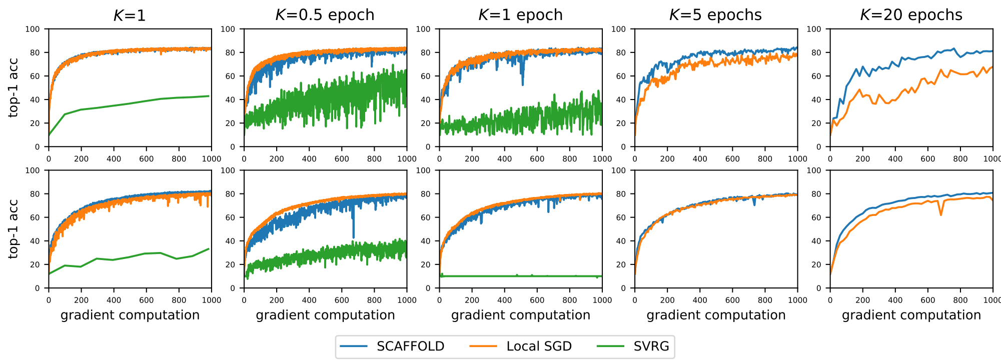
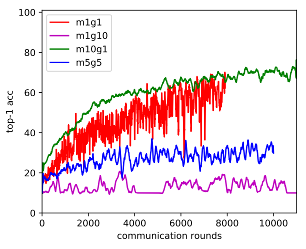


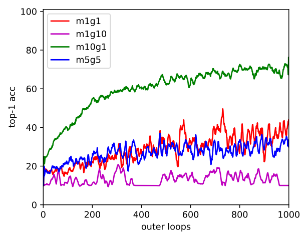
Figure 5.6: **Computation performance for variance reduction methods on ResNet-8** over ring given 1000 batches of computation. Here considers 8 workers and $\alpha = 1$. Index in x-axis stands for number of times passing over local dataset, or $|E|$. For SVRG, it calculates extra full-batch gradient for every outer loop, thus for before every communication, it needs $K + |E|$ batches of computation, while Local SGD and SCAFFOLD need K batches. $|E|$: # of batches for an epoch.

5.4.4 Failure of SVRG

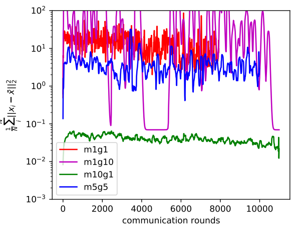
In this section, we mainly investigate why SVRG fails in reducing variance.



(a) **Decentralized baseline on Resnet-8** with LBN and local steps $K = 1$.



(b) **Generalization performance with different number of local steps**



(c) **Global model variance with different number of local steps**

Figure 5.7: **Impact of gossip round in model or global control variate within SVRG** on ResNet-8 with LBN over ring topology and non-iid data partition, $\alpha = 1$. Here local steps $K = 1$ epoch is used for all implementation. "**m1g1**": standard SVRG with 1 gossip for model, x_i^r then 1 gossip for global control variate, $\tilde{c}_i^r$, which counts 2 communication rounds performed for every outer loop. "**mxgy**": x times gossip for x_i^r, y times gossip for $\tilde{c}_i^r$, and $x+y$ communication rounds counted for every outer loop.

Consider the approximation performance of control variates and client variance w.r.t every local steps. $\Delta_1 := \frac{1}{N}\sum_i^N \frac{||c_i - \nabla F_i(y_{i,k};\xi_{i,k})||^2}{||\nabla F_i(y_{i,k};\xi_{i,k})||^2}$ and $\Delta_2 := \frac{1}{N}\sum_i^N \frac{||\tilde{c}_i - \frac{1}{N}\sum_j \nabla F_j(y_{j,k};\xi_{j,k})||^2}{||\frac{1}{N}\sum_j \nabla F_j(y_{j,k};\xi_{j,k})||^2}$

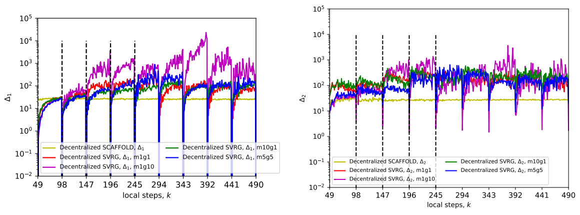


Figure 5.8: **The normalized averaged deviation between** $c_i, \nabla F_i(y_{i,k};\xi_{i,k})$ **and** $\tilde{c}_i, \frac{1}{N}\sum_j \nabla F_j(y_{j,k};\xi_{j,k})$: from previously intuitive analysis, within the reduction framework, the smaller the Δ_1 and Δ_2 is, the smaller the variance w.r.t every local update is.

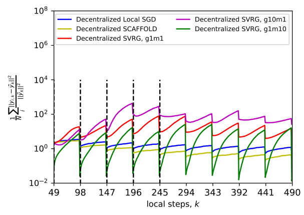


Figure 5.9: **The normalized averaged variance among workers for every local steps**: $\frac{1}{N}\frac{\sum_i ||y_{i,k} - \bar{y}_k||^2}{||\bar{y}_k||^2}$ using different methods and the same learning rates. The larger the variance, combined with Figure 5.7a, the worse the model quality in $\bar{x}$.

Correction with full-batch gradient doesn't help to reduce variance. In Figure 5.7, we consider different gossip mechanism in hope of improving performance, but they all fails to compete with decentralized SCAFFOLD. As previously discussed in section 4.1.2, variance reduction methods is only able to reduce variance if $c_i \approx \nabla F_i(y_{i,k}; \xi_{i,k})$ and $\tilde{c}_i \approx \frac{1}{N}\sum_i \nabla F_i(y_{i,k}; \xi_{i,k})$. Shown in Figure 5.8, all D-SVRG scenarios using full-batch gradient as estimator of control variates has much larger deviation in direction approximation comparing to decentralized SCAFFOLD. And instead of reducing variance, during local steps, D-SVRG keeps introducing variance. Even having more exact averaging over control variates(g10m1 in Figure 5.8) doesn't help to improve in approximation of $\nabla F_i(y_{i,k}; \xi_{i,k})$ and $\frac{1}{N}\sum_i \nabla F_i(y_{i,k}; \xi_{i,k})$ but deviates even further, thus resulting growing variance and breaking convergence in Figure 5.7a. At the same time, gossiping more over model $X = Y_0$ itself could help reduce variance among starting point $||Y_0 - \bar{Y}_0||_F^2$ for next round inner loop. Then even though the approximation is still unsatisfactory at this time and keeps introducing variance, but if using $K > 49$(49 is equivalent to 1 epoch of local dataset), it has great possibility to be negatively affected.

5.5 Decentralized SGD/SCAFFOLD training system

This section is conducted so as to investigate how interpolating SCAFFOLD into SGD could help to be against client drift and accelerate. For the following experimental exploration, consider the use case where heterogeneous data partition, $\alpha = 1$, and decentralized training of ResNet-8 with LBN on Cifar10 over ring topology. And every worker does $K = 49$ of local steps before communication. The main purpose of those experimental is to investigate how performance changes with activation pattern, and how parameters influences, and at last to give a comprehensive user guide. The main findings are as follow before detailed implementation:

1. Both communication and generalization performance is much better comparing to β_0 standard decentralized Local SGD, even though it is negatively influenced by partial activation comparing to $\beta = 1$ standard decentralized SCAFFOLD.
2. No matter activating local steps or epochs, interpolating SCAFFOLD steps within Local SGD could always realize acceleration and reaches better performance in both variance and generalization performance, in Table 5.4 and Tabel 5.5
    - Activating local steps could always works and end up with stable and reasonable result. And even only small fraction of steps doing SCAFFOLD is enough to significantly improve the performance of Local SGD.

- Activating outer loops is more risky and needs careful tuning over frequency L_1. When β_1 is small, it has great tendency to diverge or shows extremely unstable performance. That's the most frequent, the most unstable. When choosing the length of interval L_1, it's safer to use relatively large number.

3. When activating both local steps and outer loops in Table 5.12, even though given $\beta = 0.1$, only situation where larger β_1 (e.g, $\beta_1 = 0.5$) could work. The percentage of activated outer loops plays a more important role than local steps.

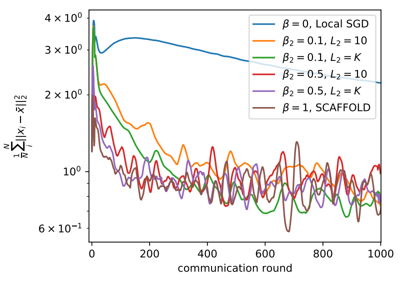
4. The frequency is important. The performance starts to be deteriorated when the more frequent either in local steps or epochs.

5.5.1 Local steps activation

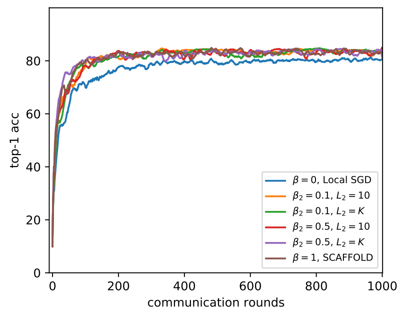
In this section, consider the situation where SCAFFOLD is activated for every outer loop, which is given $m_1 = 1$, and only consider local steps activation. choosing $m_2 \in \{0, 1\}$. β_2 indicates the percentage among local steps doing SCAFFOLD modification and through changing the length of interval L_2 could control the frequency of SCAFFOLD steps.

Table 5.4: **Communication rounds and Top-1 acc:** The number of communication rounds needed to reach top-1 accuracy of 70%, and the maximum top-1 accuracy reached within 1000 communication rounds.

β_2	L_2	Communication rounds	Top-1 acc (%)
0 (SGD)		62	83.09
0.1	K	40	85.38
	10	42	85.47
0.5	K	40	85.33
	10	39	85.44
1 (SCAFFOLD)		**30**	**85.59**



(a) **Global model variance**



(b) **Top-1 acc**

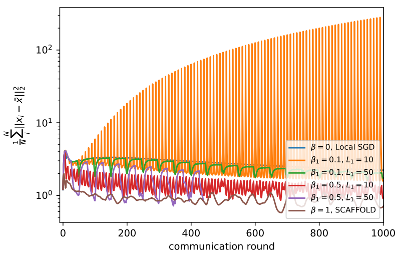
Figure 5.10: **Decetralized learning of ResNet-8 with local steps activating**

5.5.2 Outer Loops activation

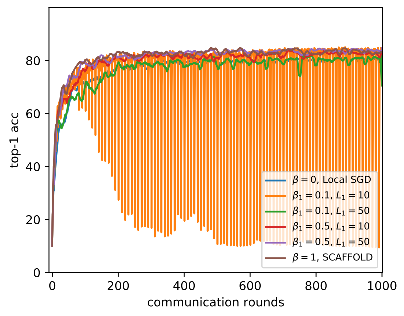
In this section, consider the situation where activating for outer loop enabling SCAFFOLD, choosing $m_1 \in \{0, 1\}$, and within the enabled outer loop, every local step does SCAFFOLD modification, which is given $m_2 = 1$.

Table 5.5: **Communication rounds and Top-1 acc:** The number of communication rounds needed to reach top-1 accuracy of 70%, and the maximum top-1 accuracy reached within 1000 communication rounds.

β_1	L_1	Communication rounds	Top-1 acc (%)
0 (SGD)		62	83.09
0.1	10	40	85.05
	50	50	82.32
0.5	2	40	68.51
	10	42	85.08
	50	32	85.38
1 (SCAFFOLD)		**30**	**85.59**



(a) **Global model variance**



(b) **Top-1 accuracy**

Figure 5.11: **Decetralized learning of ResNet-8 with outer loops activating**

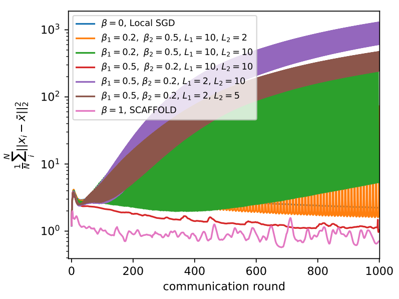
5.5.3 Activating both Outer Loops and Local steps

In this section, consider the situation where activating for both outer loop and its local steps enabling SCAFFOLD, choosing $(m_1, m_2) \in \{0, 1\} \times \{0, 1\}$.

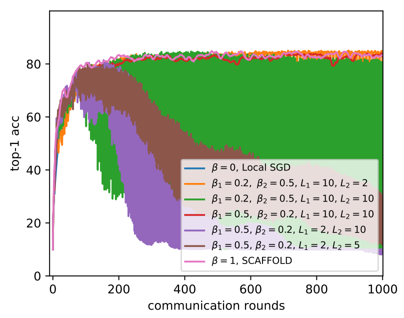
Table 5.6: **Communication rounds and Top-1 acc:** The number of communication rounds needed to reach top-1 accuracy of 70%, and the maximum top-1 accuracy reached within 1000 communication rounds.

β	β_1	β_2	L_1	L_2	Communication rounds	Top-1 acc (%)
0 (SGD)					62	83.09
0.1	0.2	0.5	10	2	69	85.08
			10		56	84.71
	0.5	0.2	10		40	85.07
			2	10	39	81.47
			2	5	55	79.93
1 (SCAFFOLD)					**30**	**85.59**

Figure 5.12: **Decetralized learning of ResNet-8 with both local steps and outer loops activating**, but keeping the overall percentage of local model updates $\beta = \beta_1\beta_2 = 0.1$ constant, with different combination of β_1 and β_2 and using L_1 and L_2 to control the length of interval thus frequency.



(a) **Global model variance**



(b) **Top-1 accuracy**

Chapter 6

Conclusion

In this thesis, we have proposed a novel decentralized variance-reduced methods, decentralized SCAFFOLD over undirected graphs on heterogeneous that achieve acceleration in terms of communication rounds for general non-convex functions and also deep neural netowrk with normalization layer. It could also be able to remedy client drift resulted from heterogeneous data and reaches much better model quality than standard dencentralized Local SGD. We have further proposed a novel decentralized training framework with mixture of SGD and SCAFFOLD. It shows that very few steps of correction within decentralized SCAFFOLD could be already enough to reduce variance, remedy client drift and accelerate at the same time, thus resulting in the least communication and computation overhead.

Bibliography

[1] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: training imagenet in 1 hour," *CoRR*, vol. abs/1706.02677, 2017. [Online]. Available: http://arxiv.org/abs/1706.02677

[2] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "Imagenet training in minutes," in *Proceedings of the 47th International Conference on Parallel Processing*, ser. ICPP 2018. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3225058.3225069

[3] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl, "Measuring the effects of data parallelism on neural network training," *Journal of Machine Learning Research*, vol. 20, no. 112, pp. 1–49, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-789.html

[4] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/1ff1de774005f8da13f42943881c655f-Paper.pdf

[5] 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," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5336–5346.

[6] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for on-device federated learning," *Technical Report*, 2019. [Online]. Available: https://arxiv.org/abs/1910.06378

[7] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal,

M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," 2021.

[8] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," 2021.

[9] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A unified theory of decentralized SGD with changing topology and local updates," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 13–18 Jul 2020, pp. 5381–5393.

[10] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "d^2: Decentralized training over decentralized data," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4848–4856. [Online]. Available: http://proceedings.mlr.press/v80/tang18a.html

[11] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 3478–3487. [Online]. Available: http://proceedings.mlr.press/v97/koloskova19a.html

[12] 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," 2017.

[13] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/ac1dd209cbcc5e5d1c6e28598e8cbbe8-Paper.pdf

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 37. PMLR, 07–09 Jul 2015, pp. 448–456.

[15] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine

Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 4387–4398.

[16] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018.

[17] A. Khaled, K. Mishchenko, and P. Richtarik, "Tighter theory for local sgd on identical and heterogeneous data," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 4519–4529.

[18] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," 2021.

[19] A. Defazio, F. R. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," *CoRR*, vol. abs/1407.0202, 2014. [Online]. Available: http://arxiv.org/abs/1407.0202

[20] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, "SARAH: A novel method for machine learning problems using stochastic recursive gradient," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 2613–2621.

[21] S. Cen, H. Zhang, Y. Chi, W. Chen, and T.-Y. Liu, "Convergence of distributed stochastic variance reduced methods without sampling extra data," *IEEE Transactions on Signal Processing*, vol. PP, pp. 1–1, 06 2020.

[22] S. De and T. Goldstein, "Efficient distributed sgd with variance reduction," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 111–120.

[23] J. Konečný, H. B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," 2015. [Online]. Available: http://arxiv.org/pdf/1511.03575v1.pdf

[24] B. Li, S. Cen, Y. Chen, and Y. Chi, "Communication-efficient distributed optimization in networks with gradient tracking and variance reduction," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 1662–1672. [Online]. Available: http://proceedings.mlr.press/v108/li20f.html

[25] R. Xin, U. A. Khan, and S. Kar, "Variance-reduced decentralized stochastic optimization with accelerated convergence," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6255–6271, 2020.

[26] A. Gerbessiotis and L. Valiant, "Direct bulk-synchronous parallel algorithms," *Journal of Parallel and Distributed Computing*, vol. 22, no. 2, pp. 251–267, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731584710859

[27] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA: USENIX Association, Mar. 2017, pp. 629–647. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/hsieh

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[29] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 2351–2363. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/18df51b97ccd68128e994804f3eccc87-Paper.pdf

[30] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local sgd," in *International Conference on Learning Representations*, 2020.

[31] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=TNkPBBYFkXg

[32] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1000–1008.

[33] A. Defazio and L. Bottou, "On the ineffectiveness of variance reduced optimization for deep learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[34] S. U. Stich, "Local sgd converges fast and communicates little," 2019.

[35] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "Megdet: A large mini-batch object detector," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6181–6189.

[36] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting batch normalization for practical domain adaptation," 2016.

[37] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han, "Domain-specific batch normalization for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[38] M. Andreux, J. O. du Terrail, C. Beguier, and E. W. Tramel, "Siloed federated learning for multi-centric histopathology datasets," 2020.

[39] X. Li, M. JIANG, X. Zhang, M. Kamp, and Q. Dou, "Fed{bn}: Federated learning on non-{iid} features via local batch normalization," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=6YEQUn0QICG

[40] Y. Wu and K. He, "Group normalization," 2018.

[41] Q. Liu, Q. Dou, L. Yu, and P. A. Heng, "Ms-net: Multi-site network for improving prostate segmentation with heterogeneous mri data," *IEEE Transactions on Medical Imaging*, vol. 39, no. 9, pp. 2713–2724, 2020.

[42] D. Basu, D. Data, C. Karakus, and S. N. Diggavi, "Qsparse-local-sgd: Distributed sgd with quantization, sparsification, and local computations," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 217–226, 2020.

[43] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[46] V. Kulkarni, M. Kulkarni, and A. Pant, “Survey of personalization techniques for federated learning,” 03 2020.

[47] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, vol. 54. PMLR, 2017, pp. 1273–1282.

Appendix A

Appendix: Convergence result

A.1 Useful properties

Lemma A.1. *For arbitrary $a,\ b \in \mathbb{R}^d$,*

$$||a+b||^2 \leq (1+\gamma)||a||^2 + (1+\gamma^{-1})||b||^2,\ \gamma > 0$$

and

$$2a^Tb \leq \gamma||a||^2 + \gamma^{-1}||b||^2.$$

Lemma A.2. *For arbitrary set of n variables $\{a_i\}_{i=1}^n$, then*

$$||\sum_i a_i||^2 \leq n\sum_i ||a_i||^2.$$

Lemma A.3. *For $A \in \mathbb{R}^{d\times n},\ B \in \mathbb{R}^{n\times n}$,*

$$||AB||_F \leq ||A||_F||B||_2.$$

Lemma A.4. *(6) Let $\{e_1, ..., e_n\}$ be n random variables in R^d which are not necessarily independent. First suppose that their mean is $\daleth = \epsilon_i$ and variance is bounded as $||\mathbb{E}\daleth - \epsilon\daleth||^{\nvDash} \leq \sigma^2$. Then, the following holds*

$$\mathbb{E}||\sum_i e_i||^2 \leq ||\sum_i \epsilon_i||^2 + n\sigma^2 \tag{A.1}$$

Lemma A.5.

$$\mathbb{E}_W||W_tX - \bar{X}||_F^2 \leq (1-\rho)||X-\bar{X}||_F^2, \tag{A.2}$$

where ρ is the second largest eigenvalue of mixing matrix W. Here deterministic mixing(adjacency) matrix is considered.

Lemma A.6. *(Implication of smoothness)From Assumption* **??**, *for any i,*

$$f_i(x) - f_i(y) \leq \langle \nabla f_i(x), x-y\rangle + \frac{L}{2}||x-y||^2,\ \forall x, y \in \mathbb{R}^d.$$

It also implies

$$||\nabla f_i(x) - \nabla f_i(y)|| \leq L||x-y||.$$

A.2 Overview of decentralized SCAFFOLD

In this section, we show that decentralized would be equivalent to centralized SCAFFOLD when complete graph.
For local model update, both centralized and decentralized SCAFFOLD do

$$y_{i,k-1}^r - \eta_c(\nabla F_i(y_{i,k}^r;\xi_{i,k}) - c_i^r + \tilde{c}_i^r),$$

where for centralized SCAFFOLD, $\tilde{c} = \tilde{c}_i,\ x = x_i,\ \forall i \in [N]$.
SCAFFOLD in the centralized if not considering client sampling.

$$\begin{aligned}
c_i^{r+1} &= c_i^r - \tilde{c} + \frac{1}{K\eta_c}(x^r - y_{i,K}^r) \\
\Delta x_i^r &= y_{i,K}^r - x^r \\
x^{r+1} &= x^r + \eta_s \frac{1}{N}\sum_i \Delta x_i^r \\
\tilde{c}^{r+1} &= \tilde{c}^r + \frac{1}{N}\sum_i (c_i^{r+1} - c_i^r)
\end{aligned}$$

SCAFFOLD in the decentralized

$$\begin{aligned}
c_i^{r+1} &= c_i^r - \tilde{c}_i + \frac{1}{K\eta_c}(x_i^r - y_{i,K}^r) \\
\Delta x_i^r &= y_{i,K}^r - \tilde{x}_i^r \\
x_i^{r+1} &= \sum_j w_{ij}(x_j^r + \eta_s \Delta x_j^r) \\
\tilde{c}_i^{r+1} &= \sum_j w_{ij}(\tilde{c}_j^r + c_j^{r+1} - c_j^r)
\end{aligned}$$

When complete graph for decentralized SCAFFOLD, $w_{ij} = \frac{1}{N},\ \forall i, j,$

$$\begin{aligned}
x^{r+1} \equiv x_i^{r+1} &= \sum_j w_{ij}(x_j^r + \eta_s \Delta x_j^r) \\
&= \frac{1}{N}\sum_j (x_j^r + \eta_s \Delta x_j^r) \quad \triangleright [x_j^r \equiv x^r,\ \forall j \in [N]] \\
&= x^r + \frac{1}{N}\sum_j \eta_s \Delta x_j^r
\end{aligned}$$

$$\begin{aligned}
\tilde{c}^{r+1} \equiv \tilde{c}_i^{r+1} &= \sum_j w_{ij}(\tilde{c}_j^r + c_j^{r+1} - c_j^r) \\
&= \frac{1}{N}\sum_j (\tilde{c}_j^r + c_j^{r+1} - c_j^r) \quad \triangleright [\tilde{c}_j^r \equiv \tilde{c}^r,\ \forall j \in [N]] \\
&= \tilde{c}^r + \frac{1}{N}\sum_j (c_j^{r+1} - c_j^r)
\end{aligned}$$

In addition, for every local control variate c_i, it actually calculates $c_i = \frac{1}{K}\sum_k \nabla F_i(y^r_{i,k};\xi_{i,k})$.

A.2.1 Additional definition

Before proceeding with the proof of our lemmas, we need some additional definitions of the various errors we track. As before, we define the effective step-size to be

$$\eta = K\eta_s\eta_c.$$

We define the variance of global model among workers

$$\Xi_r := \frac{1}{N}\sum_i \mathbf{E}||x^r_i - \bar{x}^r_i||^2$$

We define client-drift to be how much the clients move from their starting point:

$$\Phi_r := \frac{1}{KN}\sum_{i,k} \mathbf{E}||y^r_{i,k} - x^r_i||^2.$$

Because we are estimating the direction difference $c_i - \tilde{c}_i$ and employs it as correction. This leads to estimation of such direction:

$$\Gamma_r := \frac{1}{N}\sum_i \mathbf{E}||c^r_i - \tilde{c}^r_i||^2.$$

A.3 Convergence of decentralized SCAFFOLD for non-convex function

Global model update for averaged model $\eta := K\eta_s\eta_c$,

$$\begin{aligned}\eta_s\Delta\bar{x}^r &= \bar{x}^{r+1} - \bar{x}^r\\ &= -\frac{1}{N}\sum_i \eta_s\eta_c\sum_k(\nabla F_i(y^r_{i,k};\xi_{i,k}) + -c^r_i + \tilde{c}^r_i)\\ &= -\frac{\eta}{KN}\sum_{i,k}(\nabla F_i(y^r_{i,k};\xi_{i,k}) - c^r_i + \tilde{c}^r_i)\\ &= -\frac{\eta}{KN}\sum_{i,k}\nabla F_i(y^r_{i,k};\xi_{i,k}) - \frac{\eta}{K}\sum_k(\frac{1}{N}\sum_i \tilde{c}^r_i - \frac{1}{N}\sum_i c^r_i),\end{aligned}$$

where for particular, rewrite correction of direction difference $\frac{1}{N}\sum_i \tilde{c}^r_i - \frac{1}{N}\sum_i c^r_i$ in matrix format, also note that $W_\infty W = W_\infty$,

$$\begin{aligned}\tilde{C}^r W_\infty - C^r W_\infty &= (\tilde{C}^{r-1} + C^r - C^{r-1})W_\infty W - C^r W_\infty\\ &= \tilde{C}^{r-1}W_\infty + C^r W_\infty - C^{r-1}W_\infty - C^r W_\infty\\ &= \tilde{C}^{r-1}W_\infty - C^{r-1}W_\infty\\ &= \tilde{C}^0 W_\infty - C^0 W_\infty\end{aligned}$$

As long as initializing, $\tilde{C}^0 = C^0 W_\infty$, then

$$\tilde{C}^r W_\infty - C^r W_\infty r = 0 \Leftrightarrow \frac{1}{N}\sum_i \tilde{c}_i^0 = \frac{1}{N}\sum_i c_i^0 \tag{A.3}$$

$$\text{And } \rightarrow \eta_s \Delta \bar{x}^r = -\frac{\eta}{KN}\sum_{i,k} \nabla F_i(y_{i,k}^r; \xi_{i,k}). \tag{A.4}$$

Variance of global model update

Lemma A.7. *The following holds true for any $\eta = K\eta_s\eta_c \in [0, 1/L]$,*

$$\mathbf{E}||\eta_s(\bar{x}^{r+1} - \bar{x}^r)||^2 \leq \frac{\eta^2\sigma^2}{KN} + 2\eta^2 \mathbf{E}||\nabla f(\bar{x}^r)||^2 + 2\eta^2 L^2 \Phi_r \tag{A.5}$$

Proof.

$$\begin{aligned}
&\mathbf{E}||\eta_s(\bar{x}^{r+1} - \bar{x}^r)||^2 \\
&= \mathbf{E}||\eta_s \Delta \bar{x}^r||^2 \\
&= \mathbf{E}||\frac{\eta}{KN}\sum_{i,k} \nabla F_i(y_{i,k}^r; \xi_{i,k})||^2 \\
&= \underbrace{\eta^2 \mathbf{E}||\frac{1}{KN}\sum_{i,k}(\nabla F_i(y_{i,k}^r; \xi_{i,k}) - \nabla f_i(y_{i,k}^r))||^2}_{:=T_3} + \underbrace{\eta^2 \mathbf{E}||\frac{1}{KN}\sum_{i,k} \nabla f_i(y_{i,k}^r)||^2}_{:=T_4} \\
&\quad \triangleright \text{[variance decomposition]}
\end{aligned}$$

Consider using the Lemma A.4, $\mathbf{E}||\sum_{i,k}(\nabla F_i(y_{i,k}^r; \xi_{i,k}) - \nabla f_i(y_{i,k}^r))||^2 \leq KN\sigma^2$, since bounded variance on $\mathbf{E}||\nabla F_i(y_{i,k}^r; \xi_{i,k}) - \nabla f_i(y_{i,k}^r)||^2 \leq \sigma^2$.

$$\begin{aligned}
T_3 :&= \eta^2 \mathbf{E}||\frac{1}{KN}\sum_{i,k}(\nabla F_i(y_{i,k}^r; \xi_{i,k}) - \nabla f_i(y_{i,k}^r))||^2 \\
&= \eta^2 \frac{1}{K^2N^2} \mathbf{E}||\sum_{i,k}(\nabla F_i(y_{i,k}^r; \xi_{i,k}) - \nabla f_i(y_{i,k}^r))||^2 \\
&\leq \frac{\eta^2\sigma^2}{KN}
\end{aligned}$$

$$\begin{aligned}
T_4 :&= \eta^2 \mathbf{E}||\frac{1}{KN}\sum_{i,k}\nabla f_i(y_{i,k}^r)||^2 \\
&= \eta^2 \mathbf{E}||\frac{1}{KN}\sum_{i,k}(\nabla f_i(y_{i,k}^r) - \nabla f_i(\bar{x}^r) + \nabla f_i(\bar{x}^r))||^2 \\
&\leq 2\eta^2 \mathbf{E}||\nabla f(\bar{x}^r)||^2 + \frac{2\eta^2}{KN}\sum_{i,k}\mathbf{E}||\nabla f_i(y_{i,k}^r) - \nabla f_i(\bar{x}^r)||^2 \\
&\leq 2\eta^2 \mathbf{E}||\nabla f(\bar{x}^r)||^2 + 2\eta^2 L^2 \frac{1}{KN}\sum_{i,k}\mathbf{E}||y_{i,k}^r - \bar{x}^r||^2
\end{aligned}$$

□

Progress in one outer loop

Lemma A.8. *If define* $\Phi_r = \frac{1}{KN}\sum_{i,k}\mathbf{E}||y_{i,k}^r - \bar{x}^r||^2$*, then we could have*

$$\mathbf{E}f(\bar{x}^{r+1}) - \mathbf{E}f(\bar{x}^r) \leq (-\frac{\eta}{2} + \eta^2 L)\mathbf{E}||\nabla f(\bar{x}^r)||^2 + \frac{\eta^2 L}{2KN}\sigma^2 + (\frac{\eta}{2} + \eta^2 L)L^2\Phi_r \tag{A.6}$$

Proof. Consider the smoothness of $f(x)$,

$$\begin{aligned}
\mathbf{E}f(\bar{x}^{r+1}) &\leq \mathbf{E}f(\bar{x}^r) + \mathbf{E}\langle \nabla f(\bar{x}^r),\ \eta_s \Delta \bar{x}^r \rangle + \frac{L}{2}\mathbf{E}||\eta_s \Delta \bar{x}^r||^2 \\
&= \mathbf{E}f(\bar{x}^r) + \underbrace{\mathbf{E}\left\langle \nabla f(\bar{x}^r),\ -\frac{\eta}{KN}\sum_{i,k}\nabla F_i(y_{i,k}^r;\xi_{i,k}) \right\rangle}_{:=T_1} + \underbrace{\frac{L}{2}\mathbf{E}||\eta_s \Delta \bar{x}^r||^2}_{:=T_2}
\end{aligned}$$

$$
\begin{aligned}
T_1 :&= \mathbf{E}\left\langle \nabla f(\bar{x}^r),\ -\frac{\eta}{KN}\sum_{i,k}\nabla F_i(y_{i,k}^r;\xi_{i,k})\right\rangle \\
&= \mathbf{E}\left\langle \nabla f(\bar{x}^r),\ -\frac{\eta}{KN}\sum_{i,k}\nabla f_i(y_{i,k}^r)\right\rangle \quad \triangleright \text{[Linearity of expectation]} \\
&= \mathbf{E}\left\langle \nabla f(\bar{x}^r),\ -\frac{\eta}{KN}\sum_{i,k}(\nabla f_i(y_{i,k}^r) - \nabla f_i(\bar{x}^r) + \nabla f_i(\bar{x}^r))\right\rangle \\
&= \mathbf{E}\left\langle \nabla f(\bar{x}^r),\ -\frac{\eta}{KN}\sum_{i,k}(\nabla f_i(y_{i,k}^r) - \nabla f_i(\bar{x}^r)) - \eta\nabla f(\bar{x}^r)\right\rangle \\
&= -\eta\mathbf{E}||\nabla f(\bar{x}^r)||^2 + \frac{\eta}{KN}\sum_{i,k}\mathbf{E}\left\langle \nabla f(\bar{x}^r),\ \nabla f_i(\bar{x}^r) - \nabla f_i(y_{i,k}^r)\right\rangle \\
&\le -\eta\mathbf{E}||\nabla f(\bar{x}^r)||^2 + \frac{\eta}{2KN}\sum_{i,k}\mathbf{E}||\nabla f(\bar{x}^r)||^2 + \frac{\eta}{2KN}\sum_{i,k}\mathbf{E}||\nabla f_i(\bar{x}^r) - \nabla f_i(y_{i,k}^r)||^2 \\
&= -\frac{\eta}{2}\mathbf{E}||\nabla f(\bar{x}^r)||^2 + \frac{\eta}{2KN}\sum_{i,k}\mathbf{E}||\nabla f_i(\bar{x}^r) - \nabla f_i(y_{i,k}^r)||^2
\end{aligned}
$$

The last inequality uses the property that $\langle a, b\rangle \le \frac{1}{2}||a||^2 + \frac{1}{2}||b||^2$.
Combining T_1, T_2,

$$
\begin{aligned}
&\mathbf{E}f(\bar{x}^{r+1}) - \mathbf{E}f(\bar{x}^r) \\
&\le T_1 + T_2 \\
&\le (-\frac{\eta}{2} + \eta^2 L)\mathbf{E}||\nabla f(\bar{x}^r)||^2 + \frac{\eta^2\sigma^2 L}{2KN} + (\frac{\eta}{2KN} + \frac{\eta^2 L}{KN})\sum_{i,k}\mathbf{E}||\nabla f_i(y_{i,k}^r) - \nabla f_i(\bar{x}^r)||^2
\end{aligned}
$$

□

Client direction approximation

Lemma A.9. *Consider the approximation of local control variates to expected client direction, $\mathbf{E}||C^r - \nabla F(\bar{X}^{r-1})||_F^2$, then we have for any $\eta = K\eta_s\eta_c \in [0, 1/L]$, mixing rate $\rho \in (0, 1]$,*

$$\frac{1}{N}\mathbf{E}||C^r - \nabla F(\bar{X}^{r-1})||^2 \le \frac{\sigma^2}{K} + L^2\Phi_{r-1}$$

Proof.

$$\begin{aligned}\frac{1}{N}\mathbf{E}||C^r - \nabla F(\bar{X}^{r-1})||_F^2 &= \frac{1}{N}\mathbf{E}||\frac{1}{K}\sum_k \nabla F(Y_k^{r-1};\xi_k) - \nabla F(\bar{X}^{r-1})||_F^2 \\ &\overset{(A.7.1)}{\leq} \frac{\sigma^2}{K} + \frac{1}{N}\mathbf{E}||\frac{1}{K}\sum_k (\nabla F(Y_k^{r-1}) - \nabla F(\bar{X}^{r-1})||_F^2 \\ &\leq \frac{\sigma^2}{K} + \frac{1}{KN}\sum_k \mathbf{E}||\nabla F(Y_k^{r-1}) - \nabla F(\bar{X}^{r-1})||_F^2 \\ &\leq \frac{\sigma^2}{K} + \frac{L^2}{KN}\sum_k \mathbf{E}||Y_k^{r-1} - \bar{X}^{r-1}||_F^2 = \frac{\sigma^2}{K} + L^2\Phi_{r-1}\end{aligned} \tag{A.7}$$

(A.7.1) follows from application of Lemma A.4, where

$$\begin{aligned}&\frac{1}{K^2N}\sum_i^N \mathbf{E}_{\xi|Y_{k-1}^{r-1}}||\sum_k \nabla F(Y_k^{r-1};\xi_k) - \nabla F(\bar{X}^{r-1})||^2 \\ &\leq \frac{1}{K^2N}\sum_i^N \mathbf{E}_{\xi|Y_{k-1}^{r-1}}||\sum_k (\nabla F(Y_k^{r-1}) - \nabla F(\bar{X}^{r-1})||^2 + \frac{KN\sigma^2}{K^2N}\end{aligned}$$

□

Direction correction for each client

Lemma A.10. *If define the deviation between local and global control variate as* $\Gamma_r := \frac{1}{N}\mathbf{E}||C^r - \tilde{C}^r||_F^2$*, then we have for any* $\eta = K\eta_s\eta_c \in [0, 1/L]$*, mixing rate* $\rho \in (0, 1]$*,*

$$\Gamma_r \leq (1 - \frac{\rho}{2})\Gamma_{r-1} + \frac{16}{\rho}(\frac{\sigma^2}{K} + G^2 + B^2\Omega_{r-1}) + \frac{16}{\rho}L^2\Phi_{r-1} \tag{A.8}$$

Proof.

$$\begin{aligned}\Gamma_r &= \frac{1}{N}\mathbf{E}||\tilde{C}^r - C^r||_F^2 \\ &= \frac{1}{N}\mathbf{E}||(C^r - C^{r-1} + \tilde{C}^{r-1})W - C^r||_F^2 \\ &\overset{(A.9.1)}{\leq} (1+b)\frac{1}{N}\mathbf{E}||(\tilde{C}^{r-1} - C^{r-1})W - (\tilde{C}^{r-1} - C^{r-1})W_\infty||_F^2 + (1+b^{-1})\frac{1}{N}\mathbf{E}||C^r(W-I)||_F^2 \\ &\overset{(A.9.2)}{\leq} (1+b)(1-\rho)\frac{1}{N}\mathbf{E}||\tilde{C}^{r-1} - C^{r-1}||_F^2 + 4(1+\frac{1}{b})\frac{1}{N}\mathbf{E}||C^r||_F^2 \\ &\overset{(A.9.3)}{\leq} (1+b)(1-\rho)\frac{1}{N}\mathbf{E}||\tilde{C}^{r-1} - C^{r-1}||_F^2 \\ &\quad + 8(1+\frac{1}{b})\frac{1}{N}\mathbf{E}||C^r - \nabla F(\bar{X}^{r-1})||_F^2 + 8(1+\frac{1}{b})\frac{1}{N}\mathbf{E}||\nabla F(\bar{X}^{r-1})||_F^2\end{aligned} \tag{A.9}$$

(A.9.1), (A.9.3): $||x+y||^2 \le (1+b)||x||^2 + (1+b^{-1})||y||^2, \forall b > 0$, where for later (A.9.3), let $b = 1$. And $(\tilde{C}^{r-1} - C^{r-1})W_\infty = 0$ in (A.3)
(A.9.2): for any matrix X, $||XW - \bar{X}||_F^2 \le (1-\rho)||X - \bar{X}||_F^2$, and $||XW - X||_F \le ||W - I||||X||_F \le 2||X||_F$, where $||W - I||$ is the spectrum norm and its value is equals to its largest eigenvalue.

And under bounded dissimilarity assumption 3.7, rewrite the assumption in matrix format,

$$\frac{1}{N}||\nabla F(x)||_F^2 = \frac{1}{N}\sum_i ||\nabla f_i(x)||^2 \le G^2 + B^2||\nabla f(x)||^2.$$

In conclude, plugging them into (A.9.3)

$$\Gamma_r \le (1+b)(1-\rho)\Gamma_{r-1} + 8(1+\frac{1}{b})(\frac{\sigma^2}{K} + G^2 + B^2\mathbf{E}||\nabla f(\bar{x}^{r-1})||^2) + 8(1+\frac{1}{b})L^2\Phi_{r-1}$$

If $b = \frac{1}{\frac{2}{\rho}-1}$, then $(1+b)(1-\rho) \le (1-\frac{\rho}{2})$, and $(1+\frac{1}{b}) \le \frac{2}{\rho}$, which completes the proof. $\square$

Variance among workers w.r.t global model

Lemma A.11. *If define variance among workers $\Xi_r := \frac{1}{N}\mathbf{E}||X^r - \bar{X}^r||_F^2$, and the deviation between local and global control variate as $\Gamma_r := \frac{1}{N}\mathbf{E}||C^r - \tilde{C}^r||_F^2$, then we have for any $\eta = K\eta_s\eta_c$, s.t $\eta^2L^2 \le c^2\rho^{2\alpha}$ where c is a constant determined later in Lemma A.13 and $\alpha \in [\frac{1}{2}, 1]$, and mixing rate $\rho \in (0, 1]$,*

$$\Xi_r \le (1-\frac{\rho}{2})\Xi_{r-1} + \frac{8\eta^2}{\rho}(\frac{\sigma^2}{K} + G^2 + B^2\mathbf{E}||\nabla f(\bar{x}^{r-1})||^2) + \frac{8\eta c\rho}{L}\Gamma_{r-1} + 8c^2\rho\Phi_{r-1} \tag{A.10}$$

Proof. It's known that

$$X^r = (X^{r-1} + \eta_s\sum_k(Y_k^{r-1} - X^{r-1}))W, \quad \bar{X}^r = X^rW_\infty.$$

Then define the progress after local steps, $\Delta X^r = Y_k^r - X^r$, and $\Delta\bar{X}^r = \Delta X^rW_\infty$

$$\begin{aligned}\eta_s\Delta X^r = Y_K^r - X^r &= -\eta_s\eta_c\sum_{k=0}^{K-1}(\nabla F(Y_k^r;\xi_k) - C^r + \tilde{C}^r)\\ &= -\eta_s\eta_c\sum_{k=0}^{K-1}\nabla F(Y_k^r;\xi_k) - \eta_s\eta_cK(\tilde{C}^r - C^r)\\ &= -\eta(C^{r+1} + \tilde{C}^r - C^r)\end{aligned}$$

And $\eta_s \Delta \bar{X}^r = -\eta C^{r+1} W_\infty + \tilde{C}^r W_\infty - C^r W_\infty \overset{(A.3)}{=} -\eta C^{r+1} W_\infty$, since in (A.3) $C^r W_\infty = \tilde{C}^r W_\infty$.
Then,

$$\begin{aligned}
\frac{1}{N}\mathbf{E}||X^r - \bar{X}^r||_F^2 &= \frac{1}{N}\mathbf{E}||(X^{r-1} + \eta_s \Delta X^{r-1})W - (\bar{X}^{r-1} + \eta_s \Delta \bar{X}^{r-1})||_F^2 \\
&\leq \frac{1}{N}(1+b)\mathbf{E}||X^{r-1}W - \bar{X}^{r-1}||_F^2 + (1+b^{-1})\frac{1}{N}\mathbf{E}||\eta_s \Delta X^{r-1} W - \eta_s \Delta \bar{X}^{r-1}||_F^2 \\
&\leq (1+b)(1-\rho)\Xi_{r-1} + (1+b^{-1})(1-\rho)\frac{1}{N}\mathbf{E}||\eta_s \Delta X^{r-1} - \eta_s \Delta \bar{X}^{r-1}||_F^2 \\
&= (1+b)(1-\rho)\Xi_{r-1} + (1+b^{-1})(1-\rho)\frac{\eta^2}{N}\mathbf{E}||C^r + \tilde{C}^{r-1} - C^{r-1} - C^r W_\infty||_F^2
\end{aligned}$$

Here using $||A - \bar{A}||_F^2 = \sum_i ||a_i - \bar{a}||^2 \leq \sum_i ||a_i||^2 = ||A||_F^2$, we could have

$$||C^r + \tilde{C}^{r-1} - C^{r-1} - C^r W_\infty||_F^2 \leq ||C^r + \tilde{C}^{r-1} - C^{r-1}||_F^2,$$

since $(C^r + \tilde{C}^{r-1} - C^{r-1})W_\infty = C^r W_\infty$.

$$\begin{aligned}
\frac{1}{N}\mathbf{E}||C^r + \tilde{C}^{r-1} - C^{r-1}||_F^2 &\leq \frac{2}{N}\mathbf{E}||C^r - \nabla F(\bar{X}^{r-1}) + \nabla F(\bar{X}^{r-1})||_F^2 + \frac{2}{N}\mathbf{E}||\tilde{C}^{r-1} - C^{r-1}||_F^2 \\
&\leq \frac{4}{N}\mathbf{E}||C^r - \nabla F(\bar{X}^{r-1})||_F^2 + \frac{4}{N}\sum_i \mathbf{E}||\nabla f_i(\bar{x}^{r-1})||^2 + 2\Gamma_{r-1} \\
&\leq \frac{4}{N}\mathbf{E}||C^r - \nabla F(\bar{X}^{r-1})||_F^2 + 4(G^2 + B^2\mathbf{E}||\nabla f(\bar{x}^{r-1})||^2) + 2\Gamma_{r-1}
\end{aligned} \tag{A.11}$$

$$\begin{aligned}
\Xi_r \leq& (1+b)(1-\rho)\Xi_{r-1} + (1+\frac{1}{b})(1-\rho)4\eta^2(\frac{\sigma^2}{K} + G^2 + B^2\mathbf{E}||\nabla f(\bar{x}^{r-1})||^2 + \frac{1}{2}\Gamma_{r-1}) \\
&+ (1+\frac{1}{b})(1-\rho)4\eta^2 L^2 \Phi_{r-1}
\end{aligned} \tag{A.12}$$

If $b = \frac{1}{\frac{2}{\rho}-1}$, then $(1+b)(1-\rho) \leq (1-\frac{\rho}{2})$, and $(1+\frac{1}{b}) \leq \frac{2}{\rho}$. And if plugging the stepsize bound of form $\eta^2 L^2 \leq c^2\rho^{2\alpha} \leq c^2\rho$ where c is a constant determined later and $\alpha \in [\frac{1}{2}, 1]$, then completes the proof. □

Bound the local steps

Lemma A.12. *If define the averaged local model updates $\Phi_r := \frac{1}{KN}\sum_{i.k}||y_{i,k}^r - x_i^r||^2$, then we have for any $\eta = K\eta_s\eta_c$, s.t $\eta^2 L^2 \leq c^2\rho^{2\alpha}$ where c is a constant determined later in Lemma A.13 and $\alpha \in [\frac{1}{2}, 1]$, and mixing rate $\rho \in (0, 1]$,*

$$\Phi_r \leq \frac{a\eta^2 L\sigma^2}{K} + 4a\eta^3 L^2(G^2 + B^2\mathbf{E}||\nabla f(\bar{x}^r)||^2) + \frac{2ac\eta}{L}\Gamma_r + a\Xi_r \tag{A.13}$$

Proof.
Since $\eta^2 L^2 \leq c^2\rho^{2\alpha} \leq c^2$, then $4K\eta_c^2 L^2 \leq \frac{4\eta^2 L^2}{K} \leq \frac{4c^2}{K-1}$

$$\begin{aligned}\sum_{\tau=0}^{k-1}(1+\frac{1}{k-1}+4K\eta_c^2L^2)^\tau &\leq \sum_{\tau=0}^{K-1}(1+\frac{1}{K-1}+4K\eta_c^2L^2)^\tau \\ &\leq K(1+\frac{1}{K-1}+4K\eta_c^2L^2)^{K-1} \\ &\leq K(1+\frac{1+4c}{K-1})^{K-1} \\ &\leq Ke^{1+4c^2} \leq aK\end{aligned} \tag{A.14}$$

$$\begin{aligned}\Phi_r &:= \frac{1}{KN}\sum_k \mathbf{E}||Y_k^r - \bar{X}^r||_F^2 \\ &\leq \frac{1}{KN}\sum_k((\eta_c^2\sigma^2 N + 2K\eta_c^2\mathbf{E}||\nabla F(\bar{X}^r) - C^r + \tilde{C}^r||_F^2)aK \\ &+ (1+\frac{1}{K-1}+4K\eta_c^2L^2)^k\mathbf{E}||X^r - \bar{X}^r||_F^2) \\ &= (\eta_c^2\sigma^2 N + 4K\eta_c^2\mathbf{E}||\nabla F(\bar{X}^r)||_F^2 + 2K\eta_c^2\mathbf{E}|| - C^r + \tilde{C}^r||_F^2)\frac{aK}{N} \\ &+ \frac{1}{KN}\sum_{k=0}^{K-1}(1+\frac{1}{K-1}+4K\eta_c^2L^2)^k\mathbf{E}||X^r - \bar{X}^r||_F^2 \\ &\leq (\frac{\eta^2\sigma^2}{K} + 4\eta^2(G^2 + B^2\mathbf{E}||\nabla f(\bar{x}^r)||^2) + 2\eta^2\Gamma_r)a + a\Xi_r \quad \triangleright [\eta L \leq c] \\ &\leq \frac{a\eta c\sigma^2}{LK} + 4a\eta^2(G^2 + B^2\mathbf{E}||\nabla f(\bar{x}^r)) + \frac{2ac\eta}{L}\Gamma_r + a\Xi_r\end{aligned} \tag{A.15}$$

□

Lemma A.13. *Define $U_{r+1} := \mathbf{E}f(\bar{x}^{r+1}) + \frac{25}{2}\frac{\eta^2 L}{\rho}\Gamma_{r+1} + \frac{\eta L^2}{\rho}\Xi_{r+1}$, for any step size satisfying $\eta = K\eta_s\eta_c \leq c\rho^{2\alpha}$, where $c = \frac{1}{500B^2}$, and $\alpha = 1$,*

$$U_{r+1} \leq U_r + \eta^2(\frac{629L\sigma^2}{4K} + \frac{200L\sigma^2}{\rho^2 K} + \frac{L\sigma^2}{2KN} + \frac{57L}{800}G^2 + \frac{200L}{\rho^2}G^2) - \frac{\eta}{12}\mathbf{E}||f(\bar{x}^r)||^2$$

Proof. To start with, since $c = \frac{1}{500B^2}$, then in (A.14), $e^{1+4c^2} \leq 3$.

$$
\begin{aligned}
\Phi_r \leq& \frac{a\eta c\sigma^2}{LK} + 4ac^2(G^2 + B^2\mathbf{E}||\nabla f(\bar{x}^r)) + \frac{2ac\eta}{L}\Gamma_r + a\Xi_r \\
\Rightarrow \frac{25}{12}\eta L^2\Phi_r \leq& \frac{25}{12}\frac{a\eta^2 L\sigma^2}{K} + \frac{25}{3}a\eta^3 L^2(G^2 + B^2\mathbf{E}||\nabla f(\bar{x}^r)) + \frac{25}{6}ac\eta^2 L\Gamma_r + \frac{25}{12}a\eta L^2\Xi_r \\
\Gamma_{r+1} \leq& (1-\frac{\rho}{2})\Gamma_r + \frac{16}{\rho}(\frac{\sigma^2}{K} + G^2 + B^2\mathbf{E}||f(\bar{x}^r)||^2) + \frac{16}{\rho}L^2\Phi_r \quad \triangleright [\eta L \leq c\rho \text{ for last term}] \\
\Rightarrow \frac{25}{2}\frac{\eta^2 L}{\rho}\Gamma_{r+1} \leq& \frac{25}{2}\frac{\eta^2 L}{\rho}\Gamma_r - \frac{25}{2}\frac{\eta^2 L}{2}\Gamma_r + 200\eta^2 L\frac{1}{\rho^2}(\frac{\sigma^2}{K} + G^2 + B^2\mathbf{E}||f(\bar{x}^r)||^2) + 200c\frac{\eta L^2}{\rho}\Phi_r \\
\Xi_r \leq& (1-\frac{\rho}{2})\Xi_r + \frac{8\eta^2}{\rho}(\frac{\sigma^2}{K} + G^2 + B^2\mathbf{E}||f(\bar{x}^r)||^2) + \frac{8\eta c\rho}{L}\Gamma_r + 8c\rho\Phi_r \\
\Rightarrow \frac{\eta L^2}{\rho}\Xi_{r+1} \leq& \frac{\eta L^2}{\rho}\Xi_r - \frac{\eta L^2}{2}\Xi_r + \frac{8\eta^3 L^2}{\rho^2}(\frac{\sigma^2}{K} + G^2 + B^2\mathbf{E}||f(\bar{x}^r)||^2) + 8c\eta^2 L\Gamma_r + 8c^2\eta L^2\Phi_r
\end{aligned}
\tag{A.16}
$$

Combining with (A.7), consider
$(\mathbf{E}f(\bar{x}^{r+1}) + \frac{25}{2}\frac{\eta^2 L}{\rho}\Gamma_{r+1} + \frac{\eta L^2}{\rho}\Xi_{r+1}) \leq (\mathbf{E}f(\bar{x}^r) + \frac{25}{2}\frac{\eta^2 L}{\rho}\Gamma_r + \frac{\eta L^2}{\rho}\Xi_r) + \Gamma_r\eta^2 L(\frac{25}{2}c - \frac{25}{4} + 8c) + \Xi_r\eta L^2(\frac{25}{4} - \frac{25}{4}) + \Phi_r\eta L^2(-\frac{25}{12} + 200\frac{c}{\rho} + 8c^2 + 1 + \eta L) + \frac{25}{12}(\frac{3\eta^2 L\sigma^2}{K} + 12\eta^3 L^2(G^2 + B^2\Omega_r)) + \frac{25}{2}(16\eta^2 L\frac{1}{\rho^2}(\frac{\sigma^2}{K} + G^2 + B^2\Omega_{r-1})) + (\frac{8\eta^3 L^2}{\rho^2}(\frac{\sigma^2}{K} + G^2 + B^2\Omega_{r-1})) + (-\frac{\eta}{2} + \eta^2 L)\Omega_r + \frac{\eta^2 L\sigma^2}{2KN}$, for the following coefficients,

$$
\begin{aligned}
\text{for } \Gamma_r: &\quad \eta^2 L(\frac{25}{2}c - \frac{25}{4} + 8c) \leq 0, \\
\text{for } \Xi_r: &\quad \eta L^2(\frac{25}{4} - \frac{25}{4}) \leq 0, \\
\text{for } \Phi_r: &\quad \eta L^2(-\frac{25}{12} + 200\frac{c}{\rho} + 8c^2 + 1 + \eta L) \leq 0,
\end{aligned}
\tag{A.17}
$$

when $c \leq \frac{\rho}{500B^2} \leq \frac{1}{500}$, $\rho \leq 1$, $B^2 \geq 1$ in their definition. Then by more collecting,

$$
\begin{aligned}
U_{r+1} - U_r \leq& \frac{25}{12}(\frac{3\eta^2 L\sigma^2}{K} + 12\eta^3 L^2(G^2 + B^2\Omega_r)) + \frac{25}{2}(16\eta^2 L\frac{1}{\rho^2}(\frac{\sigma^2}{K} + G^2 + B^2\Omega_{r-1})) \\
&+ (\frac{8\eta^3 L^2}{\rho^2}(\frac{\sigma^2}{K} + G^2 + B^2\Omega_{r-1})) + (-\frac{\eta}{2} + \eta^2 L)\Omega_r + \frac{\eta^2 L\sigma^2}{2KN} \\
\leq& \eta^2(\frac{25L\sigma^2}{4K} + \frac{200L\sigma^2}{\rho^2 K} + \frac{L\sigma^2}{2KN} + \frac{L}{32}G^2 + \frac{200L}{\rho^2}G^2 + \frac{L\sigma^2}{25K} + \frac{L}{25}G^2) \\
&+ \eta(-\frac{1}{2} + \frac{1}{500B^2} + \frac{1}{80} + \frac{2}{5} + \frac{1}{31250})\Omega^2 \\
\leq& \eta^2(\frac{629L\sigma^2}{4K} + \frac{200L\sigma^2}{\rho^2 K} + \frac{L\sigma^2}{2KN} + \frac{57L}{800}G^2 + \frac{200L}{\rho^2}G^2) - \frac{\eta}{12}\Omega^2
\end{aligned}
\tag{A.18}
$$

□

Then unrolling the recursion, and Define $v_0 = \frac{629L\sigma^2}{4K} + \frac{200L\sigma^2}{\rho^2 K} + \frac{L\sigma^2}{2KN} + \frac{57L}{800}G^2 + \frac{200L}{\rho^2}G^2$, then

$$\psi_R = \frac{1}{R+1}\sum_r^R \frac{1}{12}\Omega_r \leq \frac{1}{R+1}\sum_r \left(\frac{U_r}{\eta} - \frac{U_{r+1}}{\eta}\right) + \eta v_0 \leq \frac{U_0}{\eta(R+1)} + \eta v_0,$$

where $U_0 = f(\bar{x}^0)$ if initializing with $||C^0 - \tilde{C}^0|| = 0,\ ||X^0 - \bar{X}^0|| = 0$.
Via Lemma 2 of sublinear convergence rate from SCAFFOLD(6),
there exists constant step size $\eta \leq \eta_{max}$,

- When $R + 1 \leq \frac{U_0}{v_0 \eta_{max}^2}$, pick η_{max},

$$\psi_R \leq \frac{U_0}{\eta_{max}(R+1)} + \frac{\sqrt{v_0 U_0}}{\sqrt{R+1}}$$

- if $\eta_{max}^2 \geq \frac{U_0}{v_0(R+1)}$, pick $\eta = \frac{U_0}{v_0(R+1)}$

$$\psi_R = \frac{2\sqrt{v_0 U_0}}{\sqrt{R+1}}$$

Then in conclude, ψ_R is smaller than the union bound of these two cases,

$$\psi_R \leq \frac{U_0}{\eta_{max}(R+1)} + \frac{2\sqrt{v_0 U_0}}{\sqrt{R+1}}$$

$v_0 = O(\frac{\sigma^2 L}{\rho^2 K} + \frac{L\sigma^2}{KN} + \frac{LG^2}{\rho^2})$ then $\psi_R = O(\frac{U_0 B^2 L}{\rho^2 R} + \sqrt{\frac{\sigma^2 L}{\rho^2 K} + \frac{L\sigma^2}{KN} + \frac{LG^2}{\rho^2}}\sqrt{\frac{U_0}{R}})$ when $\eta \leq \frac{1}{500L}\frac{\rho^2}{B^2}$, which completes the proof of theorem 4.1.