Local Adaptivity in Federated Learning: Convergence and Consistency

Jianyu Wang [*1] Zheng Xu [2] Zachary Garrett [2] Zachary Charles [2] Luyang Liu [2] Gauri Joshi [1]

Abstract

The federated learning (FL) framework trains a machine learning model using decentralized data stored at edge client devices by periodically aggregating locally trained models. Popular optimization algorithms of FL use vanilla (stochastic) gradient descent for both local updates at clients and global updates at the aggregating server. Recently, adaptive optimization methods such as AdaGrad have been studied for server updates. However, the effect of using adaptive optimization methods for local updates at clients is not yet understood. We show in both theory and practice that while local adaptive methods can accelerate convergence, they can cause a non-vanishing solution bias, where the final converged solution may be different from the stationary point of the global objective function. We propose correction techniques to overcome this inconsistency and complement the local adaptive methods for FL. Extensive experiments on realistic federated training tasks show that the proposed algorithms can achieve faster convergence and higher test accuracy than the baselines without local adaptivity.

1. Introduction

Federated learning (FL) is an emerging paradigm to perform distributed machine learning model training on decentralized edge clients (*e.g.,* mobile phones or IoT devices) under the orchestration of a central server, while keeping private training data on the client devices (Kairouz et al., 2019). In the cross-device FL setting, a global model is usually trained by a collaborative process of thousands or even millions of participating clients. In each training round of FL, the central server broadcasts the global model to a random subset of clients to perform local model training, and each participated client only uploads the model parameter changes back to the server. Then, the server aggregates local changes to update the global model and continues the next round. This FL training procedure was proposed by McMahan et al. (2017) as the *federated averaging* (FEDAVG) algorithm, and widely applied in diverse applications (Hard et al., 2018; Brisimi et al., 2018; Xu et al., 2020).

Recently, Reddi et al. (2020) proposed a generalization of FEDAVG referred to as FEDOPT. FEDOPT is a flexible algorithmic framework that allows the clients and the server to choose different optimization methods (which are referred to as CLIENTOPT and SERVEROPT) more general than stochastic gradient descent (SGD) in FEDAVG. The key idea is to treat the aggregated local changes from clients as a "pseudo-gradient" and use it as input to SERVEROPT when updating the global model. A few previous works explored the choices of server optimizer (SERVEROPT) in the FEDOPT framework. For example, Hsu et al. (2019); Wang et al. (2020b) used SGD with momentum at the server while keeping client optimizer as SGD and observed significant empirical improvements.

Adaptive methods, such as ADAGRAD (McMahan & Streeter, 2010; Duchi et al., 2011), ADAM (Kingma & Ba, 2014), have achieved superior empirical performance over SGD in centralized training of machine learning models for some applications. In particular, ADAM and its variants (Reddi et al., 2019; Zaheer et al., 2018; Zhuang et al., 2020) are widely recognized as the preferred optimizers for language-related training tasks. Motivated by their superior performance, Reddi et al. (2020) studied a specific class of FEDOPT, where CLIENTOPT is still SGD but SERVEROPT uses adaptive methods. It has been validated through extensive experiments in (Reddi et al., 2020; Tong et al., 2020) that the server-only adaptive methods can achieve faster convergence than vanilla FEDAVG in many federated training tasks.

Despite the success of using server-side adaptive methods, the power of adaptivity has not been fully utilized in the FEDOPT framework, particularly on the client side. Instead of using adaptive methods only in SERVEROPT for periodical

---

[*]Equal contribution [1]Carnegie Mellon University [2]Google Research. Correspondence to: Jianyu Wang <jianyuw1@andrew.cmu.edu>, Zheng Xu <xuzheng@google.com>, Zachary Garrett <zachgarrett@google.com>.
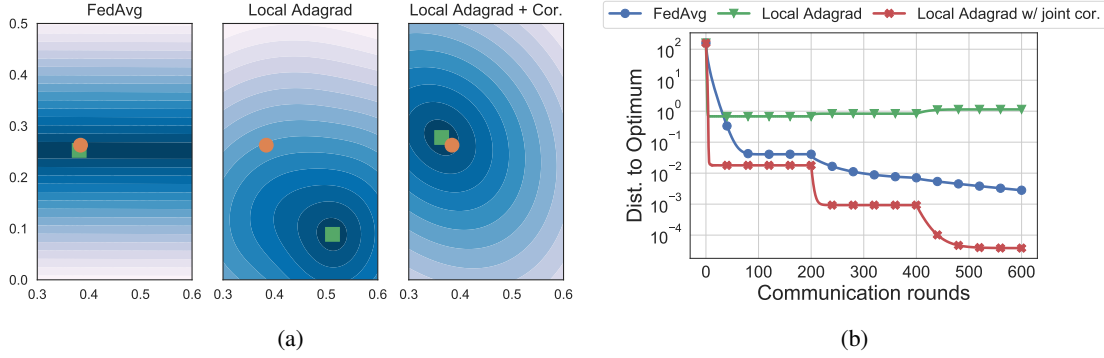
Figure 1. The effects of local adaptive methods on a synthetic 2D quadratic problem. Instantiations of FEDOPT with local adaptivity can be viewed as finding the fixed points of some operator $\mathbb{E}[\mathcal{A}]$. For various methods, we plot (a) the norm $\|\boldsymbol{x} - \mathbb{E}[\mathcal{A}(\boldsymbol{x})]\|$, where darker colors indicate closer to the fixed point (Under local ADAGRAD, the optimization landscape is better conditioned, but the fixed-point is inconsistent with the global minimum of the original global objective.); (b) training curves where the client learning rate is decayed at rounds 200, 400. We develop correction methods that can correct the inconsistency, while retaining the better conditioning.

updates per communication round, one can also choose to use adaptive methods in CLIENTOPT for every local iteration, or in both CLIENTOPT and SERVEROPT. Client adaptive methods can accelerate the local convergence by better utilizing the geometric information of the local objectives. Unfortunately, there are little to no literature exploring this promising direction. It remains an open problem whether the faster local convergence can be translated into faster global convergence and savings in communication rounds.

In this paper, we explore the usage of local adaptive methods and provide affirmative answers to the question: *Can federated learning effectively use adaptive optimization methods on local clients?* Specifically, our main contributions are listed as follows.

1. We identify that naively changing CLIENTOPT in the FE-DOPT framework from SGD to adaptive methods or other stateful optimizers can be problematic, as it does not define how to update client optimizer states (*i.e.,* pre-conditioners and momentum buffers) across rounds. We propose to overcome this discontinuity issue by restarting the update of client optimizer states at the beginning of each round. This simple method enables the usage of local adaptivity in FE-DOPT. We also show that, on many practical training tasks, using adaptive optimizers on clients can achieve faster convergence and higher test accuracy than previous methods.

2. We further provide a theoretical analysis for strongly convex functions to understand the effects of various client optimizers. Our theorem suggests that performing FEDOPT can be viewed as finding the fixed points of some operator $\mathbb{E}[\mathcal{A}]$, the expression of which is determined by client optimizers. Local adaptivity can change $\mathbb{E}[\mathcal{A}]$ in a way that yields faster convergence. However, this enhanced optimization may come with the cost of a *non-vanishing* solution bias – the point we converge to may be far away from the global minimizer of the original objective function, as illustrated

in Figure 1.

3. In order to overcome the side effects of using adaptive client optimizers, we propose (1) local correction technique that can help to mitigate the non-vanishing solution bias; (2) global correction technique that can help to perserve the fast convergence property of adaptive methods. Using one or both of them on the top of adaptive client optimizers can achieve the highest test accuracy on all considered federated training tasks.

Related Works. In terms of using adaptive methods on clients, Xie et al. (2019); Karimireddy et al. (2020) proposed to use the same optimizer states (pre-conditioners and momentum buffers) on all clients, which is similar to server adaptive methods in (Reddi et al., 2020) and does not exploit local adaptivity. To the best of our knowledge, this paper is the first to let clients separately update their optimizer states and study the effects of adaptive client optimizers. Besides, there are few recent literature (Wang et al., 2020a; Charles & Konečný, 2021; Pathak & Wainwright, 2020; Malinovskiy et al., 2020) also study the phenomenon that FEDAVG-style algorithms can converge to a mismatched solution from the optimal one. We refer readers to Section 4 for detailed comparisons with these works.

2. Preliminaries

The Federated Learning Setup. In federated learning, the goal is to minimize the objective

$$F(\boldsymbol{x}) = \mathbb{E}_{i \sim \mathcal{C}}[F_i(\boldsymbol{x})] \qquad (1)$$

where $F_i(\boldsymbol{x}) = \frac{1}{|\mathcal{D}_i|} \sum_{\xi \in \mathcal{D}_i} f_i(\boldsymbol{x}, \xi)$ denotes the local objective function at client i, and $f_i(\boldsymbol{x}, \xi)$ is the loss function where ξ represents one data sample from the local dataset. In contrast to classic distributed learning, federated learning does not allow clients to share their local data with

others or with the central server. As a consequence, the local data distributions $\mathcal{D}_i$ may differ across clients. The term $\mathcal{C}$ denotes the distribution over the collection of all clients. The probability of selecting client i is $w_i = \mathbb{P}_{\mathcal{C}}(i)$, which is proportional to the dataset size of client i. Thus, for a finite set of M clients, the objective function becomes $F(\boldsymbol{x}) = \sum_{i=1}^{M} w_i F_i(\boldsymbol{x})$.

Operator View of Client Optimizers. In this paper, we are going to use operators to represent the general optimizers on clients. In particular, for each client i, we have the following definition.

Definition 1 (Client Optimizer Operator). *We define an operator $\mathcal{A}_i(\boldsymbol{x}; k, \Xi_i, \boldsymbol{s}_i) : \mathbb{R}^d \to \mathbb{R}^d$, which outputs the updated model after performing k steps of a given client optimizer on the local objective F_i, from initial model $\boldsymbol{x}$, with initial optimizer states $\boldsymbol{s}_i$ and random sources Ξ_i (e.g., noise in stochastic gradients).*

The operator $\mathcal{A}_i$ defines how the local models and the client optimizer states (*i.e.,* pre-conditioners and momentum buffers) are updated. When the parameters $k, \boldsymbol{s}_i, \Xi_i$ are given, the output of $\mathcal{A}_i$ will only depend on the starting point $\boldsymbol{x}$. The specific form of $\mathcal{A}_i$ changes with the choice of optimizers and its analytical expression can be complicated. For example, when the client optimizer is pre-conditioned SGD (a generalized version of ADAGRAD) (Martens, 2020), we have

$$\mathcal{A}_i(\boldsymbol{x}; k, \Xi_i, \boldsymbol{s}_i) = \boldsymbol{x} - \eta_i \sum_{s=0}^{k-1} \boldsymbol{P}_i^{(s)} g_i(\boldsymbol{x}^{(s)}; \xi^{(s)}) \quad (2)$$

where η_i is the client learning rate, $\boldsymbol{x}^{(s)}$ is the local model after s local iterations, $g(\boldsymbol{x}^{(s)}; \xi^{(s)})$ denotes the stochastic gradient evaluated on a random mini-batch $\xi^{(s)}$, and $\boldsymbol{P}_i^{(s)}$ is referred to as the *pre-conditioner*.

Operator View of FEDOPT. Based on Definition 1, at each round t of FEDOPT, the server broadcasts the global model $\boldsymbol{x}^{(t)}$ as an initial point to clients. Then, client i uses $\mathcal{A}_i$ to obtain the locally updated model as follows:

$$\boldsymbol{x}_i^{(t,\tau_i)} = \mathcal{A}_i(\boldsymbol{x}^{(t)}; \tau_i, \Xi_i^{(t)}, \boldsymbol{s}_i^{(t)}). \quad (3)$$

After that, the server aggregates local models changes $\boldsymbol{x}^{(t)} - \boldsymbol{x}_i^{(t,\tau_i)}$ and uses SERVEROPT to update the global model to $\boldsymbol{x}^{(t+1)}$. Since the server optimizer does not influence the properties of $\mathcal{A}_i$ and the main purpose of this paper is to understand the effects of client optimizers, we assume SERVEROPT is gradient descent (GD) unless stated otherwise, which means that each client participates every round and there is no server pre-conditioner. Our results can be naturally extended to other server optimizers and sampled clients without changing the main insights on client optimizers. Then, the GD-based global update rule of FEDOPT can

be written as

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \alpha \sum_{i=1}^{M} w_i [\boldsymbol{x}^{(t)} - \mathcal{A}_i(\boldsymbol{x}^{(t)}; \tau_i, \Xi_i^{(t)}, \boldsymbol{s}_i^{(t)})]$$
(4)

where α denotes the server learning rate for SERVEROPT.

3. How to Apply Adaptive Methods on Clients

In this section, we describe the challenges of using local adaptivity in FEDOPT and propose a simple, yet effective method to enable the usage of local adaptivity in FEDOPT without incurring additional communication costs.

Challenge in FEDOPT: Discontinuity of Client Optimizer States. According to the update rules (3) and (4) of FEDOPT, we observe that FEDOPT can not be directly applied to adaptive client optimizers, as it lacks the articulation of handling the initial optimizer states $\boldsymbol{s}_i$ across rounds and clients. To be specific, in FEDOPT, the client updates are isolated not only from each other but also from different rounds due to the local iterations. While FEDOPT uses SERVEROPT to bridge the gap between local iterates in different rounds (*i.e.,* it specifies how to obtain $\boldsymbol{x}^{(t+1)}$ from $\{\boldsymbol{x}_i^{(t,\tau_i)}\}$), it does not define how to update the initial optimizer states $\boldsymbol{s}_i$ across rounds. We refer to this problem as *discontinuity of client optimizer states*.

A natural but impractical idea to tackle the discontinuity problem is to let each client inherit its own client optimizer states from previous rounds. However, these previous states can be stale and inaccurate because they are evaluated at point $\boldsymbol{x}_i^{(t-1,\tau_i)}$ and do not take account of the possibly dramatic changes from $\boldsymbol{x}_i^{(t-1,\tau_i)}$ to the current iterate $\boldsymbol{x}^{(t)}$. When the server only selects a random subset of clients at each round, the staleness will be further exacerbated, as one client may be disconnected from the server for multiple rounds. Besides, this solution is impossible in many applications of FL (*e.g.,* the cross-device setting) that requires clients to be stateless and not to maintain any persistent states across rounds, due to privacy and system constraints (Kairouz et al., 2019).

Our Proposal: Restarting Client Optimizer States. To work with the above challenges, we propose a simple yet effective approach. Instead of applying potentially complicated mechanisms to synchronize optimizer states across clients, we restart the updates of client optimizer states at the beginning of each round. That is, resetting the preconditioners to a constant and resetting the momentum buffers to zero. This strategy does not require any information from previous rounds, so it is compatible with application settings that require stateless clients. Also, it does not incur addition communication costs, as only the local model changes are aggregated.

To verify the advantage of this simple strategy, in Figure 1b,

we provide simulation results on a toy problem of quadratic functions with deterministic gradients. The result shows the restarting strategy on local ADAGRAD (*i.e.,* the green curve) can achieve significantly faster convergence speed, comparing to vanilla FEDAVG (*i.e.,* the blue curve) when the tolerance to the global optimal is relatively large. The fast convergence is also observed on practical federated training tasks (see Figure 2 and more discussions in Section 6).

4. Effects of Client Optimizers on Convergence and Consistency

Another important observation from the toy example in Figure 1b is that local ADAGRAD converges to a point which is quite far away from the global optimum. Even if we decay the client learning rate, this inconsistency problem still cannot be mitigated or becomes even worse. In this section, we will theoretically analyze the effect of client optimizers on the convergence and provide insights into the trade-off between convergence speed and the consistency of the solution.

FEDOPT Performs Fixed Point Iteration. When using the restarting strategy proposed in Section 3, the argument s_i representing the initial optimizer state in the operator $\mathcal{A}_i$ can be omitted since it is set to its default initial value and is the same on all clients. We further define an operator $\mathcal{A}(\boldsymbol{x}) = \sum_{i=1}^{M} w_i \mathcal{A}_i(\boldsymbol{x}; \tau_i, \Xi_i)$. Accordingly, the global update rule (see Eqn. (4)) of FEDOPT is equivalent to a stochastic fixed point iteration, shown as follows:

$$\boldsymbol{x}^{(t+1)} = (1 - \alpha)\boldsymbol{x}^{(t)} + \alpha \mathbb{E}[\mathcal{A}(\boldsymbol{x}^{(t)})]$$
$$+ \alpha(\mathcal{A}(\boldsymbol{x}^{(t)}) - \mathbb{E}[\mathcal{A}(\boldsymbol{x}^{(t)})]) \quad (5)$$

where α denotes the server learning rate and the expectation is taken over all random sources in local stochastic gradients at the current round. The deterministic version of (5) is known as Mann's iteration (Ryu & Boyd, 2016) and will converge to the fixed points of $\mathbb{E}[\mathcal{A}]$ under certain conditions. Replacing server GD with another server optimizer can accelerate the convergence but it will not influence the properties and the fixed points of the operator $\mathbb{E}[\mathcal{A}]$.

We will analyze the convergence of FEDOPT (update rule of which is (5)) in the setting where each local objective function $F_i(\boldsymbol{x})$ is L_i-Lipschitz smooth and μ_i-strongly convex. Besides, we make the following assumptions on the properties of each client optimizer $\mathcal{A}_i$ and show how the global convergence is influenced by these properties. If the client optimizer is SGD, then the contractive local operator and bounded cumulative variance properties described below follow directly from the strong convexity and smoothness of the objective function, suggesting the following assumptions are reasonable and mild.

Assumption 1 (Contractive Local Operator). *The local expected operator* $\mathbb{E}[\mathcal{A}_i]$ *is contractive, satisfying that* $\|\mathbb{E}[\mathcal{A}_i(\boldsymbol{x}; k, \Xi)] - \mathbb{E}[\mathcal{A}_i(\boldsymbol{y}; k, \Xi)]\|^2 \leq h_i(k) \cdot \|\boldsymbol{x} - \boldsymbol{y}\|^2$ *for any* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, *where* $0 < h_i(k) < 1$ *is a decreasing function of the number of local steps* k.

Assumption 2 (Bounded Cumulative Variance). *We assume the local stochastic gradient* g_i *is an unbiased estimator of* ∇F_i *and has bounded variance:* $\mathbb{E}\|g_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{x})\|^2 \leq \sigma^2$. *Similarly, the local operator* $\mathcal{A}_i$ *has bounded cumulative variance:* $\mathbb{E}\|\mathcal{A}_i(\boldsymbol{x}; k, \Xi) - \mathbb{E}[\mathcal{A}_i(\boldsymbol{x}; k, \Xi)]\|^2 \leq q_i(k) \cdot \sigma^2$, *where* $q_i(k) > 0$ *is a non-decreasing function of the number of local steps* k.

The function $h_i(k)$ in Assumption 1 measures the local training progress of CLIENTOPT. In intuition, when CLIENTOPT has faster convergence rate or takes more local steps, the output of $\mathbb{E}[\mathcal{A}_i]$ will be closer to the minimum $\boldsymbol{x}_i^*$, and hence h_i will become smaller. Besides, the function $q_i(k)$ in Assumption 2 quantifies how the noise in stochastic gradients are accumulated through k local updates. The analytical forms of h_i, q_i depend on the choice of CLIENTOPT. For example, when the client optimizer is SGD with fixed client learning rate, we formally prove in Appendix B that $h_i(k) = (1 - \eta_i \mu_i)^{2k}$ and $q_i(k) = \eta_i^2 k$. For adaptive optimizers, such as ADAGRAD and ADAM, while it is difficult to obtain the analytical form of h_i, we empirically validate Assumption 1 in Appendix B.4 and found they can yield smaller h_i values than vanilla SGD.

Theorem 1 (Convergence of FEDOPT and Minimizer Inconsistency). *Under Assumptions 1 and 2, the operator* $\mathbb{E}[\mathcal{A}]$ *is contractive and has a unique fixed point* $\widetilde{\boldsymbol{x}}$. *After total T communication rounds,* $\mathbb{E}\|\boldsymbol{x}^{(T)} - \widetilde{\boldsymbol{x}}\|^2$ *can be upper bounded as follows for some positive constant c:*

$$c \cdot \left(\left[\sum_{i=1}^{M} w_i h_i \right]^T \frac{\|\boldsymbol{x}^{(0)} - \widetilde{\boldsymbol{x}}\|^2}{1 - \sum_{i=1}^{M} w_i h_i} + \frac{1}{T} \frac{\sigma^2 \sum_{i=1}^{M} w_i^2 q_i}{(1 - \sum_{i=1}^{M} w_i h_i)^2} \right).$$
$$(6)$$

In the presence of data heterogeneity, the fixed point $\widetilde{\boldsymbol{x}}$ *of* $\mathbb{E}[\mathcal{A}]$ *is not necessarily the same as the optimum* $\boldsymbol{x}^*$ *of the global objective F. We refer to this problem as minimizer inconsistency.*

Better Client Optimizers Can Accelerate the Global Convergence. Theorem 1 provides a nice connection between local and global convergence. Specifically, if clients use better client optimizers that have smaller contraction constant h_i, then the worst-case global convergence to the fixed point $\widetilde{\boldsymbol{x}}$ can be improved, as the error bound (6) monotonically increases with h_i. Therefore, a natural strategy to speedup the global convergence is to separately minimize h_i on each client while keeping q_i unchanged. One can achieve this by either tuning a proper learning rate for each client, or using adaptive client optimizers which could reduce the

effective local condition number. Intuitively, using better client optimizers makes the operator $\mathbb{E}[\mathcal{A}]$ to be more contractive, and hence, it becomes easier to find its fixed points (as corroborated in Figure 1a, the loss surface is better conditioned when using local ADAGRAD). This acceleration effects is complementary to that of the server optimizer, which changes the form of the fixed point iteration without affecting $\mathbb{E}[\mathcal{A}]$. Therefore, one can use both better client and server optimizers to get the best performance, as we will later illustrate in Section 6.

The Minimizer Inconsistency Problem. Theorem 1 also shows that the speedup effects of client optimizers may come with a price: the fixed point $\widetilde{x}$ can be different from the optimal minimizer x^*. When the objective function is quadratic, CLIENTOPT is GD and all clients have the same learning rates η and number of local steps τ, it has been shown in (Charles & Konečný, 2021; Pathak & Wainwright, 2020) that $\|\widetilde{x} - x^*\| = \mathcal{O}(\eta(\tau - 1))$ such that one can balance the trade-off between convergence speed and the minimizer inconsistency by gradually decaying the client learning rate. However, through a simple counterexample below, we will show that *using adaptive optimizers or different hyper-parameters on clients leads to an additional gap between $\widetilde{x}$ and x^*, which does not vanish to zero with the learning rate*. This non-vanishing bias explains our empirical observations in Figure 1b and calls for new techniques to overcome it.

Quadratic Example for Theorem 1. In this toy example, we assume the loss function for client i is given by $F_i(x) = \frac{1}{2}x^\top H_i x - e_i^\top x + c_i$, where H_i is symmetric and positive definite and e_i, c_i are some vectors. Accordingly, the minimum of F_i is simply $x_i^* = H_i^{-1}e_i$ and the minimum of the global objective is $x^* = (\sum_{i=1}^M w_i H_i)^{-1}\sum_{i=1}^M w_i H_i x_i^*$.

Theorem 2. *For the quadratic problem, if the client optimizer is preconditioned GD with a fixed pre-conditioner P_i, then $\widetilde{x} = (\sum_{i=1}^M w_i[I - K_i])^{-1}\sum_{i=1}^M w_i[(I - K_i)x_i^*] \neq x^*$, where $K_i = (I - \eta_i P_i H_i)^{\tau_i}$. If we let $\eta_i = \gamma_i \eta$ and η approach to zero, then it follows that*

$$\widetilde{x} = (\sum_{i=1}^M w_i \gamma_i \tau_i P_i H_i)^{-1}(\sum_{i=1}^M w_i \gamma_i \tau_i P_i H_i x_i^*) \neq x^*.$$

Theorem 2 shows that as long as one of the factors (*i.e.*, client learning rate, the number of local steps, the preconditioners) is different across clients, then there would appear a non-vanishing gap between $\widetilde{x}$ and x^*. In order to get an intuition behind this phenomenon, we can check the exact expression of $x^* - \mathcal{A}(x^*)$. If $\widetilde{x} = x^*$, then $x^* - \mathcal{A}(x^*)$ should always be zero. However, we get the following via Taylor approximation:

$$\frac{x^* - \mathcal{A}(x^*)}{\max_i \eta_i \tau_i} = \sum_{i=1}^M w_i \frac{\eta_i \tau_i P_i}{\max_i \eta_i \tau_i}\nabla F_i(x^*) + \mathcal{O}(\eta_i \tau_i) \quad (7)$$

where the first term (on the right hand side) corresponds to the non-vanishing gap between $\widetilde{x}$ and x^* as it can be treated

as a skewed version of the global gradient $\nabla F(x^*) = \sum_{i=1}^M w_i \nabla F_i(x^*) = 0$, and the second term can be omitted when the client learning rate is sufficiently small. It is worth noting that if $\eta_i = 0$ or $\tau_i = 0$, then $x = \mathcal{A}(x)$ for any $x \in \mathbb{R}^d$, as there is no optimization progress at all. Therefore, in order to exclude this invalid solution, we divide $(\max_i \eta_i \tau_i)$ in (7).

Connections with Previous Works on Minimizer Inconsistency. When client learning rates, number of local steps are the same across all clients, and non-adaptive, deterministic CLIENTOPT are used, $\|\widetilde{x} - x^*\|$ can vanish to zero along with the learning rates. This phenomenon has been observed and analyzed by few recent literature in different forms, see (Charles & Konečný, 2021; Malinovskiy et al., 2020; Pathak & Wainwright, 2020). Theorem 1 generalizes these results by allowing heterogeneous local hyper-parameters and adaptive, stochastic client optimizers. In addition, the non-vanishing bias was studied in (Wang et al., 2020a) by assuming different local learning rates and local steps at clients. In this paper, we further generalize the results by showing that even when the learning rates and local steps are the same, using local adaptive methods will lead to a non-vanishing gap. We summarize the differences in Table 6 of Appendix D.

5. Correction Techniques to Overcome the Non-vanishing Solution Bias

Local Correction: Reweighting Local Gradients. From (7) in the quadratic example, one can observe that the non-vanishing minimizer inconsistency comes from the fact that FEDOPT *implicitly and improperly* weights each local gradients by a matrix $N_i = \eta_i \tau_i P_i$. So a natural solution to overcome the non-vanishing bias is to normalize the local model changes before aggregating them on the server. Instead of sending $x - \mathcal{A}_i(x) = N_i \nabla F_i(x) + \mathcal{O}(\eta_i^2)$ to the server, each client can send the locally normalized version $N_i^{-1}(x - \mathcal{A}_i(x))$. This simple change can ensure that the first term in (7) is always zero no matter which value is the client learning rate. As a consequence, there is no non-vanishing solution bias. The local correction technique also automatically avoids the invalid solution $x = \mathcal{A}(x), \forall x$ by dividing the client learning rate.

The local correction technique can be extended to more general settings and work with any adaptive optimizers, such as ADAGRAD, ADAM. If the local model changes of a specific client optimizer can be written as $x - \mathcal{A}_i(x; \tau_i) = \eta_i \sum_{k=0}^{\tau_i - 1} B_i^{(k)}\nabla F_i(x^{(k)})$ where $B_i^{(k)}$ is symmetric and positive-definite and $x^{(k)}$ denotes the k-th iterate during local updates, then we can choose the local correction ma-

trix to be $\boldsymbol{N}_i = \eta_i \sum_{k=0}^{\tau_i - 1} \boldsymbol{B}_i^{(k)}$ such that

$$\sum_{i=1}^{M} w_i \boldsymbol{N}_i^{-1} (\boldsymbol{x}^* - \mathcal{A}_i(\boldsymbol{x}^*))$$
$$= \underbrace{\sum_{i=1}^{M} w_i \nabla F_i(\boldsymbol{x}^*)}_{=\nabla F(\boldsymbol{x}^*)=0} + \mathcal{O}(\eta_i \tau_i) = \mathcal{O}(\eta_i \tau_i). \quad (8)$$

If the client optimizer is ADAGRAD, then $\boldsymbol{B}_i^{(k)}$ is just the pre-conditioner $\boldsymbol{P}_i^{(k)}$ used at the k-th local iteration; if the client optimizer is ADAM, then $\boldsymbol{B}_i^{(k)}$ is a weighted summation of local pre-conditioners. In Appendix A.3, we provide the pseudo-code of finding the expression of $\boldsymbol{N}_i$ for common optimizers; and in Appendix F, we formally prove that local correction can help FEDOPT with deterministic client optimizers converge to the stationary points of the original objective function (even when it is non-convex). For stochastic client optimizers, we empirically validate the effectiveness of local correction through extensive experiments in Section 6.

Global Correction: Preserving the Fast Convergence. Equation (8) shows that using the local correction technique may change the scale of the aggregated model updates. In particular, the first-order approximation of the aggregated model updates just equals to plain gradients (the first term in (8)) and may lose the local pre-conditioning effects. In order to address this problem, one can use either adaptive server optimizer that are more robust to the scale of its inputs, or a novel global correction technique that can help to preserve the scale of local updates. To be specific, in global correction, the server uses $\boldsymbol{N}_s^{-1} \sum_{i=1}^{M} w_i \boldsymbol{N}_i^{-1}[\boldsymbol{x} - \mathcal{A}_i(\boldsymbol{x})]$ as the pseudo-gradient of SERVEROPT, where $\boldsymbol{N}_s$ is given as $\boldsymbol{N}_s = \sum_{i=1}^{M} w_i \boldsymbol{N}_i^{-1}$. We name this technique as global correction, because it is applied on the server side. In order to obtain $\boldsymbol{N}_s$, the clients need to send the local correction matrices $\boldsymbol{N}_i$ to the server. But the server does not need to broadcast the correction matrix $\boldsymbol{N}_s$ back. Therefore, if matrices $\{\boldsymbol{N}_i\}$ are diagonal, then the communication cost per round of using global correction is only $1.5\times$ than that of without using it.

6. Experiments

We focus on three language-related tasks (Reddi et al., 2020), which have the favored sparse structures for adaptive methods: (i) Next word prediciton using RNN on Stack Overflow (SO NWP); (ii) next character prediction using RNN on Shakespeare (Shakes. NCP); (iii) tag prediction using linear regression on Stack Overflow (SO TP). For the first two tasks, we report the validation/test accuracy. For SO TP task, we report the validation/test Recall@5. Moreover, we also evaluate the methods in image classification task on the CIFAR100 dataset (Krizhevsky, 2009). The detailed descriptions (hyper-parameter tuning ranges and choices) of these federated training tasks are provided in Appendix A.
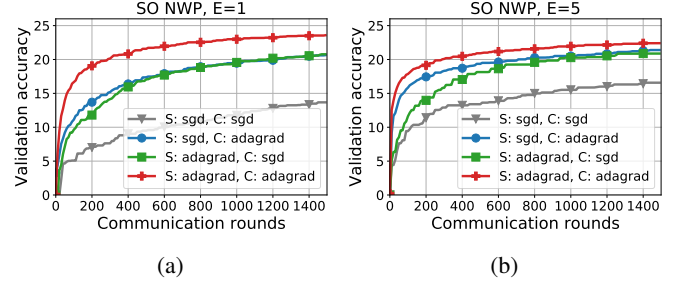

Figure 2. Example of training curves of FEDOPT variants on the SO NWP task. The adaptive client optimizer is used with the proposed restarting strategy. In the legend, "C" and "S" denote the CLIENTOPT and the SERVEROPT, respectively. Using ADAGRAD on both clients and server is significantly faster than other variants.

Our implementation is based on the Tensorflow Federated (TFF) package.

Faster Convergence. In Figure 2, we first compare the training curves of different ways of using ADAGRAD in FEDOPT on the SO NWP training task. One can either use ADAGRAD on clients, or on the server, or on both. It can be observed that using local ADAGRAD can significantly speedup the convergence compared to using vanilla SGD as the client optimizer. In particular, client-only ADAGRAD is even slightly faster than server-only ADAGRAD. When the local epoch increases from 1 to 5, the improvement of client-only ADAGRAD over server-only ADAGRAD becomes more obvious. If we apply ADAGRAD on both clients and the server, then FEDOPT achieves the fastest convergence (about $4\times$ faster than server-only ADAGRAD to achieve 20% validation accuracy in Figure 2a).

Higher Final Accuracy. We further report the test accuracy or Recall@5 on different training tasks in Table 1. Similar to the above discussions on faster convergence, changing CLIENTOPT from SGD to ADAGRAD and using the restarting strategy prposed in section 3 consistently improves the test accuracy on multiple training tasks. And applying ADAGRAD on both clients and the server achieves the highest test accuracy. For example, when the SERVEROPT is ADAGRAD, the test accuracy improves from 21.80% to 24.40% on SO NWP, and from 57.68% to 57.85% on Shakes. NCP. However, note that on the SO TP task, server ADAGRAD plus client ADAGRAD with the restarting strategy performs worse than server-only ADAGRAD (65.79 versus 66.39). This performance degradation may come from the non-vanishing solution bias, as we discussed in Section 4.

Then, we evaluate the proposed correction techniques in Table 2 and compare the results with the best server-only adaptive methods on each training tasks (the best server optimizer is selected based on the results in (Reddi et al., 2020)). On SO TP, by overcoming the non-vanishing bias, client ADAGRAD with local correction can achieve much higher
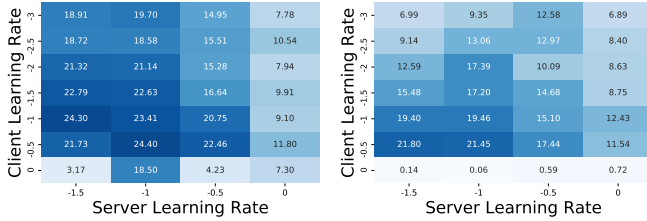
	C:SGD	C:ADAGRAD
S:SGD	14.19	21.68
S:ADAGRAD	21.80	24.40

(a) SO NWP

	C:SGD	C:ADAGRAD
S:SGD	57.12	57.47
S:ADAGRAD	57.68	57.85

(b) Shakes. NCP

	C:SGD	C:ADAGRAD
S:SGD	34.50	39.67
S:ADAGRAD	66.39	65.79

(c) SO TP

Table 1. The test accuracy (%) and recall@5 ($\times 100$) after 1500 communication rounds of FEDOPT variants on different training tasks. Darker color means better performance.

Training Tasks	SERVEROPT	CLIENTOPT			
		SGD	ADAGRAD	+ Local Cor.	+ Joint Cor.
SO NWP	ADAM	24.40	24.70	24.81	**24.85**
Shakes. NCP	ADAGRAD	57.68	57.85	57.75	**58.06**
SO TP	ADAGRAD	66.39	65.79	**67.04**	66.94

Table 2. Comparison of our proposed methods (column 2,3,4) and the best server-only adaptive methods (column 1) for each training task. The table shows the test accuracy (%) or recall@5 ($\times 100$) after 1500 communication rounds. Bolded ones are the best results for each training tasks. In the table, we fix $\epsilon = 10^{-7}$ in CLIENTOPT and tune ϵ_s in SERVEROPT. Therefore, the performance of our proposed methods (column 2,3,4) can be further improved by tuning the ϵ parameter.

Client Learning Rate vs Server Learning Rate:

	-1.5	-1	-0.5	0
3	18.91	19.70	14.95	7.78
2.5	18.72	18.58	15.51	10.54
2	21.32	21.14	15.28	7.94
1.5	22.79	22.63	16.64	9.91
1	24.30	23.41	20.75	9.10
0.5	21.73	24.40	22.46	11.80
0	3.17	18.50	4.23	7.30

(a) S: ADAGRAD, C: ADAGRAD

	-1.5	-1	-0.5	0
3	6.99	9.35	12.58	6.89
2.5	9.14	13.06	12.97	8.40
2	12.59	17.39	10.09	8.63
1.5	15.48	17.20	14.68	8.75
1	19.40	19.46	15.10	12.43
0.5	21.80	21.45	17.44	11.54
0	0.14	0.06	0.59	0.72

(b) S: ADAGRAD, C: SGD

Figure 3. How the test accuracy changes with server and client learning rate in FEDOPT. Using local adaptive methods make the algorithm be more robust to the hyper-parameter changes.

Recall@5 than vanilla SGD optimizer (67.04 versus 66.39). Similar improvements also appear in SO NWP and Shakes. NCP tasks. Besides client ADAGRAD, our proposed methods also work for other common momentum-based adaptive optimizers (such as ADAM and YOGI (Zaheer et al., 2018)). We present the experimental results on SO NWP in Table 3. It can be observed that our proposed method relatively improves the test accuracy of server ADAM plus client SGD by 3.9% (25.35% versus 24.40%), and vanilla FEDAVG by 78.6% (25.35% versus 14.19%).

Less Sensitive to Hyper-parameter Changes. In Figure 3, we report how the test accuracy changes with server and client learning rates on the SO NWP task. We observe that for server-only ADAGRAD, there is only two out of 28 learning rate combinations that can achieve a 20%+ test accuracy. On the other hand, using ADAGRAD on both clients and server are more robust to the learning rate changes. There are 10 out of 28 combinations reaching a 20%+ test accuracy. The less sensitivity to learning rate changes can help people to save hyper-parameter tuning time in practice.

Comparison with the Synchronizing States Strategy. We further compare the proposed restarting optimizer states strategy to a synchronizing strategy where local pre-conditioners are synchronized after each round. Surprisingly, the additional communication in the synchronized strategy does not help and achieves even worse accuracy. For instance, on Shakes. NCP, server ADAGRAD plus synchronized client ADAGRAD has a test accuracy of 56.89%, which is lower than that of the restarting one (57.85%).

Results on Image Classification Task. At last, we examine the performance of the proposed methods on pathological version of CIFAR100, following the setup in (Reddi et al., 2020). The observations are similar to language related tasks. Naively using client adaptive methods in FEDOPT is not guaranteed to have better performance. But when combining local adaptvity with the proposed correction techniques, the performance consistently improves. Specifically, after 8000 communication rounds, while server ADAM plus client ADAGRAD has worse test accuracy than server-only ADAM (54.68% vs 56.64%), using local correction can improve the accuracy to 56.80%.

7. Conclusions

In this paper, we first propose techniques that enable the use of adaptive optimization methods for local updates at clients in federated learning. Through the analysis on effects of client optimizers for smooth and strongly convex functions, we show that although local adaptive methods can accelerate the convergence in some scenarios, it introduces an additional non-vanishing gap between the converged point and the optimal solution. To mitigate the side effects of using local adaptivity, we further propose local and global correction techniques. We verified the advantages of the proposed methods regarding fast convergence, better test accuracy with extensive experiments on benchmark datasets.

CLIENTOPT	No Cor.	Local Cor.	Joint Cor.
YOGI (Zaheer et al., 2018)	24.80	25.29	**25.33**
ADAM	24.86	25.15	**25.35**

Table 3. Performance (test accuracy (%) after 1500 rounds of training) of momentum-based adaptive client optimizers on SO NWP. The SERVEROPT is fixed as ADAM. Recall that the test accuracy of sever ADAM plus client SGD is 24.40%.

References

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Brisimi, T. S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. C., and Shi, W. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

Charles, Z. and Konečný, J. Convergence and accuracy trade-offs in federated learning and meta-learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2575–2583. PMLR, 2021.

Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Hsu, T.-M. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Karimireddy, S. P., Jaggi, M., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020.

Khaled, A., Mishchenko, K., and Richtárik, P. Tighter theory for local SGD on identical and heterogeneous data. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koloskova, A., Loizou, N., Boreiri, S., Jaggi, M., and Stich, S. U. A unified theory of decentralized SGD with changing topology and local updates. In *International Conference on Machine Learning*, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Malinovskiy, G., Kovalev, D., Gasanov, E., Condat, L., and Richtarik, P. From local sgd to local fixed-point methods for federated learning. In *International Conference on Machine Learning*, pp. 6692–6701. PMLR, 2020.

Malinovsky, G., Kovalev, D., Gasanov, E., Condat, L., and Richtarik, P. From local SGD to local fixed point methods for federated learning. *arXiv preprint arXiv:2004.01442*, 2020.

Martens, J. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

McMahan, H. B. and Streeter, M. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*, 2010.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Pathak, R. and Wainwright, M. J. FedSplit: An algorithmic framework for fast federated optimization. *arXiv preprint arXiv:2005.05238*, 2020.

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

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Ryu, E. K. and Boyd, S. Primer on monotone operator methods. *Appl. Comput. Math*, 15(1):3–43, 2016.

Stich, S. U. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*, 2019.

Tong, Q., Liang, G., and Bi, J. Effective federated adaptive gradient methods with non-iid decentralized data. *arXiv preprint arXiv:2009.06557*, 2020.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems*, 33, 2020a.

Wang, J., Tantia, V., Ballas, N., and Rabbat, M. SlowMo: Improving communication-efficient distributed SGD with slow momentum. In *Proceedings of the International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=SkxJ8REYPH.

Woodworth, B., Patel, K. K., and Srebro, N. Minibatch vs local sgd for heterogeneous distributed learning. *arXiv preprint arXiv:2006.04735*, 2020.

Xie, C., Koyejo, O., Gupta, I., and Lin, H. Local AdaAlter: Communication-efficient stochastic gradient descent with adaptive learning rates. *arXiv preprint arXiv:1911.09030*, 2019.

Xu, J., Glicksberg, B. S., Su, C., Walker, P., Bian, J., and Wang, F. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, pp. 1–19, 2020.

Zaheer, M., Reddi, S., Sachan, D., Kale, S., and Kumar, S. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*, 31:9793–9803, 2018.

Zhuang, J., Tang, T., Ding, Y., Tatikonda, S. C., Dvornek, N., Papademetris, X., and Duncan, J. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in Neural Information Processing Systems*, 33, 2020.

A. Experimental Details

In this section, we describe our experimental setup in more detail. A brief summary of tunning ranges and default values for hyper-parameters are summarized in Table 4.

Tasks	SO NWP	Shakes. NCP	SO TP
Models	RNN	RNN	LR
Total number of clients	342477	715	342477
Maximal elements per client	128	N/A	1000
Active clients per round	50	10	10
Mini-batch size	16	4	100
Default Local epochs	1	1	1
Total rounds	1500	1500	1500
Client learning rate $\log_{10}(\eta)$	$\{-3, -2.5, \ldots, -1, 0, 0.5\}$		$\{-1, -0.5, \ldots, 2, 2.5\}$
Server learning rate $\log_{10}(\alpha)$	$\{-1.5, -1, -0.5, 0, 0.5, 1\}$		$\{-0.5, 0, 0.5, 1, 1.5\}$
Parameter ϵ_s in SERVEROPT		$\log_{10}(\epsilon_s) = \{-7, -5, -3, -1\}$	
Parameter ϵ in CLIENTOPT		$\log_{10}(\epsilon) = \{-7\}$	

Table 4. Experimental settings, default values and tuning ranges of hyper-parameters.

A.1. Training Tasks

Next-Word Prediction on Stack Overflow (SO NWP for short). Stack Overflow is a language modeling dataset from the question and answer site, Stack Overflow. The datasets consists of questions and answers from $342,477$ unique users, each of which is treated as a client in our experiments. We perform next-word prediction task on this dataset and restrict to the 10,000 most frequently used words. The preprocess procedure of this datasets follows (Reddi et al., 2020). Specifically, we let each client only use the first 128 sentences of its local dataset, in order to avoid that clients have extremely different amount of data. Padding and truncation are used to ensure that sentences have 20 words. The metrics we report are the top-1 validation and test accuracy over the $10,000$-word vocabulary. It does not include padding, out-of-vocab, or beginning or end or sentence tokens. The neural network model we use is a RNN with single LSTM layer, which is the same as that of (Reddi et al., 2020).

Tag Prediction on Stack Overfolow (SO TP for short). Tag prediction via logistic regression is another training task on the Stack Overflow datasets. It has the same number of clients and vocabulary size as SO NWP. Besides, following the setup in (Reddi et al., 2020), we use the 500 most frequent tags and a one-versus-rest classification strategy.

Next Character Prediction on Shakespeare (Shakes. NCP for short). Shakespeare is another language modeling dataset built from the works from William Shakespeare. Each client corresponds to a speaking role with at least two lines. The lines of each speaking role is splitted into sequences of 80 characters, padding if necessary. The vocabulary size is 90. The neural networks model we use is the same as (Reddi et al., 2020): a RNN with two LSTM layers.

A.2. Best Performing Hyper-parameters

We report the best performed hyper-parameters in Table 5. The hyper-parameters are selected in a way such that the average validation accuracy (or recall@5) over the last 100 rounds achieves the highest value.

A.3. Pseudo-Codes of The Proposed Algorithms

We present the pseudo-codes for our proposed algorithms in Algorithm 1. It is worth noting that Algorithm 1 only provides a concrete specification of the proposed local and global correction techniques. Beyond our choices, there may exist many other algorithmic variants.

Training Tasks	SERVEROPT	CLIENTOPT			
		SGD	ADAGRAD	+ Local Cor.	+ Joint Cor.
SO NWP	SGD	$(-0.5, 0, \text{NA})$	$(-0.5, 0, \text{NA})$	$(0, 0.5, \text{NA})$	$(-0.5, 0, \text{NA})$
	ADAGRAD	$(-0.5, -1.5, -3)$	$(-0.5, -1, -5)$	$(-0.5, -1, -5)$	$(-0.5, -1, -7)$
	ADAM	$(-0.5, -1.5, -5)$	$(-0.5, -1.5, -3)$	$(-1, -1.5, -5)$	$(-1, -1.5, -5)$
Shakes. NCP	SGD	$(0, 0, \text{NA})$	$(-0.5, -0.5, \text{NA})$	$(-0.5, 2, \text{NA})$	$(-0.5, 0, \text{NA})$
	ADAGRAD	$(0.5, -1, -1)$	$(0, -1, -3)$	$(-0.5, -1, -5)$	$(-0.5, -0.5, -1)$
SO TP	SGD	$(2.5, 0, \text{NA})$	$(1.5, 0, \text{NA})$	$(1.5, 1.5, \text{NA})$	$(1.5, 0, \text{NA})$
	ADAGRAD	$(0.5, 1, -5)$	$(-0.5, 1, -5)$	$(-0.5, 1, -7)$	$(-0.5, 1, -5)$
	ADAM	$(1.5, -0.5, -5)$	$(1, -0.5, -5)$	$(0.5, -0.5, -7)$	$(1, -0.5, -5)$

Table 5. Best performed hyper-parameters in all training tasks. The tuple in each cell corresponds to $(\eta, \alpha, \epsilon_s)$: client learning rate, server learning rate, and adaptivity parameter ϵ_s in the SERVEROPT.

Algorithm 1 Local adaptive FEDOPT with the restarting strategy and local correction , global correction techniques

---

1: **Input:** Initial model $\boldsymbol{x}^{(0)}$, CLIENTOPT, SERVEROPT
2: **for** $t \in \{0, 1, \ldots, T-1\}$ **do**
3:     **for** client i in the random set $\mathcal{S}^{(t)}$ **in parallel do**
4:         Initialization $\boldsymbol{x}_i^{(t,0)} = \boldsymbol{x}^{(t)}$, $\boldsymbol{m}_i^{(t,-1)} = 0, \boldsymbol{P}_i^{(t,-1)} = c$, $\boldsymbol{N}_i^{(t)} = 0, \boldsymbol{M}_i^{(t,-1)} = 0$
5:         **for** $k \in \{0, \ldots, \tau_i - 1\}$ **do**
6:             Compute local stochastic gradient $g_i(\boldsymbol{x}_i^{(t,k)})$ and update local pre-conditioner $\boldsymbol{P}_i^{(t,k)}$
7:             Update momentum $\boldsymbol{m}_i^{(t,k)} = \beta_1 \boldsymbol{m}_i^{(t,k-1)} + (1-\beta_1) g_i(\boldsymbol{x}_i^{(t,k)})$
8:             Update local model $\boldsymbol{x}_i^{(t,k+1)} = \boldsymbol{x}_i^{(t,k)} - \eta \boldsymbol{P}_i^{(t,k)} \boldsymbol{m}_i^{(t,k)}$
9:             $\boldsymbol{M}_i^{(t,k)} = \beta_1 \boldsymbol{M}_i^{(t,k-1)} + (1-\beta_1) \boldsymbol{P}_i^{(t,k)}$
10:            $\boldsymbol{N}_i^{(t)} \leftarrow \boldsymbol{N}_i^{(t)} + \boldsymbol{M}_i^{(t,k)}$
11:         **end for**
12:         Local changes $\Delta_i^{(t)} = \boldsymbol{x}^{(t)} - \boldsymbol{x}_i^{(t,\tau_i)}$
13:         $\Delta_i^{(t)} \leftarrow (\boldsymbol{N}_i^{(t)})^{-1} \Delta_i^{(t)}$
14:     **end for**
15:     Aggregate $\Delta^{(t)} = \frac{1}{|\mathcal{S}^{(t)}|} \sum_{i \in \mathcal{S}^{(t)}} \Delta_i^{(t)}$
16:     Aggregate $\boldsymbol{N}_s^{(t)} = \frac{1}{|\mathcal{S}^{(t)}|} \sum_{i \in \mathcal{S}^{(t)}} (\boldsymbol{N}_i^{(t)})^{-1}$ and set $\Delta^{(t)} \leftarrow (\boldsymbol{N}_s^{(t)})^{-1} \Delta^{(t)}$
17:     $\boldsymbol{x}^{(t+1)} = \text{SERVEROPT}(\boldsymbol{x}^{(t)}, \Delta^{(t)}, \alpha, t)$
18: **end for**

B. Justifications of Assumptions 1 and 2 for Vanilla SGD Client Optimizer

B.1. Two Useful Lemmas

Before we dive into the proofs, we would like to first introduce two useful lemmas, which will be repeatedly applied later on.

Lemma 1. *Suppose function F is twice-differentiable. Then, for any points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, we have*

$$\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y}) = \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y}) \tag{9}$$

where $\boldsymbol{H} = \int_0^1 \nabla^2 F(\boldsymbol{y} + s(\boldsymbol{x} - \boldsymbol{y})) \mathrm{d}s$.

Proof. Due to the linearity of integral, we have

$$\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y}) = \int_0^1 \nabla^2 F(\boldsymbol{y} + s(\boldsymbol{x} - \boldsymbol{y}))(\boldsymbol{x} - \boldsymbol{y}) \mathrm{d}s = \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y}). \tag{10}$$

$\square$

Lemma 2. *Suppose $\boldsymbol{x}$ and $\boldsymbol{y}$ are random variables with the same random sources ξ. Then, if a function F is L-lipschitz smooth and μ-strongly convex, then we have*

$$\left\|\mathbb{E}[\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})]\right\|^2 \geq \mu^2 \left\|\mathbb{E}[\boldsymbol{x} - \boldsymbol{y}]\right\|^2 \tag{11}$$

$$\left\|\mathbb{E}[\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})]\right\|^2 \leq L \left\langle \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}], \mathbb{E}[\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})] \right\rangle \tag{12}$$

$$(\mu + L) \left\langle \mathbb{E}[\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})], \mathbb{E}[(\boldsymbol{x} - \boldsymbol{y})] \right\rangle \geq \left\|\mathbb{E}[\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})]\right\|^2 + \mu L \left\|\mathbb{E}[\boldsymbol{x} - \boldsymbol{y}]\right\|^2 \tag{13}$$

Proof. Based on lemma 1, we have $\mathbb{E}[\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})] = \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})]$. And hence,

$$\left\|\mathbb{E}[\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})]\right\|^2 = \left\langle \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})], \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})] \right\rangle \tag{14}$$

$$= \mathbb{E} \left\langle \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y}), \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})] \right\rangle. \tag{15}$$

Since $\mu \preccurlyeq \boldsymbol{H} \preccurlyeq L$, it follows that

$$\mu \left\langle \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}], \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})] \right\rangle \leq \mathbb{E} \left\langle \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y}), \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})] \right\rangle \leq L \left\langle \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}], \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})] \right\rangle. \tag{16}$$

Combining (15) and (16), we prove the second inequality in Lemma 2. Furthermore, note that

$$\left\langle \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}], \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})] \right\rangle = \mathbb{E} \left\langle \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}], \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y}) \right\rangle \geq \mu \mathbb{E} \left\langle \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}], (\boldsymbol{x} - \boldsymbol{y}) \right\rangle. \tag{17}$$

We have

$$\left\langle \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})], \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})] \right\rangle \geq \mu^2 \left\langle \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}], \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}] \right\rangle. \tag{18}$$

This completes the proof of the first inequality in Lemma 2. Using the similar technique as above, one can prove that

$$\left\|\mathbb{E}[H(\boldsymbol{x} - \boldsymbol{y}) - \mu(\boldsymbol{x} - \boldsymbol{y})]\right\|^2 \leq (L - \mu) \left\langle \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}], \mathbb{E}[(\boldsymbol{H} - \mu \boldsymbol{I})(\boldsymbol{x} - \boldsymbol{y})] \right\rangle. \tag{19}$$

Besides, note that

$$\left\|\mathbb{E}[H(\boldsymbol{x} - \boldsymbol{y}) - \mu(\boldsymbol{x} - \boldsymbol{y})]\right\|^2 = \left\|\mathbb{E}[H(\boldsymbol{x} - \boldsymbol{y})]\right\|^2 + \mu^2 \left\|\mathbb{E}[\boldsymbol{x} - \boldsymbol{y}]\right\|^2$$
$$- 2\mu \left\langle \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})], \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}] \right\rangle. \tag{20}$$

Substituting (20) into (19) and rearranging, we obtain that

$$(L + \mu) \left\langle \mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})], \mathbb{E}[\boldsymbol{x} - \boldsymbol{y}] \right\rangle \geq \left\|\mathbb{E}[\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})]\right\|^2 + \mu L \left\|\mathbb{E}[\boldsymbol{x} - \boldsymbol{y}]\right\|^2. \tag{21}$$

Here we complete the proof of the last inequality in Lemma 2. $\square$

B.2. Proof of Assumption 1 for Vanilla SGD Client Optimizer

For the ease of writing, we denote $\boldsymbol{x}^{(k)} = \mathcal{A}_i(\boldsymbol{x}; k, \Xi)$ and $\boldsymbol{y}^{(k)} = \mathcal{A}_i(\boldsymbol{y}; k, \Xi)$ where $\Xi = \{\xi^{(0)}, \ldots, \xi^{(k-1)}\}$ represents the random sources through k local steps. According to the update rule of SGD, we have

$$\mathbb{E}[\boldsymbol{x}^{(k+1)}] - \mathbb{E}[\boldsymbol{y}^{(k+1)}] = \mathbb{E}[\boldsymbol{x}^{(k)} - \eta g_i(\boldsymbol{x}^{(k)})] - \mathbb{E}[\boldsymbol{y}^{(k)} - \eta g_i(\boldsymbol{y}^{(k)})] \tag{22}$$

$$= \mathbb{E}[\boldsymbol{x}^{(k)} - \boldsymbol{y}^{(k)}] - \eta_i \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)}) - \nabla F_i(\boldsymbol{y}^{(k)})]. \tag{23}$$

We are going to use *induction* to prove $\left\|\mathbb{E}[\boldsymbol{x}^{(k)} - \boldsymbol{y}^{(k)}]\right\|^2 \leq (1 - \eta_i \mu_i)^{2k} \left\|\boldsymbol{x} - \boldsymbol{y}\right\|^2$. When $k = 1$, we have

$$\left\|\mathbb{E}[\boldsymbol{x}^{(1)} - \boldsymbol{y}^{(1)}]\right\|^2 = \left\|\boldsymbol{x} - \boldsymbol{y} - \eta_i \nabla F_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{y})\right\|^2 \tag{24}$$

$$= \left\|\boldsymbol{x} - \boldsymbol{y}\right\|^2 + \eta_i^2 \left\|\nabla F_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{y})\right\|^2 - 2\eta_i \left\langle \boldsymbol{x} - \boldsymbol{y}, \nabla F_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{y}) \right\rangle \tag{25}$$

$$\leq \left(1 - \frac{2\mu_i L_i}{\mu_i + L_i}\right) \left\|\boldsymbol{x} - \boldsymbol{y}\right\|^2 + \left(\eta_i^2 - \frac{2\eta_i}{\mu_i + L_i}\right) \left\|\nabla F_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{y})\right\|^2 \tag{26}$$

where the last inequality comes from the Lipschitz smoothness and strongly convexity of F_i. When the client learning rate satisfies $\eta_i(\mu_i + L_i) < 2$, we have

$$\left\| \mathbb{E}[\boldsymbol{x}^{(1)} - \boldsymbol{y}^{(1)}] \right\|^2 \leq \left(1 - \frac{2\mu_i L_i}{\mu_i + L_i}\right) \|\boldsymbol{x} - \boldsymbol{y}\|^2 + \left(\eta_i^2 \mu_i^2 - \frac{2\mu_i \eta_i}{\mu_i + L_i}\right) \|\boldsymbol{x} - \boldsymbol{y}\|^2 \tag{27}$$

$$= (1 - \eta_i \mu_i)^2 \|\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{28}$$

So Assumption 1 is true when $k = 1$. Now we assume that Assumption 1 holds for some $k > 1$ and examine the value of $\left\| \mathbb{E}[\boldsymbol{x}^{(k+1)} - \boldsymbol{y}^{(k+1)}] \right\|$. In particular, we have

$$\left\| \mathbb{E}[\boldsymbol{x}^{(k+1)} - \boldsymbol{y}^{(k+1)}] \right\|^2 = \left\| \mathbb{E}[\boldsymbol{x}^{(k)} - \boldsymbol{y}^{(k)}] \right\|^2 + \eta_i^2 \left\| \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)}) - \nabla F_i(\boldsymbol{y}^{(k)})] \right\|^2$$
$$- 2\eta \left\langle \mathbb{E}[\boldsymbol{x}^{(k)} - \boldsymbol{y}^{(k)}], \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)}) - \nabla F_i(\boldsymbol{y}^{(k)})] \right\rangle. \tag{29}$$

According to (11) and (13) of Lemma 2, we have

$$\left\| \mathbb{E}[\boldsymbol{x}^{(k+1)} - \boldsymbol{y}^{(k+1)}] \right\|^2 \leq \left(1 - \frac{2\eta_i \mu_i L_i}{\mu_i L_i}\right) \left\| \mathbb{E}[\boldsymbol{x}^{(k)} - \boldsymbol{y}^{(k)}] \right\|^2$$
$$+ \left(\eta_i^2 - \frac{2\eta_i}{\mu_i + L_i}\right) \left\| \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)}) - \nabla F_i(\boldsymbol{y}^{(k)})] \right\|^2 \tag{30}$$

$$\leq \left(1 - \frac{2\eta_i \mu_i L_i}{\mu_i L_i}\right) \left\| \mathbb{E}[\boldsymbol{x}^{(k)} - \boldsymbol{y}^{(k)}] \right\|^2$$
$$+ \left(\eta_i^2 \mu_i^2 - \frac{2\eta_i \mu_i^2}{\mu_i + L_i}\right) \left\| \mathbb{E}[\boldsymbol{x}^{(k)} - \boldsymbol{y}^{(k)}] \right\|^2 \tag{31}$$

$$\leq (1 - \eta_i \mu_i)^2 \left\| \mathbb{E}[\boldsymbol{x}^{(k)} - \boldsymbol{y}^{(k)}] \right\|^2 \tag{32}$$

$$\leq (1 - \eta_i \mu_i)^{2(k+1)} \|\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{33}$$

Here we complete the induction procedure and prove that Assumption 1 holds for vanilla SGD client optimizer and $h_i(k) = (1 - \eta_i \mu_i)^{2k}$.

B.3. Proof of Assumption 2 for Vanilla SGD Client Optimizer

For the ease of writing, we define $\boldsymbol{x}^{(k)} = \mathcal{A}_i(\boldsymbol{x}; k)$ and $\overline{\boldsymbol{x}}^{(k)} = \mathbb{E}[\boldsymbol{x}^{(k)}]$. We are going to use induction to prove that

$$\mathbb{E}\left\| \boldsymbol{x}^{(k)} - \overline{\boldsymbol{x}}^{(k)} \right\|^2 \leq k \eta_i^2 \sigma^2. \tag{34}$$

When $k = 1$, we have

$$\mathbb{E}\left\| \boldsymbol{x}^{(1)} - \overline{\boldsymbol{x}}^{(1)} \right\|^2 = \mathbb{E}\left\| -\eta_i g_i(\boldsymbol{x}; \xi) + \eta_i \nabla F_i(\boldsymbol{x}) \right\|^2 \leq \eta_i^2 \sigma^2. \tag{35}$$

We assume that Assumption 2 holds for some $k > 1$. Then, according to the update rule of SGD, we have

$$\mathbb{E}\left\|\boldsymbol{x}^{(k+1)} - \mathbb{E}[\boldsymbol{x}^{(k+1)}]\right\|^2$$

$$=\mathbb{E}\left\|\boldsymbol{x}^{(k)} - \eta_i g_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\boldsymbol{x}^{(k)}] + \eta_i \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\|^2 \tag{36}$$

$$=\mathbb{E}\left\|-\eta_i g_i(\boldsymbol{x}^{(k)}) + \eta_i \nabla F_i(\boldsymbol{x}^{(k)}) + \boldsymbol{x}^{(k)} - \eta_i \nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\boldsymbol{x}^{(k)}] + \eta_i \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\|^2 \tag{37}$$

$$=\eta_i^2 \mathbb{E}\left\|g_i(\boldsymbol{x}^{(k)}) - \nabla F_i(\boldsymbol{x}^{(k)})\right\|^2 + \mathbb{E}\left\|\boldsymbol{x}^{(k)} - \eta_i \nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\boldsymbol{x}^{(k)}] + \eta_i \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\|^2 \tag{38}$$

$$\leq \eta_i^2 \sigma^2 + \mathbb{E}\left\|\boldsymbol{x}^{(k)} - \eta_i \nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\boldsymbol{x}^{(k)}] + \eta_i \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\|^2 \tag{39}$$

$$=\eta_i^2 \sigma^2 + \mathbb{E}\left\|\boldsymbol{x}^{(k)} - \mathbb{E}[\boldsymbol{x}^{(k)}]\right\|^2 + \eta_i^2 \mathbb{E}\left\|\nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\|^2$$
$$- 2\eta_i \left\langle \boldsymbol{x}^{(k)} - \mathbb{E}[\boldsymbol{x}^{(k)}], \nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\rangle \tag{40}$$

$$\leq (k+1)\eta_i^2 \sigma^2 + \eta_i^2 \mathbb{E}\left\|\nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\|^2$$
$$- 2\eta_i \mathbb{E}\left\langle \boldsymbol{x}^{(k)} - \mathbb{E}[\boldsymbol{x}^{(k)}], \nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\rangle . \tag{41}$$

Then, we define $\epsilon = \boldsymbol{x}^{(k)} - \overline{\boldsymbol{x}}^{(k)}$. Accordingly, we have

$$\mathbb{E}\left\|\nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\|^2$$

$$=\mathbb{E}_\epsilon \left\|\nabla F_i(\overline{\boldsymbol{x}}^{(k)} + \epsilon) - \mathbb{E}_\zeta[\nabla F_i(\overline{\boldsymbol{x}}^{(k)} + \zeta)]\right\|^2 \tag{42}$$

$$=\mathbb{E}_\epsilon \left\|\mathbb{E}_\zeta[\nabla F_i(\overline{\boldsymbol{x}}^{(k)} + \epsilon) - \nabla F_i(\overline{\boldsymbol{x}}^{(k)} + \zeta)]\right\|^2 \tag{43}$$

$$\leq L_i \mathbb{E}_\epsilon \left\langle \mathbb{E}_\zeta[\epsilon - \zeta], \mathbb{E}_\zeta[\nabla F_i(\overline{\boldsymbol{x}}^{(k)} + \epsilon) - \nabla F_i(\overline{\boldsymbol{x}}^{(k)} + \zeta)]\right\rangle \tag{44}$$

$$=L_i \mathbb{E}_\epsilon \left\langle \epsilon, \mathbb{E}_\zeta[\nabla F_i(\overline{\boldsymbol{x}}^{(k)} + \epsilon) - \nabla F_i(\overline{\boldsymbol{x}}^{(k)} + \zeta)]\right\rangle \tag{45}$$

$$=L_i \mathbb{E}\left\langle \boldsymbol{x}^{(k)} - \mathbb{E}[\boldsymbol{x}^{(k)}], \nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\rangle \tag{46}$$

where (44) is because of the second inequality (12) in Lemma 2. As a consequence, when $\eta_i L_i < 2$, we have

$$\eta_i \mathbb{E}\left\|\nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\|^2 \leq 2\eta_i \mathbb{E}\left\langle \boldsymbol{x}^{(k)} - \mathbb{E}[\boldsymbol{x}^{(k)}], \nabla F_i(\boldsymbol{x}^{(k)}) - \mathbb{E}[\nabla F_i(\boldsymbol{x}^{(k)})]\right\rangle . \tag{47}$$

Substituting (47) back into (41), it follows that

$$\mathbb{E}\left\|\boldsymbol{x}^{(k+1)} - \mathbb{E}[\boldsymbol{x}^{(k+1)}]\right\|^2 \leq (k+1)\eta_i^2 \sigma^2 . \tag{48}$$

Here we complete the induction and prove that Assumption 2 holds for SGD and $q_i(k) = k\eta_i^2 \sigma^2$.

B.4. Empirical Validations for Adaptive Client Optimizers

While for vanilla SGD client optimizer, we can get the analytical expressions of h_i, q_i in Assumptions 1 and 2, it can be complicated to perform the same analysis for adaptive client optimizers. So in this subsection, we are going to provide some empirical evidence that adaptive optimizers (such as ADAM) also satisfies Assumption 1 and can yield smaller h_i values than vanilla SGD.

In particular, we evaluate the performance of vanilla SGD and ADAM on the MNIST dataset (Deng, 2012). For each optimizer, we train two logistic regression models, which start from two different initial points $\boldsymbol{x}, \boldsymbol{y}$ but traverse the same sequence of mini-batches of data. After repeating the same experiment multiple times with different random seeds, we report $h_i = \|\mathbb{E}[\mathcal{A}_i(\boldsymbol{x}; k)] - \mathbb{E}[\mathcal{A}_i(\boldsymbol{y}; k)]\|^2 / \|\boldsymbol{x} - \boldsymbol{y}\|^2$ in Figure 4. One can observe that, given a number of local steps k, ADAM can have a smaller value of h_i than vanilla SGD.
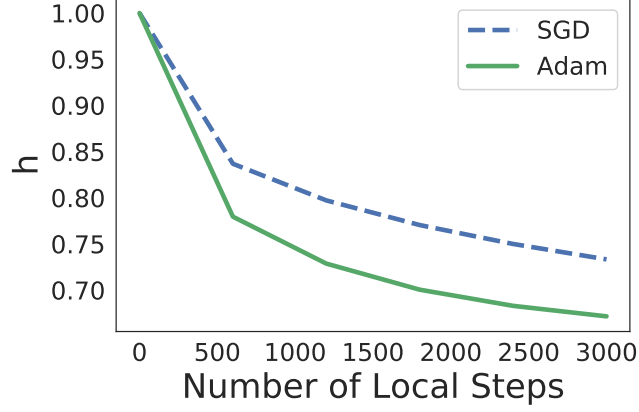
Figure 4. Empirical validation of Assumption 1 for various client optimizers. The value of h is evaluated by training logistic regression models on the MNIST dataset. For each optimizer, we select the best client learning rate from $\{0.5, 0.2, 0.02, 0.002\}$.

C. Proof of Theorem 1

We first prove $\mathbb{E}[\mathcal{A}]$ is a contractive operator. Based on its definition, we have

$$\left\| \mathbb{E}[\mathcal{A}(\boldsymbol{x})] - \mathbb{E}[\mathcal{A}(\boldsymbol{y})] \right\|^2 = \left\| \sum_{i=1}^{M} w_i \mathbb{E}[\mathcal{A}_i(\boldsymbol{x}) - \mathcal{A}_i(\boldsymbol{y})] \right\|^2 \tag{49}$$

$$\leq \sum_{i=1}^{M} w_i \left\| \mathbb{E}[\mathcal{A}_i(\boldsymbol{x}) - \mathcal{A}_i(\boldsymbol{y})] \right\|^2 \tag{50}$$

$$\leq \sum_{i=1}^{M} w_i h_i \left\| \boldsymbol{x} - \boldsymbol{y} \right\|^2 \tag{51}$$

where (50) comes from the Jensen's inequality and (51) is based on Assumption 1. Since $0 \leq \sum_{i=1}^{M} w_i h_i < 1$, according to the Banach fixed-point theorem (Ryu & Boyd, 2016), operator $\mathbb{E}[\mathcal{A}]$ is contractive and has a unique fixed point, denoted by $\widetilde{\boldsymbol{x}}$.

Then, according to the global update rule (5) of FEDOPT, we have

$$\mathbb{E} \left\| \boldsymbol{x}^{(t+1)} - \widetilde{\boldsymbol{x}} \right\|^2 = \left\| (1-\alpha)(\boldsymbol{x}^{(t)} - \widetilde{\boldsymbol{x}}) + \alpha(\mathbb{E}[\mathcal{A}(\boldsymbol{x}^{(t)})] - \widetilde{\boldsymbol{x}}) \right\|^2$$

$$+ \alpha^2 \mathbb{E} \left\| \sum_{i=1}^{M} w_i \mathcal{A}_i(\boldsymbol{x}^{(t)}) - \mathbb{E}[\sum_{i=1}^{M} w_i \mathcal{A}(\boldsymbol{x}^{(t)})] \right\|^2. \tag{52}$$

The first term in (52) can be bounded as follows

$$\left\| (1-\alpha)(\boldsymbol{x}^{(t)} - \widetilde{\boldsymbol{x}}) + \alpha(\mathbb{E}[\mathcal{A}(\boldsymbol{x}^{(t)})] - \widetilde{\boldsymbol{x}}) \right\|^2$$

$$\leq (1-\alpha) \left\| \boldsymbol{x}^{(t)} - \widetilde{\boldsymbol{x}} \right\|^2 + \alpha \left\| \mathbb{E}[\mathcal{A}(\boldsymbol{x}^{(t)})] - \widetilde{\boldsymbol{x}} \right\|^2 \tag{53}$$

$$\leq \left[1 - \alpha(1 - \sum_{i=1}^{M} w_i h_i) \right] \left\| \boldsymbol{x}^{(t)} - \widetilde{\boldsymbol{x}} \right\|^2 \tag{54}$$

where (53) comes from the fact that for any vectors $\boldsymbol{a}, \boldsymbol{b}$, we have $\left\| \alpha \boldsymbol{a} + (1-\alpha) \boldsymbol{b} \right\|^2 \leq \alpha \left\| \boldsymbol{a} \right\|^2 + (1-\alpha) \left\| \boldsymbol{b} \right\|^2$, and the

last inequality is from the contraction property of $\mathbb{E}[\mathcal{A}]$. For the second term in (52), we have

$$\mathbb{E}\left\|\sum_{i=1}^{M} w_i \mathcal{A}_i(\boldsymbol{x}^{(t)}) - \mathbb{E}[\sum_{i=1}^{M} w_i \mathcal{A}(\boldsymbol{x}^{(t)})]\right\|^2 = \sum_{i=1}^{M} w_i^2 \mathbb{E}\left\|\mathcal{A}_i(\boldsymbol{x}^{(t)}) - \mathbb{E}[\mathcal{A}_i(\boldsymbol{x}^{(t)})]\right\|^2 \tag{55}$$

$$\leq \sigma^2 \sum_{i=1}^{M} w_i^2 q_i \tag{56}$$

Substituting (54) and (56) back into (52) we have

$$\mathbb{E}\left\|\boldsymbol{x}^{(t+1)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq \left[1 - \alpha(1 - \sum_{i=1}^{M} w_i h_i)\right]\left\|\boldsymbol{x}^{(t)} - \widetilde{\boldsymbol{x}}\right\|^2 + \alpha^2 \sigma^2 \sum_{i=1}^{M} w_i^2 q_i. \tag{57}$$

Taking the total expectation on both sides, one can get

$$\mathbb{E}\left\|\boldsymbol{x}^{(t+1)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq \left[1 - \alpha(1 - \sum_{i=1}^{M} w_i h_i)\right]\mathbb{E}\left\|\boldsymbol{x}^{(t)} - \widetilde{\boldsymbol{x}}\right\|^2 + \alpha^2 \sigma^2 \sum_{i=1}^{M} w_i^2 q_i. \tag{58}$$

In order to get the final convergence rate, we need a technical lemma from (Stich, 2019), stated as follows.

Lemma 3 (Stich (2019)). *Suppose there are two non-negative sequences $\{r_t\}, \{s_t\}$ that satisfy the relation*

$$r_{t+1} \leq (1 - a\gamma_t)r_t - b\gamma_t s_t + c\gamma_t^2 \tag{59}$$

for all $t \geq 0$ and for parameters $b > 0, a, c \geq 0$ and non-negative stepsizes $\{\gamma_t\}$ with $\gamma_t \leq 1/d$ for a parameter $d \geq a, d > 0$. Then, there exists weights $w_t \geq 0, W_T := \sum_{t=0}^{T} w_t$, such that:

$$\frac{b}{W_T}\sum_{t=0}^{T} s_t w_t + a r_{T+1} \leq 32 d r_0 \left(1 - \frac{a}{d}\right)^{\frac{T}{2}} + \frac{36c}{aT} \leq 32 d r_0 \exp\left[-\frac{aT}{2d}\right] + \frac{36c}{aT} \tag{60}$$

By setting $r_t = \mathbb{E}\left\|\boldsymbol{x}^{(t)} - \widetilde{\boldsymbol{x}}\right\|^2, s_t = 0, a = 1 - \sum_{i=1}^{M} w_i h_i, c = \sigma^2 \sum_{i=1}^{M} w_i^2 q_i, d = 1$, we can obtain from Lemma 3:

$$\mathbb{E}\left\|\boldsymbol{x}^{(T+1)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq \frac{32\left\|\boldsymbol{x}^{(0)} - \widetilde{\boldsymbol{x}}\right\|^2}{1 - \sum_{i=1}^{M} w_i h_i}\left[\sum_{i=1}^{M} w_i h_i\right]^{\frac{T}{2}} + \frac{36\sigma^2 \sum_{i=1}^{M} w_i^2 q_i}{T(1 - \sum_{i=1}^{M} w_i h_i)^2} \tag{61}$$

$$\leq c \cdot \left(\left[\sum_{i=1}^{M} w_i h_i\right]^{\frac{T}{2}} \frac{\left\|\boldsymbol{x}^{(0)} - \widetilde{\boldsymbol{x}}\right\|^2}{1 - \sum_{i=1}^{M} w_i h_i} + \frac{1}{T}\frac{\sigma^2 \sum_{i=1}^{M} w_i^2 q_i}{(1 - \sum_{i=1}^{M} w_i h_i)^2}\right) \tag{62}$$

where c is a positive constant. Here we complete the proof of Theorem 1.

Special Case: Client Optimizer is GD. When the client optimizer is GD (*i.e.,* $\sigma = 0$), all clients have the same η, τ, μ, L, and $w_i = 1/M$, we have $h_i = (1 - \eta\mu)^{2\tau}$. Then, we can directly set $\alpha = 1$ in (58), we get

$$\mathbb{E}\left\|\boldsymbol{x}^{(T)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq (1 - \eta\mu)^{2\tau T}\left\|\boldsymbol{x}^{(0)} - \widetilde{\boldsymbol{x}}\right\|^2. \tag{63}$$

If all local operator $\mathcal{A}_i$ have the same fixed point $\boldsymbol{x}_i^* = \boldsymbol{x}^*$, then we have $\widetilde{\boldsymbol{x}} = \boldsymbol{x}^*$ and (63) recovers the convergence rate of local GD in the IID data setting.

Special Case: Client Optimizer is SGD. When the client optimizer is SGD, all clients have the same η, τ, μ, L, and $w_i = 1/M$, we have $h_i = (1 - \eta\mu)^{2\tau}$ and $q_i = \tau\eta^2\sigma^2$. Substituting these into (58), we get

$$\mathbb{E}\left\|\boldsymbol{x}^{(T)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq [1 - \alpha(1 - (1 - \eta\mu)^{2\tau})] \cdot \mathbb{E}\left\|\boldsymbol{x}^{(t)} - \widetilde{\boldsymbol{x}}\right\|^2 + \frac{\alpha^2\eta^2\sigma^2\tau}{M}. \tag{64}$$

Now we are going to prove that for any $T \geq 0$, with $\alpha = 2/[(1 - (1 - \eta\mu)^{2\tau})(t + \beta)]$,

$$\mathbb{E}\left\|\boldsymbol{x}^{(T)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq \frac{4\sigma^2\eta^2\tau}{M(1 - (1 - \eta\mu)^{2\tau})^2(T + \beta)} \tag{65}$$

where β is a constant, that satisfies that $0 < \beta \leq 4\sigma^2\eta^2\tau/[(1 - (1 - \eta\mu)^{2\tau})\left\|\boldsymbol{x}^{(0)} - \widetilde{\boldsymbol{x}}\right\|]^2$. When $t = 0$, the inequality (65) automatically holds according to the definition of β. Then, we assume (65) for some $t > 1$ and examine the situation $t + 1$.

$$\mathbb{E}\left\|\boldsymbol{x}^{(t+1)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq \left[1 - \frac{2}{t + \beta}\right]\frac{4\sigma^2\eta^2\tau}{M(1 - (1 - \eta\mu)^{2\tau})^2(t + \beta)} + \frac{4\sigma^2\eta^2\tau}{M(1 - (1 - \eta\mu)^{2\tau})^2(t + \beta)^2} \tag{66}$$

$$= \frac{t + \beta - 1}{t + \beta}\frac{4\sigma^2\eta^2\tau}{M(1 - (1 - \eta\mu)^{2\tau})^2(t + \beta)} \tag{67}$$

$$\leq \frac{4\sigma^2\eta^2\tau}{M(1 - (1 - \eta\mu)^{2\tau})^2(t + 1 + \beta)}. \tag{68}$$

So (65) also holds for $t + 1$. We complete the induction procedure and conclude that

$$\mathbb{E}\left\|\boldsymbol{x}^{(T)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq \frac{4\sigma^2\eta^2\tau}{M(1 - (1 - \eta\mu)^{2\tau})^2(T + \beta)} \tag{69}$$

$$= \frac{\sigma^2}{\mu^2 M\tau(T + \beta)}\left(\frac{2\eta\mu\tau}{1 - (1 - \eta\mu)^{2\tau}}\right)^2. \tag{70}$$

When $\tau = 1$, we have $\widetilde{\boldsymbol{x}} = \boldsymbol{x}^*$ and

$$\mathbb{E}\left\|\boldsymbol{x}^{(T)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq \frac{\sigma^2}{\mu^2 M(T + \beta)}\left(\frac{2\eta\mu}{1 - (1 - \eta\mu)^2}\right)^2 \tag{71}$$

$$= \frac{\sigma^2}{\mu^2 M(T + \beta)}\left(\frac{2\eta\mu}{\eta\mu(2 - \eta\mu)}\right)^2 \tag{72}$$

$$= \frac{\sigma^2}{\mu^2 M(T + \beta)}\left(\frac{2}{2 - \eta\mu}\right)^2 \tag{73}$$

$$\leq \frac{4\sigma^2}{\mu^2 M(T + \beta)} \tag{74}$$

where the last inequality follows from $\eta\mu \leq 1$. The result (74) recovers the optimal rate for distributed synchronous SGD (Bottou et al., 2018). When $\tau > 1$, one can obtain that

$$\mathbb{E}\left\|\boldsymbol{x}^{(T)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq \frac{\sigma^2}{\mu^2 M\tau(T + \beta)}\left[z(\eta\mu)\right]^2 \tag{75}$$

where $z(x) = 2x\tau/(1 - (1 - x)^{2\tau})$ for $x > 0$. When $\eta\mu \to 0$, we have $z(\eta\mu) \simeq 1$. In other cases, we are going to prove that if $\eta\mu\tau$ is upper bounded, then $z(\eta\mu)$ can also be upper bounded by some constant. In particular, we first need to prove $z(x)$ is monotonically increasing with x by checking the derivative of $z(x)$:

$$z'(x) = \frac{2\tau[1 - (1 - x)^{2\tau}] - 4x\tau^2(1 - x)^{2\tau - 1}}{[1 - (1 - x)^{2\tau}]^2} \tag{76}$$

$$= \frac{2\tau}{[1 - (1 - x)^{2\tau}]^2}\left[1 - (1 - x)^{2\tau} - 2x\tau(1 - x)^{2\tau - 1}\right] \tag{77}$$

$$= \frac{2\tau}{[1 - (1 - x)^{2\tau}]^2}\left[1 - [1 + (2\tau - 1)x](1 - x)^{2\tau - 1}\right] \tag{78}$$

$$\geq \frac{2\tau}{[1 - (1 - x)^{2\tau}]^2}\left[1 - (1 + x)^{(2\tau - 1)}(1 - x)^{2\tau - 1}\right] \tag{79}$$

$$= \frac{2\tau}{[1 - (1 - x)^{2\tau}]^2}\left[1 - (1 - x^2)^{2\tau - 1}\right] > 0. \tag{80}$$

Suppose $\eta\mu\tau \leq 1$, it follows that

$$\max z(\eta\mu) = z(1/\tau) = \frac{2}{1 - (1 - \frac{1}{\tau})^{2\tau}} \leq \frac{2}{1 - e^{-2}} < 3. \tag{81}$$

Substituting (81) into (75), we have

$$\mathbb{E}\left\|\boldsymbol{x}^{(T)} - \widetilde{\boldsymbol{x}}\right\|^2 \leq \frac{9\sigma^2}{\mu^2 M\tau(T+\beta)} \tag{82}$$

which matches the lower bound of local SGD in the IID data setting (Woodworth et al., 2020), in which all local operator share the same fixed point $\widetilde{\boldsymbol{x}} = \boldsymbol{x}^* = \boldsymbol{x}_i^*$ for all i.

D. Connection with Previous Works on the Minimizer Inconsistency

When client learning rates, number of local steps are the same across all clients, and non-adaptive, deterministic CLIENTOPT are used, $\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}^*\|$ can vanish to zero along with the learning rates. This phenomenon has been observed and analyzed by few recent literature in different forms, see (Charles & Konečnỳ, 2021; Malinovskiy et al., 2020; Pathak & Wainwright, 2020). Theorem 1 generalizes these results by allowing heterogeneous local hyper-parameters and adaptive, stochastic client optimizers. In addition, the non-vanishing bias was studied in (Wang et al., 2020a) by assuming different local learning rates and local steps at clients. In this paper, we further generalize the results by showing that even when the learning rates and local steps are the same, using local adaptive methods will lead to a non-vanishing gap. We summarize the differences in Table 6.

Papers	Different η, τ	Stochastic CLIENTOPT	Adaptive CLIENTOPT	Non-vanishing bias
(Charles & Konečnỳ, 2021)	✗	✗	✗	✗
(Malinovsky et al., 2020)	✗	✗	✗	✗
(Pathak & Wainwright, 2020)	✗	✗	✗	✗
(Wang et al., 2020a)	✓	✓	✗	✓
This paper	✓	✓	✓	✓

Table 6. Comparison with previous works that studied minimizer inconsistency in different forms.

E. Proof of Theorem 2

In the quadratic problem, we can write down the analytical expression of operator $\mathcal{A}_i$. Specifically, for the K-th local iterate of client i, we have

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \eta_i \boldsymbol{P}_i \nabla F_i(\boldsymbol{x}^{(k)}) \tag{83}$$

$$= \boldsymbol{x}^{(k)} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i(\boldsymbol{x}^{(k)} - \boldsymbol{x}_i^*) \tag{84}$$

$$= (\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)(\boldsymbol{x}^{(k)} - \boldsymbol{x}_i^*) + \boldsymbol{x}_i^*. \tag{85}$$

That is,

$$\boldsymbol{x}^{(k+1)} - \boldsymbol{x}_i^* = (\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)^{k+1}(\boldsymbol{x} - \boldsymbol{x}_i^*). \tag{86}$$

According to the definition of $\mathcal{A}_i$, we have

$$\mathcal{A}_i(\boldsymbol{x}; \tau_i) = (\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)^{\tau_i}(\boldsymbol{x} - \boldsymbol{x}_i^*) + \boldsymbol{x}_i^*, \tag{87}$$

$$\mathcal{A}(\boldsymbol{x}) = \sum_{i=1}^{M} w_i[(\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)^{\tau_i}(\boldsymbol{x} - \boldsymbol{x}_i^*) + \boldsymbol{x}_i^*]. \tag{88}$$

We first show that $\mathcal{A}$ is contractive. Note that

$$\mathcal{A}(\boldsymbol{x}) - \mathcal{A}(\boldsymbol{y}) = \left[\sum_{i=1}^{M} w_i(\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)^{\tau_i}\right](\boldsymbol{x} - \boldsymbol{y}). \tag{89}$$

Therefore, as long as the operator norm of $\sum_{i=1}^{M} w_i (\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)^{\tau_i}$ is smaller than 1, the operator $\mathcal{A}$ is contractive and has a unique fixed point $\widetilde{\boldsymbol{x}}$. Next, we are going to find the analytical expression of $\widetilde{\boldsymbol{x}}$. We have

$$\widetilde{\boldsymbol{x}} - \mathcal{A}(\widetilde{\boldsymbol{x}}) = \sum_{i=1}^{M} w_i [\boldsymbol{I} - (\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)^{\tau_i}](\widetilde{\boldsymbol{x}} - \boldsymbol{x}_i^*) = 0. \tag{90}$$

After minor rearranging, it follows that

$$\widetilde{\boldsymbol{x}} = \left[\sum_{i=1}^{M} w_i [\boldsymbol{I} - (\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)^{\tau_i}] \right]^{-1} \left[\sum_{i=1}^{M} w_i [\boldsymbol{I} - (\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)^{\tau_i}] \boldsymbol{x}_i^* \right]. \tag{91}$$

When $\eta_i = \gamma_i \eta$ and η approaches to zero, we have $\boldsymbol{I} - (\boldsymbol{I} - \eta_i \boldsymbol{P}_i \boldsymbol{H}_i)^{\tau_i} \simeq \eta_i \tau_i \boldsymbol{P}_i \boldsymbol{H}_i$ and

$$\lim_{\eta \to 0} \widetilde{\boldsymbol{x}} = \left[\sum_{i=1}^{M} w_i \gamma_i \tau_i \boldsymbol{P}_i \boldsymbol{H}_i \right]^{-1} \left[\sum_{i=1}^{M} w_i \gamma_i \tau_i \boldsymbol{P}_i \boldsymbol{H}_i \boldsymbol{x}_i^* \right]. \tag{92}$$

Here we complete the proof.

F. Proof for the Convergence of Local Correction

F.1. Main Results

Without loss of generalities, suppose that at the t-th round, the local model changes of client i can be written as

$$\boldsymbol{x}^{(t)} - \mathcal{A}_i(\boldsymbol{x}^{(t)}; \tau_i) = \eta_i \sum_{k=0}^{\tau_i - 1} \boldsymbol{B}_i^{(t,k)} \nabla F_i(\boldsymbol{x}_i^{(t,k)}) \tag{93}$$

where η_i is the client learning rate, $\{\boldsymbol{B}_i^{(t,k)}\}$ are symmetric and positive definite matrices, and $\boldsymbol{x}_i^{(t,k)}$ denotes the local iterate after performing k local steps. When the local correction technique is applied, the client will send the following normalized local changes to the server:

$$\boldsymbol{h}_i^{(t)} = \frac{1}{\sum_{k=0}^{\tau_i - 1} \boldsymbol{B}_i^{(t,k)}} \sum_{k=0}^{\tau_i - 1} \boldsymbol{B}_i^{(t,k)} \nabla F_i(\boldsymbol{x}_i^{(t,k)}) := \sum_{k=0}^{\tau_i - 1} \boldsymbol{A}_i^{(t,k)} \nabla F_i(\boldsymbol{x}_i^{(t,k)}) \tag{94}$$

where $\boldsymbol{A}_i^{(t,k)} = \boldsymbol{B}_i^{(t,k)} / \sum_{k=0}^{\tau_i - 1} \boldsymbol{B}_i^{(t,k)}$ and $\sum_{k=0}^{\tau_i - 1} \boldsymbol{A}_i^{(t,k)} = \boldsymbol{I}$. Then, the server will aggregate the normalized local changes and update the global model as follows

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \alpha \sum_{i=1}^{M} w_i \boldsymbol{h}_i^{(t)} \tag{95}$$

where α denotes the server learning rate.

Our convergence analysis will be centered around the following assumptions.

Assumption 3. *Each local objective is Lipschitz smooth, that is, $\|\nabla F_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\|, \forall i \in [M]$.*

Assumption 4. *The matrices $\{\boldsymbol{A}_i^{(t,k)}\}$ are positive-definite symmetric matrices and have bounded operator norm: $\left\| \boldsymbol{A}_i^{(t,k)} \right\|_{op} \leq \Lambda / tau_i$.*

Assumption 5. *The pre-conditioned gradients at each local iteration have bounded norm, i.e., $\left\| \boldsymbol{B}_i^{(t,k)} \nabla F_i(\boldsymbol{x}_i^{(t,k)}) \right\| \leq G$.*

Theorem 3 (Convergence Guarantee for Local Correction Technique). *Suppose all clients have the same client learning rate η and the same number of local sptes τ. Under Assumptions 3 to 5, if the server learning rate is set as $\alpha = \eta\tau \leq 1/L$ and the client learning rate is*

$$\eta = \min \left\{ \frac{1}{\tau L}, \frac{1}{\tau T^{\frac{1}{3}}} \left(\frac{D}{L^2 \Lambda G^2} \right)^{\frac{1}{3}} \right\}. \tag{96}$$

where $D = F(\boldsymbol{x}^{(0)}) - F_{inf}$, then the global iterate (95) converges at the following rate:

$$\min_{t \in [0,T]} \left\| \nabla F(\boldsymbol{x}^{(t)}) \right\|^2 = \mathcal{O}\left(\frac{1}{T} + \frac{\Lambda^{\frac{1}{3}} G^{\frac{2}{3}}}{T^{\frac{2}{3}}} \right). \tag{97}$$

It is worth noting that the convergence rate (97) matches previous results in (Koloskova et al., 2020; Khaled et al., 2020) in the deterministic, non-IID data setting. Theorem 3 shows that using adaptive client optimizers together with local correction can preserve the same convergence rate as vanilla SGD client optimizer and there is no non-vanishing solution bias.

F.2. Technical Lemmas

Lemma 4. *Suppose* $\boldsymbol{A}_k \in \mathbb{R}^{d \times d}, k \in [1, K]$ *are symmetric positive definite matrices.*

$$\left\| \sum_{k=1}^{K} \boldsymbol{A}_k \boldsymbol{b}_k \right\|^2 \le \| \boldsymbol{A}_s \|_{op}^2 \left\| \boldsymbol{A}_s^{-1} \right\|_{op} \sum_{k=1}^{K} \| \boldsymbol{A}_k \|_{op} \| \boldsymbol{b}_k \|^2 \tag{98}$$

where $\boldsymbol{A}_s = \sum_{k=1}^{K} \boldsymbol{A}_k$.

Proof. We define $\widetilde{\boldsymbol{A}}_k = (\sum_{k=1}^{K} \boldsymbol{A}_k)^{-1} \boldsymbol{A}_k = \boldsymbol{A}_s^{-1} \boldsymbol{A}_k$. It directly follws that $\sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k = \boldsymbol{I}$. For the left hand side of (98), we have

$$\left\| \sum_{k=1}^{K} \boldsymbol{A}_k \boldsymbol{b}_k \right\|^2 = \left\| \boldsymbol{A}_s \sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k \boldsymbol{b}_k \right\|^2 = \left\| \boldsymbol{A}_s \bar{\boldsymbol{b}} \right\|^2 \le \| \boldsymbol{A}_s \|_{op}^2 \| \bar{\boldsymbol{b}} \|^2 \tag{99}$$

where $\bar{\boldsymbol{b}} = \sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k \boldsymbol{b}_k$. On the other hand, let $\boldsymbol{v}_k = \boldsymbol{b}_k - \bar{\boldsymbol{b}}$ and note that

$$\mathrm{Tr}\left(\sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k \boldsymbol{b}_k \boldsymbol{b}_k^\top \right) = \mathrm{Tr}\left(\sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k (\boldsymbol{b}_k - \bar{\boldsymbol{b}} + \bar{\boldsymbol{b}})(\boldsymbol{b}_k - \bar{\boldsymbol{b}} + \bar{\boldsymbol{b}})^\top \right) \tag{100}$$

$$= \mathrm{Tr}\left(\sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k \left(\boldsymbol{v}_k \boldsymbol{v}_k^\top + \boldsymbol{v}_k \bar{\boldsymbol{b}}^\top + \bar{\boldsymbol{b}} \boldsymbol{v}_k^\top + \bar{\boldsymbol{b}}\, \bar{\boldsymbol{b}}^\top \right) \right) \tag{101}$$

$$= \underbrace{\mathrm{Tr}\left(\sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k \boldsymbol{v}_k \boldsymbol{v}_k^\top \right)}_{\ge 0} + \underbrace{\mathrm{Tr}\left(\sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k \boldsymbol{v}_k \bar{\boldsymbol{b}}^\top \right) + \mathrm{Tr}\left(\sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k \bar{\boldsymbol{b}} \boldsymbol{v}_k^\top \right)}_{=0} + \mathrm{Tr}\left(\bar{\boldsymbol{b}}\, \bar{\boldsymbol{b}}^\top \right). \tag{102}$$

For the third term, we have

$$\mathrm{Tr}\left(\sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k \bar{\boldsymbol{b}} \boldsymbol{v}_k^\top \right) = \mathrm{Tr}\left(\sum_{k=1}^{K} \boldsymbol{v}_k^\top \widetilde{\boldsymbol{A}}_k \bar{\boldsymbol{b}} \right) = \mathrm{Tr}\left(\sum_{k=1}^{K} \boldsymbol{v}_k^\top \widetilde{\boldsymbol{A}}_k^\top \bar{\boldsymbol{b}} \right) = 0. \tag{103}$$

Therefore, we can obtain that

$$\| \bar{\boldsymbol{b}} \|^2 = \mathrm{Tr}\left(\bar{\boldsymbol{b}}\, \bar{\boldsymbol{b}}^\top \right) \le \mathrm{Tr}\left(\sum_{k=1}^{K} \widetilde{\boldsymbol{A}}_k \boldsymbol{b}_k \boldsymbol{b}_k^\top \right) \tag{104}$$

$$= \sum_{k=1}^{K} \boldsymbol{b}_k^\top \widetilde{\boldsymbol{A}}_k \boldsymbol{b}_k \tag{105}$$

$$\le \sum_{k=1}^{K} \left\| \widetilde{\boldsymbol{A}}_k \right\|_{op} \| \boldsymbol{b}_k \|^2 \tag{106}$$

$$\le \left\| \boldsymbol{A}_s^{-1} \right\|_{op} \sum_{k=1}^{K} \| \boldsymbol{A}_k \|_{op} \| \boldsymbol{b}_k \|^2. \tag{107}$$

Substituting (107) into (99), it follows that

$$\left\| \sum_{k=1}^{K} \boldsymbol{A}_k \boldsymbol{b}_k \right\|^2 \leq \|\boldsymbol{A}_s\|_{\text{op}}^2 \left\|\boldsymbol{A}_s^{-1}\right\|_{\text{op}} \sum_{k=1}^{K} \|\boldsymbol{A}_k\|_{\text{op}} \|\boldsymbol{b}_k\|^2 . \tag{108}$$

$\square$

F.3. Proof of Theorem 3

Since each local objective is L-smooth, we have

$$F(\boldsymbol{x}^{(t+1)}) - F(\boldsymbol{x}^{(t)})$$

$$\leq -\alpha \left\langle \nabla F(\boldsymbol{x}^{(t)}), \sum_{i=1}^{M} w_i \boldsymbol{h}_i^{(t)} \right\rangle + \frac{\alpha^2 L}{2} \left\| \sum_{i=1}^{M} w_i \boldsymbol{h}_i^{(t)} \right\|^2 \tag{109}$$

$$= -\frac{\alpha}{2} \left[\left\| \nabla F(\boldsymbol{x}^{(t)}) \right\|^2 + \left\| \sum_{i=1}^{M} w_i \boldsymbol{h}_i^{(t)} \right\|^2 - \left\| \nabla F(\boldsymbol{x}^{(t)}) - \sum_{i=1}^{M} w_i \boldsymbol{h}_i^{(t)} \right\|^2 \right] + \frac{\alpha^2 L}{2} \left\| \sum_{i=1}^{M} w_i \boldsymbol{h}_i^{(t)} \right\|^2 \tag{110}$$

$$\leq -\frac{\alpha}{2} \left\| \nabla F(\boldsymbol{x}^{(t)}) \right\|^2 + \frac{\alpha}{2} \left\| \nabla F(\boldsymbol{x}^{(t)}) - \sum_{i=1}^{M} w_i \boldsymbol{h}_i^{(t)} \right\|^2 \tag{111}$$

$$\leq -\frac{\alpha}{2} \left\| \nabla F(\boldsymbol{x}^{(t)}) \right\|^2 + \frac{\alpha}{2} \sum_{i=1}^{M} w_i \left\| \nabla F_i(\boldsymbol{x}^{(t)}) - \boldsymbol{h}_i^{(t)} \right\|^2 \tag{112}$$

where (110) uses the fact: $\langle a, b \rangle = \frac{1}{2}[\|a\|^2 + \|b\|^2 - \|a - b\|^2]$, (111) follows from the assumption $\alpha L < 1$, and (112) is obtained by applying Jensen's Inequality. For the second term in (112), we can further bound it as follows:

$$\left\| \nabla F_i(\boldsymbol{x}^{(t)}) - \boldsymbol{h}_i^{(t)} \right\|^2 = \left\| \sum_{k=0}^{\tau_i - 1} \boldsymbol{A}_i^{(t,k)} \left[\nabla F_i(\boldsymbol{x}^{(t)}) - \nabla F_i(\boldsymbol{x}_i^{(t,k)}) \right] \right\|^2 \tag{113}$$

$$\leq \sum_{k=0}^{\tau_i - 1} \left\| \boldsymbol{A}_i^{(t,k)} \right\|_{\text{op}} \left\| \nabla F_i(\boldsymbol{x}^{(t)}) - \nabla F_i(\boldsymbol{x}_i^{(t,k)}) \right\|^2 \tag{114}$$

$$\leq L^2 \sum_{k=0}^{\tau_i - 1} \left\| \boldsymbol{A}_i^{(t,k)} \right\|_{\text{op}} \left\| \boldsymbol{x}^{(t)} - \boldsymbol{x}_i^{(t,k)} \right\|^2 \tag{115}$$

$$\leq \frac{L^2 \Lambda}{\tau_i} \sum_{k=0}^{\tau_i - 1} \left\| \boldsymbol{x}^{(t)} - \boldsymbol{x}_i^{(t,k)} \right\|^2 \tag{116}$$

$$= \frac{L^2 \Lambda}{\tau_i} \sum_{k=0}^{\tau_i - 1} \left\| \Delta_i^{(t,k)} \right\|^2 \tag{117}$$

where (114) follows Lemma 4, (115) is based on the Lipschitz smoothness of the local objectives, and (116) uses the assumption that matrices $\boldsymbol{A}_i^{(t,k)}$ have bounded operator norm. Substituting (117) into (112), we have

$$F(\boldsymbol{x}^{(t+1)}) - F(\boldsymbol{x}^{(t)}) \leq -\frac{\alpha}{2} \left\| \nabla F(\boldsymbol{x}^{(t)}) \right\|^2 + \frac{\alpha L^2 \Lambda}{2} \sum_{i=1}^{M} \frac{w_i}{\tau_i} \sum_{k=0}^{\tau_i - 1} \left\| \Delta_i^{(t,k)} \right\|^2 . \tag{118}$$

Taking the sum from $t = 0$ to $t = T - 1$ and rearranging, we obtain

$$\frac{1}{T} \sum_{t=0}^{T-1} \left\| \nabla F(\boldsymbol{x}^{(t)}) \right\|^2 \leq \frac{2(F(\boldsymbol{x}^{(0)}) - F(\boldsymbol{x}^{(T)}))}{\alpha T} + \frac{L^2 \Lambda}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{M} \frac{w_i}{\tau_i} \sum_{k=0}^{\tau_i - 1} \left\| \Delta_i^{(t,k)} \right\|^2 \tag{119}$$

$$\leq \frac{2(F(\boldsymbol{x}^{(0)}) - F_{\text{inf}})}{\alpha T} + \frac{L^2 \Lambda}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{M} \frac{w_i}{\tau_i} \sum_{k=0}^{\tau_i - 1} \left\| \Delta_i^{(t,k)} \right\|^2 . \tag{120}$$

On the other hand, note that

$$\left\|\Delta_i^{(t,k)}\right\|^2 = \eta_i^2 \left\|\sum_{s=0}^{k-1} \boldsymbol{B}_i^{(t,s)} \nabla F_i(\boldsymbol{x}_i^{(t,s)})\right\|^2 \le k^2 \eta_i^2 G^2 \le \tau_i^2 \eta_i^2 G^2. \tag{121}$$

Substituting (121) into (120), we have

$$\frac{1}{T} \sum_{t=0}^{T-1} \left\|\nabla F(\boldsymbol{x}^{(t)})\right\|^2 \le \frac{2(F(\boldsymbol{x}^{(0)}) - F_{\text{inf}})}{\alpha T} + \frac{L^2 \Lambda}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{M} w_i \eta_i^2 \tau_i^2 G^2 \tag{122}$$

$$= \frac{2(F(\boldsymbol{x}^{(0)}) - F_{\text{inf}})}{\alpha T} + L^2 \Lambda G^2 \sum_{i=1}^{M} w_i \eta_i^2 \tau_i^2 \tag{123}$$

If we let $\alpha = \sum_{i=1}^{M} w_i \eta_i \tau_i \le 1/L$, then it follows that

$$\frac{1}{T} \sum_{t=0}^{T-1} \left\|\nabla F(\boldsymbol{x}^{(t)})\right\|^2 \le \frac{2(F(\boldsymbol{x}^{(0)}) - F_{\text{inf}})}{\sum_{i=1}^{M} w_i \eta_i \tau_i T} + L^2 \Lambda G^2 \sum_{i=1}^{M} w_i \eta_i^2 \tau_i^2. \tag{124}$$

When $\eta_i = \eta, \tau_i = \tau, w_i = 1/M$, we have

$$\frac{1}{T} \sum_{t=0}^{T-1} \left\|\nabla F(\boldsymbol{x}^{(t)})\right\|^2 \le \frac{2(F(\boldsymbol{x}^{(0)}) - F_{\text{inf}})}{\eta \tau T} + L^2 \Lambda G^2 \eta^2 \tau^2 \tag{125}$$

$$\le \frac{DL}{T} + \frac{(D^2 L^2 \Lambda G^2)^{\frac{1}{3}}}{T^{\frac{2}{3}}} \tag{126}$$

where $D := F(\boldsymbol{x}^{(0)}) - F_{\text{inf}}$ and the client learning rate is set as

$$\eta = \min\left\{\frac{1}{\tau L}, \frac{1}{\tau T^{\frac{1}{3}}}\left(\frac{D}{L^2 \Lambda G^2}\right)^{\frac{1}{3}}\right\}. \tag{127}$$