Hybrid Decentralized Optimization: First- and Zeroth-Order Optimizers Can Be Jointly Leveraged for Faster Convergence

Abstract

Distributed optimization has become one of the standard ways of speeding up machine learning training, and most of the research in the area focuses on distributed first-order, gradient-based methods. Yet, there are settings where some computationally-bounded nodes may not be able to implement first-order, gradient-based optimization, while they could still contribute to joint optimization tasks. In this paper, we initiate the study of hybrid decentralized optimization, studying settings where nodes with zeroth-order and first-order optimization capabilities co-exist in a distributed system, and attempt to jointly solve an optimization task over some data distribution. We essentially show that, under reasonable parameter settings, such a system can not only withstand noisier zeroth-order agents, but can even benefit from integrating such agents into the optimization process, rather than ignoring their information. At the core of our approach is a new analysis of distributed optimization with noisy and possibly-biased gradient estimators, which may be of independent interest. Experimental results on standard optimization tasks confirm our analysis, showing that hybrid first-zeroth order optimization can be practical.

1 Introduction

One key enabler of the extremely rapid recent progress of machine learning has been distributed optimization: the ability to efficiently optimize over large quantities of data, and large parameter counts, among multiple nodes or devices, in order to share the computational load, and therefore reduce end-to-end training time. Distributed machine learning has become commonplace, and it is not unusual to encounter systems which distribute model training among tens or even hundreds of nodes.

By and large, the standard distribution strategy in the context of machine learning tasks has been data-parallel [Bottou 2010], using first-order gradient estimators. We can formalize this as follows: considering a classical empirical risk minimization setting, we have a set of samples $S$ from a distribution, and wish to minimize the function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, which is the average of losses over samples from $S$. In other words, we wish to find $x^* = \arg\min_x \sum_{s \in S} f_s(x) / |S|$. Assuming that we have $n$ compute nodes which can process samples in parallel, data-parallel SGD consists of iterations in which each node computes gradient estimator for a batch of samples, and then nodes then exchange this information, either globally, via all-to-all communication, or pair-wise. Specifically, in this paper we will focus on the highly-popular decentralized optimization case, in which nodes interact in randomly chosen pairs, exchanging model information, following each local optimization step.

There is already a vast amount of literature on decentralized optimization in the case where nodes have access to first-order, gradient-based estimators. While this setting is prevalent, it does not cover the interesting case where, among the set of nodes, a fraction only have access to weaker, zeroth-order gradient estimators, corresponding to less computationally-capable devices, but which may still possess useful local data and computation.
In this paper, we initiate the study of hybrid decentralized optimization in the latter setting. Specifically, we aim to answer the following key question:

**Can zeroth-order estimators be integrated in a decentralized setting, and can they boost convergence?**

Roughly, we show that the answer to this question is affirmative. To arrive at it, we must overcome a number of non-trivial technical obstacles, and the answer must be qualified by key parameters, such as the first-order/zeroth-order split in the population, and the estimator variance and bias. More precisely, a key difficulty we must overcome in the algorithm and in the analysis is that, under standard implementations, zeroth-order estimators are biased, breaking one of the key analytic assumptions in existing work on decentralized optimization, e.g. Lian et al. [2017a], Wang and Joshi [2021], Koloskova et al. [2020a,b], Nadiradze et al. [2021].

Our analysis approach overcomes this obstacle, and provides the first convergence bounds for hybrid decentralized optimization via a novel potential argument. Roughly, assuming a $d$-dimensional, $L$-smooth and $\ell$-strongly-convex finite-sum objective function $f$, and a population of $n$ nodes, in which $n_1$ have first-order stochastic gradient estimators of variance $\sigma_1$, and $n_0$ have zeroth-order estimators of variance $\sigma_0$, then our analysis shows that the “stochastic noise” in the convergence of our hybrid decentralized optimization algorithm in this population is given, up to constants, by the following three quantities:

$$\eta\left(\frac{d\sigma_0^2}{n^2} + n_1\xi_1^2\right), \quad \eta\left(\frac{d\sigma_1^2}{n^2} + n_1\sigma_1^2\right), \quad \text{and} \quad \frac{\eta^2 Ldn_0}{n}.$$

In this expression, $\eta$ is the learning rate, and the quantities $\xi_1$ and $\xi_0$ are bounds on the average variance of first-order and zeroth-order estimators at the nodes, respectively, given by the way in which the data is split among these two types of agents. Intuitively, the first term is the variance due to the (random) data split, whereas the second term is the added variance due to noise in the two types of gradient estimators. (The zeroth-order terms are scaled by the dimension, as is common in this case.) The third term bounds the bias induced by the zeroth-order gradient estimators. Using this characterization, we show that there exist reasonable parameter settings such that, if zeroth-order nodes do not have extremely high variance, they may in fact be useful for convergence, especially since the third bias term can be controlled via the learning rate $\eta$.

Our analysis approach should be of independent interest: first, we provide a simple and general way of characterizing convergence in a population mixing first- and zeroth-order agents, which can be easily parametrized given population and estimator properties. (For instance, we can directly cover the case when the zeroth-order estimators are unbiased Chen [2020]; as in this case the bias term becomes zero.) Second, we do so in a very general communication model which allows agents to interact at different rates (due to randomness), covering both the pair-wise interaction model Angluin et al. [2006], Nadiradze et al. [2021], and the global matching interactions model Lian et al. [2017a], Wang and Joshi [2021], Koloskova et al. [2020a,b].

A key remaining question is whether the above characterization can be validated for practical setting. For this, we implemented our algorithm, and examined the convergence under various optimization tasks, population relative sizes, and estimator implementations. Specifically, we implemented three different types of zeroth-order estimators: a standard biased one, e.g. Nesterov and Spokoiny [2017], a de-biased estimator Chen [2020], and the novel gradient-free estimator Baydin et al. [2022], and examined their behavior when mixed with first-order estimators. In brief, our results show that, even for fairly high-dimensional and complex tasks, such as fine-tuning the classification layer of a deep neural network, our approach continues to converge. Importantly, we observe that our approach allows a system to incorporate information from the zeroth-order agents in an efficient and robust, showing higher convergence speed relative to the case where only first-order information is considered for optimization.

**Related Work.** The study of decentralized optimization algorithms dates back to Tsitsiklis [1984], and is related to the study of gossip algorithms for information dissemination Kempe et al. [2003], Xiao and Boyd [2004]. The distinguishing feature of this setting is that optimization occurs jointly, but in the absence of a coordinator node. Several classic first-order algorithms have been ported and analyzed in the gossip setting, such as subgradient methods for convex objectives Nedic and Ozdaglar [2009], Johansson et al. [2009], Shamir and Srebro [2014] or ADMM Wei and Ozdaglar [2012], Lutzeyer et al. [2013]. References Lian et al. [2017a,b], Assran et al. [2018], consider SGD-type algorithms in the non-convex setting, while references Tang et al. [2018], Koloskova et al. [2020a], Nadiradze et al. [2021] analyzed the use of quantization in the gossip setting. By contrast, zeroth-order optimization has been relatively less investigated: Sahu and Kar [2020] propose a distributed deterministic zeroth-order Frank-Wolfe-type algorithm, whereas recent work by Yuan et al. [2021] investigated the rates which can be achieved by decentralized zeroth-order algorithms, proposing a multi-stage method which can match the rate of centralized algorithms in some parameter.
We assume each node. Relative to the latter reference, we focus on simpler decentralized algorithms, which can be easily interface with first-order optimizers, and perform a significantly more in-depth experimental validation.

Stochastic zeroth-order optimization has been classically applied for gradient-free optimization of convex functions, e.g. [Nesterov and Spokoiny (2017)], and has been extended to tackling high-dimensionality and saddle-point constraints, e.g. [Balasubramanian and Ghadimi (2022)]. (The area has tight connections to bandit online optimization, under time-varying objective functions, e.g. [Flaxman et al. (2004), Agarwal et al. (2010), Shamir (2017)]; however, our results are not immediately relevant to this direction, as we are interested in interactions with agents possessing first-order information as well.) In this paper, we also investigate Monte-Carlo techniques for unbiasing [Chen (2020)], improved single-point function evaluation for better gradient estimation [Jongeneel et al. (2021)], as well as the very recent forward-mode unbiased estimator of [Baydin et al. (2022)].

2 Preliminaries

2.1 The System Model

We consider a standard model for the decentralized optimization setting, which is similar to Koloskova et al. (2020a,b), Lian et al. (2017a), Nadiradze et al. (2021). Specifically, we have \( n \geq 2 \) agents, of which \( n_0 \) agents have zeroth-order gradient oracles, and \( n_1 \) have first-order gradient oracles. (We describe the exact optimization setup in the next section.) Beyond their oracle type, the agents are assumed to be anonymous for the purposes of the protocol. The execution will proceed in discrete steps, or rounds, where in each step, two agents are chosen to interact, uniformly at random. Specifically, when chosen, each agent performs some local computation, e.g. obtains some gradient information from their local oracle. Then, the two agents exchange parameter information, and update their local models, after which they are ready to proceed to the next round. Notice that this random interaction model is asynchronous, in the sense that the number of interactions taken by agents up to some point in time may be different, due to randomness. The basic unit of time used in the analysis, which we call fine-grained time, will be the total number of interactions among agents up to some given point in the execution. To express global progress, we will consider parallel time, which is the average number of interactions up to some point, and can be obtained by dividing by \( n \) the total number of interactions. This corresponds to the intuition that \( \Theta(n) \) interactions may occur in parallel. In experiments, we will examine the convergence of the local model at a fixed node.

This model is an instantiation of the classic population model of distributed computing [Angluin et al. (2006)], in an optimization setting. The model is similar to the one adopted by [Nadiradze et al. (2021)] for analyzing asynchronous decentralized SGD, and is more general than the ones adopted by [Koloskova et al. (2020a,b), Lian et al. (2017a), Wang and Joshi (2021)] for decentralized analysis, since the latter assume that nodes are paired via perfect global random matchings in each round. (Our analysis would easily extend to global matching, yielding virtually the same results.)

2.2 Optimization Setup

We assume each node \( i \) has a local data distribution \( D_i \), and that the loss function corresponding to the samples at node \( i \), denoted by \( f^i(x) : \mathbb{R}^d \to \mathbb{R} \) can be approximated using its stochastic form \( F^i(x, \xi) \) for each parameter \( x \in \mathbb{R}^d \) and (randomly chosen) sample \( \xi_i \sim D_i \), where \( f^i(x) = \mathbb{E}_{\xi_i \sim D_i} [F^i(x, \xi_i)] \). For simplicity of notation, we assume that nodes in the set \( N_0 = \{1, 2, ..., n_0\} \) are zeroth-order nodes and the nodes in the set \( N_1 = [n]/N_0 \) are first-order nodes. Let \( n_0 \) and \( n_1 \) be the sizes of the sets \( N_0 \) and \( N_1 \) correspondingly.

In this setup nodes communicate to solve a distributed stochastic optimization problem, i.e.

\[
    f^* = \min_{x \in \mathbb{R}^d} \left[ f(x) := \frac{1}{n_0} \sum_{i \in N_0} f^i(x) + \frac{1}{n_1} \sum_{i \in N_1} f^i(x) \right].
\]

This means that we wish to optimize the function \( f \) which corresponds to the loss over all data samples. Since in the analysis we will wish to throttle the ratio of zeroth-order to first-order agents, we split the entire data among zeroth-order nodes, and we do the same thing for the first-order nodes. (Our analysis can be extended to settings where this is not the case, but this will allow us for instance to study what happens when either \( n_0 \) or \( n_1 \) goes to zero, without changing our objective function.) We make the following assumptions on the optimization objectives:

**Assumption 1** (Strong convexity). We assume that the function \( f \) is strongly convex with parameter \( \ell > 0 \), i.e. for all \( x, y \in \mathbb{R}^d \):

\[
    (x - y)^T (\nabla f(x) - \nabla f(y)) \geq \ell \| x - y \|^2.
\]
Assumption 2 (Smooth gradient). All the stochastic gradients $\nabla F^i$ are $L$-Lipschitz for some constant $L > 0$, i.e. for all $\xi^i \sim D^i$ and $x, y \in \mathbb{R}^d$:

$$\|\nabla F^i(x, \xi^i) - \nabla F^i(y, \xi^i)\| \leq L\|x - y\|. \quad (1)$$

If in addition $F^i$ are convex functions, then

$$\|\nabla F^i(x, \xi^i) - \nabla F^i(y, \xi^i)\| \leq 2L(F^i(x, \xi^i) - F^i(y, \xi^i) - \langle x - y, \nabla F^i(\nu, \xi^i) \rangle).$$

Using Assumption 2 one can easily find that the gradients of $f$ and $f^i(x) \forall i \in [n]$ are also satisfying the above inequalities. Further, we make the following assumptions about the data split and the stochastic gradient estimators:

Assumption 3 (Balanced data distribution). The average variance of $\nabla f^i(x)$s for both zero and first order nodes is bounded by a global constant values, i.e. for all $x \in \mathbb{R}^d$:

$$\frac{1}{n_0} \sum_{i \in N_0} \|\nabla f^i(x) - \nabla f(x)\|^2 \leq \varsigma_0^2;$$

$$\frac{1}{n_1} \sum_{i \in N_1} \|\nabla f^i(x) - \nabla f(x)\|^2 \leq \varsigma_1^2.$$

Assumption 4 (Unbiasedness and bounded variance). For each $i$, $\nabla F^i(x, \xi^i)$ is an unbiased estimator of $\nabla f^i(x)$ and its variance is bounded by a constant $s_i^2$, i.e. for all $x \in \mathbb{R}^d$:

$$\mathbb{E}_{\xi^i \sim D^i}[\nabla F^i(x, \xi^i)] = \nabla f^i(x);$$

$$\mathbb{E}_{\xi^i} \|\nabla F^i(x, \xi^i) - \nabla f^i(x)\| \leq s_i^2.$$

Each node has access to an estimator $G^i(x)$ that estimates the local gradient $\nabla f^i(x)$ at point $x$. For nodes which can perform the gradient computation over a batch of data, i.e. first-order nodes, $G_i^i(x)$ is $\nabla F^i(x, \xi^i)$, where $\xi^i \sim D^i$.

2.3 Zeroth-order Optimization

We now provide a brief introduction relative to standard basic facts and assumptions concerning zeroth-order optimization. Let the function $f^i_n(x) := \mathbb{E}_{\xi^i \sim D^i}[f^i(x + \nu u)], u \sim N(0, I_d)$ be the smoothed version of each function $f^i(x)$. Then, node $i$ can estimate the gradient of $f^i_n$ by only evaluating some points of $f^i$.

Definition 5 (Zeroth-order estimator).

$$G^i_n(x, u, \xi^i) = \frac{F^i(x + \nu u, \xi^i) - F^i(x, \xi^i)}{\nu} u, \quad (2)$$

where $u \sim N(0, I_d)$ and $\xi^i \sim D^i$.

Note that under Assumption 4 one can easily prove that $G^i_n(x, u, \xi^i)$ is an unbiased estimator of $\nabla f^i_n$ since

$$\mathbb{E}_{u, \xi^i}[G^i_n(x, u, \xi^i)] = \mathbb{E}_u \left[ \frac{f^i(x + \nu u) - f^i(x)}{\nu} u \right] = \nabla f^i_n(x). \quad (3)$$

As a technical note, in our analysis we will set $\nu := \frac{\eta}{c}$, where $\eta$ is the learning rate and $c$ is a constant to be defined later. Therefore, for simplicity we can define $G^i(x) := G^i_n(x, u, \xi^i)$, where $G^i_n(x, u, \xi^i)$ is as defined in Definition 5 and $\nu = \frac{\eta}{c}$. Since zeroth-order nodes cannot perform gradient computation directly, we use this $G^i(x)$ as their gradient estimator. We restate the following well-known fact:

Lemma 6 (Nesterov and Spokoiny [2017], Theorem 1.1 in Balasubramanian and Ghadimi [2022]). For a Gaussian random vector $u \sim N(0, I_d)$ we have that

$$\mathbb{E}[\|u\|^k] \leq (d + k)^{k/2} \quad (4)$$

for any $k \geq 2$. Moreover, the following statements hold for any function $f$ whose gradient is Lipschitz continuous with constant $L$.

a) The gradient of $f_\nu$ is Lipschitz continuous with constant $L_\nu$ such that $L_\nu \leq L$. 

4
b) For any \( x \in \mathbb{R}^d \),
\[
|f_\nu(x) - f(x)| \leq \frac{\nu^2}{2} L d,
\]
\[
\|\nabla f_\nu(x) - \nabla f(x)\| \leq \frac{\nu^2}{2} L (d + 3)^2.
\]

c) For any \( x \in \mathbb{R}^n \),
\[
\frac{1}{\nu^2} \mathbb{E}_u[(f(x + \nu u) - f(x))^2\|u\|^2] \leq \frac{\nu^2}{2} L^2(d + 6)^2 + 2(d + 4)\|\nabla f(x)\|^2.
\]

3 The HDO Algorithm

Algorithm Description. We now describe a decentralized optimization algorithm, designed to be executed by a population of \( n \) nodes, interacting in pairs chosen uniformly at random as per our model. We assume that \( n_1 \) of the nodes have access to first-order estimators and \( n_0 \) of them have access to zeroth-order estimators, hence \( n = n_1 + n_0 \). Two copies of the training data are distributed, once among the first-orders and once among the zeroth-orders. Thus, each first- and zeroth-order node has access to \( \frac{1}{n_1} \) of the entire training data, respectively. We assume that each node \( i \) has access to a local stochastic estimator of the gradient, which we denote by \( G^i \), and maintains a model estimate \( X^i \), as well as the global learning rate \( \eta \). Without loss of generality, we assume that the models are initialized to the same randomly-chosen point. Specifically, upon every interaction, the interacting agents \( i \) and \( j \) perform the following steps:

**Algorithm 1:** HDO pseudocode for each interaction between randomly chosen nodes \( i \) and \( j \)

1. // Nodes perform local steps.
2. \( X^i \leftarrow X^i - \eta G^i(X^i) \);
3. \( X^j \leftarrow X^j - \eta G^j(X^j) \);
4. // Nodes average their local models.
5. \( \text{avg} \leftarrow (X^i + X^j)/2 \);
6. \( X^i \leftarrow \text{avg} \);
7. \( X^j \leftarrow \text{avg} \);

Discussion. On the face of it, the algorithm is straightforward: upon each interaction, each node first performs a local model update based on its estimator, and then nodes average their local models following the interaction. Importantly, we do not distinguish between estimator types in the interactions, and nodes are immediately ready to proceed to the next round. Yet, this extremely simple structure in the algorithm comes at the cost of a very careful analysis, which will have to show that the above algorithmic pattern works, in spite of the fact that the nodes have different estimators, with different variances and (potentially) bias properties.

4 The Convergence of the HDO Algorithm

This section is dedicated to proving that the following result holds:

**Theorem 7.** Assume an objective function \( f: \mathbb{R}^d \rightarrow \mathbb{R} \), equal to the average loss over all data samples, whose optimum \( x^* \) we are trying to find using Algorithm 1. Let \( n_0 \) be the number of zeroth-order nodes, and \( n_1 \) be the number of first-order agents. Given the data split described in the previous section, let \( f_i \) be local objective function of node \( i \).

Assume that the functions \( f \) and \( f_i \) satisfy Assumptions 3 and 7. Let the total number of steps in the algorithm \( T \), be large enough such that \( T \log T = \Omega(\frac{n(n_1 + n_0)(L + 1)(\sqrt{\nu} + 1)}{\epsilon}) \), and let the learning rate be \( \eta = \frac{4n \log T}{T \epsilon} \). Assume that zeroth-order nodes use estimators with \( \nu = \frac{n_0}{n_1} \). For \( 1 \leq t \leq T \), let the sequence of weights \( w_t \) be given by \( w_t = (1 - \frac{\eta T}{\epsilon})^{-t} \) and let \( S_T = \sum_{t=1}^{T} w_t T \). Finally, define \( \mu_t = \sum_{t=1}^{n} X_i^t / n \) and \( y_T = \sum_{t=1}^{T} w_t \mu_{t-1} / S_T \) to be the mean over local model parameters. Then, we can show that HDO provides the following convergence rate:
\[
\mathbb{E}[f(y_T) - f(x^*)] + \frac{\ell \mathbb{E} \| \mu_T - x^* \|^2}{8} = O\left( \frac{L \| \mu_0 - x^* \|^2}{T \log T} + \frac{\log(T)(dn_0 \sigma_0^2 + n_1 \varsigma_1^2)}{T \ell n} + \frac{\log(T)(dn_0 \sigma_0^2 + n_1 \sigma_1^2)}{T \ell n} + \frac{\log(T)dn_0}{T \ell n} \right).
\]

**Speedup.** We first emphasize that, in the above bound, the time \( T \) refers to the total number of interactions among agents, as opposed to parallel time, corresponding to \( T/n \). Notice that this rate is reminiscent of that of sequential SGD for the strongly-convex case. However, there are some distinctions: our notion of time is different, as we are counting the total number of gradient oracle queries by the nodes, and there are some additional trailing terms, whose meaning we discuss below.

We interpret this formula from the perspective of an arbitrary local model. For this, notice that the notion of parallel time corresponding to the number of total interactions \( T \), which is by definition \( T_p = T/n \), corresponds (up to constants) to the average number of interactions and gradient oracle queries performed by each node up to time \( T \). Thus, for any single model, convergence with respect to its number of performed SGD steps \( T_p \) would be \( O(\log(nT_p)/(nT_p)) \) (assuming all parameters are constant), which would correspond to \( \Omega(\frac{n}{\log(nT)}) = \Omega(\frac{n}{\log(T)}) \) speedup compared to a variant of sequential SGD. Notice that this is quite favorable to our algorithm, since we are considering biased zeroth-order estimators for some of the nodes in the population. Hence, assuming that \( T \) is polynomial in \( n \), we get an almost-linear speedup of \( \Omega\left( \frac{n}{\log(n)} \right) \).

**Impact of Zeroth-Order Nodes.** Notice that our convergence bound cleanly separates in the terms which come from zeroth-order nodes and terms which come from first-order nodes. For \( n_0 = 0 \), we get asymptotically the same bound as we would get if all nodes performed pure first-order SGD steps. Similarly, when \( n_0 = n \) we should be able to achieve asymptotically-optimal convergence for biased zeroth-order estimators. Further, notice that, if the bias is negligible, then the last term in the upper bound disappears, and we obtain a trade-off between two populations with different variances. We can also observe the following theoretical threshold: we asymptotically match the convergence rate in the case with all nodes performing SGD steps, as long as \( dn_0 = O(n) \) (assuming all other parameters are constant).

### 4.1 Analysis

**Proof Overview.** The convergence proof, given in full in the Appendix, can be split conceptually into two steps. The first aims to bound the variance of the local models \( X_i^t \) for each time step \( t \) and node \( i \) with respect to the mean \( \mu_t = \sum_i X_i^t \). It views this variance as a potential \( \Gamma_t \), which as we show has supermartingale-like behavior for small enough learning rate: specifically, this quantity tends to increase due to gradient steps, but is pushed towards the mean \( \mu_t \) by the averaging process.

The key technical component here is Lemma 8, which provides a careful bound for the evolution of the potential at a step, by modelling optimization as a dynamic load balancing process: each interaction corresponds to a weight generation step (in which gradient estimators are generated) and a load balancing step, in which the “loads” of the two nodes (corresponding to their model values) are balanced through averaging.

In the second step of the proof, we first bound the rate at which the mean \( \mu_t \) converges towards \( x^* \), where we crucially (and carefully) leverage the variance bound obtained above. The main challenge in this part is dealing with biased zeroth-order estimators. In fact, even dealing with biased first-order estimators is not trivial, since for example, they are the main reason for the usage of error feedback when stochastic gradients are compressed using biased quantization [Alistarh et al. 2018]. This is our second key technical result.

With this in hand, we can complete the proof by applying a standard argument which characterizes the rate at which \( \mathbb{E}[f(y_T) - f(x^*)] \) and \( \mathbb{E}[\| \mu_t - x^* \|^2] \) converge towards 0.

**Notation and Preliminaries.** In this section, we provide a more in-depth sketch of the analysis of the HDO protocol. We begin with some notation. Recall that \( n \) is the number of nodes, split into first-order (\( n_1 \)) and zeroth-order (\( n_0 \)). We will analyze a sequence of time steps \( t = 1, 2, \ldots, T \), each corresponding to an individual interaction between two nodes, which are usually denoted by \( i \) and \( j \).

**Step 1: Parameter Concentration.** Next, let \( X_t \) be a vector of model estimates at time step \( t \), that is \( X_t = (X_1^t, X_2^t, \ldots, X_n^t) \). Also, let \( \mu_t = \frac{1}{n} \sum_{i=1}^{n} X_i^t \), be an average estimate at time step \( t \). The following potential function
measures the variance of the models:

$$\Gamma_t = \frac{1}{n} \sum_{i=1}^{n} \|X_i^t - \mu_t\|^2.$$

With this in place, one of our key technical results is to provide a supermartingale-type bound on the evolution of the potential $\Gamma_t$, in terms of $\eta$, and average second moment of estimators at step $t$, defined as $M_t^G := \frac{1}{n} \sum_i \|G^i(X_i^t)\|^2$.

**Lemma 8.** For any time step $t$:

$$\mathbb{E}[\Gamma_{t+1}] \leq (1 - \frac{1}{2n}) \mathbb{E}[\Gamma_t] + \frac{4}{n} \eta^2 \mathbb{E}[M_t^G].$$

Notice that, if we had a universal second moment bound on the estimators, that is, for any vector $X$ and node $i$ $\mathbb{E}[G^i(X)]^2 \leq M$, for some $M > 0$, then we would be able to unroll the recursion, and, for any $t \geq 0$ upper bound $\mathbb{E}[\Gamma_t]$ by $\eta^2 M^2$. In the absence of such upper bound we must derive the following upper bound on $\mathbb{E}[M_t^G]$:

**Lemma 9.** Assume $\nu := \frac{2}{n}$ is fixed, where $\eta$ and $c$ are the learning rate and a constant respectively. Then, for any time step $t$ we have:

$$\mathbb{E}[M_t^G] \leq 6(d+4)\eta L^2 \mathbb{E}[\Gamma_t] + \frac{6}{n} \left[ \frac{6(d+4)n_0\varsigma^2 + 3\varsigma_1^2}{n} \right] + 6(2d+9)\mathbb{E}[f(\mu_t) - f(x^*)] + \frac{2}{n} \left[ \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2 \right] + \eta^2 \frac{n_0}{2nc^2} L^2 (d+6)^3.$$

First, we check how this upper bound affects the upper bound given by Lemma 8. For small enough $\eta$, the term containing $\mathbb{E}[\Gamma_t]$ (which comes from the upper bound on $\mathbb{E}[M_t^G]$) can be upper bounded by $\frac{1}{4n} \mathbb{E}[\Gamma_t]$, and hence it will just change the factor in front of $\mathbb{E}[\Gamma_t]$ to $(1 - 1/4n)$.

Second, since the above bound contains the term with $\mathbb{E}[f(\mu_t) - f(x^*)]$ we are not able to bound the potential $\Gamma$ per step, instead, for weights $w_t = (1 - \frac{\eta}{\eta_0})^{-t}$, we can upper bound $\sum_{t=1}^{T} w_t \mathbb{E}[\Gamma_{t-1}]$ (please see Lemma 23 in the Appendix). The crucial property is that the upper bound on the weighted sum $\sum_{t=1}^{T} w_t \mathbb{E}[\Gamma_{t-1}]$, is

$$O(\eta^2 \sum_{t=1}^{T} w_t \mathbb{E}[f(\mu_{t-1}) - f(x^*)]) + \sum_{t=1}^{T} w_t O(\eta^2).$$

(for simplicity, above we assumed that all other parameters are constant.)

**Step 2: Convergence of the Mean and Risk Bound.** The above result allows us to characterize how well the individual parameters are concentrated around their mean. In turn, this will allow us to provide a recurrence for how fast the parameter average is moving towards the optimum. To help with the intuition, we provide the lemma which is simplified version of the one given in the additional material (please see Lemma 24):

**Lemma 10.** For small enough $\eta$ and $t \geq 1$ we have that:

$$\mathbb{E}\left[\left\|\mu_t - x^*\right\|^2\right] \leq (1 - \frac{\eta}{2n}) \mathbb{E}\left[\|\mu_{t-1} - x^*\|^2\right] - \Omega(\frac{\eta}{n}) \mathbb{E}\left[f(\mu_{t-1}) - f(x^*)\right] + O\left(\frac{\eta^2}{n}\right) \mathbb{E}[\Gamma_{t-1}] + O\left(\frac{\eta^2}{n^2}\right).$$

Note that $O$ and $\Omega$ hide all other parameters (we assume that all other parameters are constant). As mentioned, the main challenge in the proof of this lemma is taking care of biased zeroth-order estimators.

Recall that $w_t = (1 - \frac{\eta}{\eta_0})^{-t}$, by definition. We proceed by multiplying both sides of the above inequality by $w_t$ and then summing it up for $1 \leq t \leq T$. Then, once we plug the upper bound on $\sum_{t=1}^{T} w_t \mathbb{E}[\Gamma_{t-1}]$, for small enough $\eta$ the term $\Omega(\frac{\eta}{n}) O(\frac{\eta^2}{n}) \sum_{t=1}^{T} w_t \mathbb{E}[f(\mu_{t-1} - f(x^*))]$ vanishes as it is dominated by the term $- \sum_{t=1}^{T} \Omega(\frac{\eta}{n}) \mathbb{E}[f(\mu_{t-1}) - f(x^*)].$
5 Experimental Results

Experimental Setup and Goals. In this section, we validate our results numerically by implementing HDO, and examining its behaviour in different scenarios. To investigate the convergence behaviour of our algorithm in a setting that matches our analysis, we investigated its behavior on real-world classification tasks from LibSVM [Chang and Lin, 2011]; in addition, to investigate a more realistic scenario, in which nodes jointly fine-tune the last (classification) layer of a ResNet50 deep neural network (DNN) on the CIFAR-10 dataset. Our main goal is examining whether, under reasonable parameter settings, zeroth-order nodes can be used to enhance the optimization process. Further details regarding the models and datasets are presented in the Appendix B. Our full experimental setup, including code, is available at the following URL: https://anonymous.4open.science/r/Hybrid-Decentralized-Optimization-BCE1

In this section, we validate our results numerically by simulating HDO’s execution for varying numbers of nodes, varying ratios between first- and zeroth-order nodes, and various “strengths” for the zeroth order gradient estimators. Specifically, we are interested in the convergence behavior of the algorithm, relative to the total number of optimization steps, measured as either the loss value over time, or, alternatively, as the accuracy on the hidden validation set.

Results. In the first experiment, described in Figure 1, we examine the performance of individual zeroth-order gradient estimators over time, as a function of the number of random vectors used for the the gradient estimation. We use the CIFAR-10 finetuning task as an example. We choose values 5, 10 and 15 for the number of random vectors, and compare against the unbiased forward-only estimator recently proposed by Baydin et al. [2022]. (We have found the performance of the latter estimator to be similar to a de-biased regular one [Chen, 2020], and we therefore omit results for explicit de-biasing so as to not overcrowd the figure.) The results clearly show an accuracy-vs-steps advantage for higher number of random vectors, and for the unbiased zeroth-order estimators vs. biased ones. Since the computational overhead of unbiasing estimators is fairly low, we will adopt unbiased zeroth-order estimators in the following experiments.

In the second experiment, executed on the Flowers classification task [Chang and Lin, 2011], and described in Figure 2, we examine the convergence speed, in terms of training loss at a fixed node, for various sizes of homogeneous populations, containing only one type of estimators. The node is chosen so that its number of interactions is in the median, taken among all nodes. Specifically, we compare the convergence of the node in a system with 1, 6, or 12 zeroth-order (ZO) nodes, each using 50 random vectors for estimation, relative to a system with 1 or 6 first-order (FO) nodes. The results confirm the intuition, as well as our analysis: first-order nodes always outperform the same number of zeroth-order ones; however, zeroth-order nodes can in fact outperform first-order ones if their number is larger (see relationship between 1 FO and 6 ZO, and between 6 FO and 12 ZO).

In our third experiment (Figure 3), we fix a population size \( n = 16 \), and examine the impact of the ratio between \( n_1 \), the number of first-order nodes, and \( n_0 \), the number of zeroth-order nodes, on the convergence of the algorithm. Here, we return to the task of final-layer CNN fine-tuning on CIFAR-10. As expected, the population formed exclusively of...
first-order agents has the fastest convergence, while the convergence order follows the intuition that more first-orders in
the population provide faster convergence, which is backed up by our analysis.

Our next experiment, presented in Figure 4, aims to examine whether a hybrid system, formed of both FO and ZO
agents, can provide a convergence boost relative to a homogeneous system, formed by only one type of agents. (We
examine the Flowers classification task, but results are identical for CIFAR-10 classification as well.) For this reason,
we deploy either 2 FO or ZO nodes, 6 ZO nodes, or a hybrid system formed of 2 FO and 6 ZO nodes. The results
show that 1) a larger population of 6 ZO nodes can outperform smaller populations of 2 FO or ZO nodes; and that 2)
this larger uniform population is itself outperformed by a hybrid population. Results are obtained over 5 parallel runs,
providing us with the confidence intervals shown in the figure.

6 Discussion, Limitations, and Future Work

We have provided a first analysis of the convergence of decentralized gradient-based methods in a population of nodes
which mixes first- and zeroth-order gradient estimators. Our analysis shows that, even when biased or very noisy,
information from zeroth-order agents can still be successfully incorporated into a given protocol, and can in fact provide
convergence improvements.

The above experimental results clearly validate our analysis and the initial premise of our paper, by showing that
first-order and zeroth-order estimators can in fact be successfully hybrid-ized in a decentralized population of agents.
This is good news for environments combining computational devices with heterogeneous computational powers, and
shows that one can leverage some agents’ local data even if the agents do not have the ability to extract gradients.

Specifically, a practical embodiment of our approach could be a decentralized learning system in which some more
computationally-powerful agents perform backpropagation, acting as first-order agents, whereas a larger fraction of the
nodes, with computationally-bounded devices, only estimate gradient information based on forward-passes over their
local data, and share this information with the overall system during pair-wise interactions.

Our analysis can also be generalized to tackle more general underlying interaction graph topologies: we omitted
this here for brevity, however the potential analysis can be extended to general regular interaction graphs, where the
convergence of the algorithm will depend on the eigenvalue gap of the given graph. Another possible extension which
we plan to investigate is that of additional gradient estimators estimators, and of larger-scale practical deployments to
validate the applicability of our approach in practical settings. The goal of our experimental simulation has been to
validate the practical feasibility of our approach, and we estimate that this goal has been achieved.

7 Acknowledgement

This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020
research and innovation programme (grant agreement No 805223 ScaleML). The authors would like to acknowledge
Eugenia Iofinova for useful discussions during the inception of this project.
References

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT’2010, pages 177–186. Springer, 2010.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jio Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. arXiv preprint arXiv:1705.09056, 2017a.

Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of local-update sgd algorithms. Journal of Machine Learning Research, 22(213):1–50, 2021. URL http://jmlr.org/papers/v22/20-147.html

Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. In International Conference on Learning Representations, 2020a. URL https://openreview.net/forum?id=SkgGCrKvH

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U. Stich. A unified theory of decentralized sgd with changing topology and local updates. In ICML, pages 5381–5393, 2020b. URL http://proceedings.mlr.press/v119/koloskova20a.html

Giorgi Nadiradze, Amir-mojtaba Sabour, Peter Davies, Shigang Li, and Dan Alistarh. Asynchronous decentralized sgd with quantized and local updates. Advances in Neural Information Processing Systems, 34, 2021.

Guanting Chen. Unbiased gradient simulation for zeroth-order optimization. In 2020 Winter Simulation Conference (WSC), pages 2947–2959, 2020. doi:10.1109/WSC48552.2020.9384045.

Dana Angluin, James Aspnes, Zoë Diamadi, Michael J Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. Distributed computing, 18(4):235–253, 2006.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. Foundations of Computational Mathematics, 17(2):527–566, 2017.

Atılım Güneş Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation, 2022. URL https://arxiv.org/abs/2202.08587

John Nikolas Tsitsiklis. Problems in decentralized decision making and computation. Technical report, Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, 1984.

David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In 44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings., pages 482–491. IEEE, 2003.

Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. Systems & Control Letters, 53(1):65–78, 2004.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. IEEE Transactions on Automatic Control, 54(1):48, 2009.

Björn Johansson, Maben Rabi, and Mikael Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. SIAM Journal on Optimization, 20(3):1157–1170, 2009.

Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 850–857. IEEE, 2014.

Ermin Wei and Asuman Ozdaglar. Distributed alternating direction method of multipliers. In 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), pages 5445–5450. IEEE, 2012.

Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In 52nd IEEE conference on decision and control, pages 3671–3676. IEEE, 2013.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. arXiv preprint arXiv:1710.06952, 2017b.

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Michael Rabbat. Stochastic gradient push for distributed deep learning. arXiv preprint arXiv:1811.10792, 2018.

Hanlin Tang, Ce Zhang, Shaoduo Gan, Tong Zhang, and Ji Liu. Decentralization meets quantization. CoRR, abs/1803.06443, 2018.

Anit Kumar Sahu and Soummya Kar. Decentralized zeroth-order constrained stochastic optimization algorithms: Frank–wolfe and variants with applications to black-box adversarial attacks. Proceedings of the IEEE, 108(11):1890–1905, 2020.
Deming Yuan, Lei Wang, Alexandre Proutiere, and Guodong Shi. Distributed zeroth-order optimization: Convergence rates that match centralized counterpart. 2021.

Krishnakumar Balasubramanian and Saeed Ghadimi. Zeroth-order nonconvex stochastic optimization: Handling constraints, high dimensionality, and saddle points. Found. Comput. Math., 22(1):35–76, feb 2022. ISSN 1615-3375. doi:10.1007/s10208-021-09499-8. URL https://doi.org/10.1007/s10208-021-09499-8

Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. arXiv preprint cs/0408007, 2004.

Alekh Agarwal, Ofer Dekel, and Lin Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In Colt, pages 28–40. Citeseer, 2010.

Ohad Shamir. An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. The Journal of Machine Learning Research, 18(1):1703–1713, 2017.

Wouter Jongeneel, Man-Chung Yue, and Daniel Kuhn. Small errors in random zeroth-order optimization are imaginary, 2021. URL https://arxiv.org/abs/2103.05478

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cedric Renggli. The convergence of sparsified gradient methods. In NIPS, pages 5977–5987, 2018.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. ACM Trans. Intell. Syst. Technol., 2(3):27:1–27:27, May 2011. ISSN 2157-6904. doi:10.1145/1961189.1961199. URL http://doi.acm.org/10.1145/1961189.1961199

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). Available Online, 2010. URL http://www.cs.toronto.edu/~kriz/cifar.html

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In Indian Conference on Computer Vision, Graphics and Image Processing, Dec 2008.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747
A Appendix

B Experimental setup

In this section, we describe our experimental setup in detail. We begin by carefully describing the way in which we simulated HybridSGD in the sequential form. Then, we proceed by explaining different types of gradient estimators that we used in our experiments, together with their implementation methods. Finally, we detail the datasets, tasks, and models used for our experiments.

B.1 Simulation

We attempt to simulate a realistic decentralized deployment scenario sequentially, as follows. We assume \( n \) nodes, each of which initially has a model copy. Each node has an oracle to estimate the gradient of loss function with respect to its model. It is assumed that \( n_1 \) nodes have access to first-order oracle and \( n_0 \) nodes have access to zeroth-order oracle, which can be biased or unbiased. Two copies of training dataset are distributed among first- and zeroth-order nodes so that each first-zeroth-order node has access to \( \frac{1}{n_1} / \frac{1}{n_0} \) of training data. At each simulation step, we select two nodes uniformly at random and make them interact with each other. During the interaction, first each node takes a SGD step and then they share their models and adapt the averaged model as their new model. To track the performance of our algorithm, we select the node which has taken median number of steps among the population, and evaluate its model on an unseen validation dataset. We measure Validation loss and accuracy of the model with respect to its local steps. Moreover, we keep track of the training loss by updating it after each interaction using \( \alpha_t = 0.95 * \alpha_{t-1} + 0.05 * \beta_t \) where \( \alpha \) and \( \beta \) are the training loss and average interaction loss respectively.

B.2 Estimator types

First-order Using this estimator node \( i \) can estimate the gradient \( \nabla f^i(X^i) \) by computing \( \nabla F^i(X^i, \xi^i) \), where \( f^i \) and \( F^i \) are the node’s local loss function and its stochastic estimator respectively. The computation is done using Pytorch built-in .backward() method.

Unbiased Zeroth-order We implemented this estimator using forward-mode differentiation technique inspired by . Using this method, for a randomly chosen vector \( u \sim N(0, I_d) \), node \( i \) can compute \( \tilde{F}^i(X^i, \xi^i) \) and \( u.\nabla F^i(X^i, \xi^i) \) in a single forward pass. It will then use \( (u.\nabla F^i(X^i, \xi^i))u \) as its gradient estimator. Note that the node does not need to compute \( u.\nabla F^i(X^i, \xi^i) \) hence it is a zeroth-order estimation of the gradient. Moreover, \( E_{u \sim N(0, I_d)}[(u.\nabla F^i(X^i, \xi^i))u] = \nabla F^i(X^i, \xi^i) \) which means the gradient estimator is unbiased.

Biased Zeroth-order For a fixed \( \nu \) and a randomly chosen \( u \sim N(0, I_d) \), node \( i \) can estimate the gradient \( \nabla F^i(X^i, \xi^i) \) simply by computing \( F^i(X^i+\nu u, \xi) - F^i(X^i, \xi) \) or \( F^i(X^i+\nu u, \xi) - F^i(X^i-\nu u, \xi) \). The computations consist of evaluating only function values, thus they are called zeroth-order estimators. However, both of them are biased estimators as their expected values would be equal to the gradient of the smoothed-version of the function, \( \nabla F_u^i(X^i, \xi^i) \), which is close but not necessarily equal to \( \nabla F^i(X^i, \xi) \).

Note that the approximation of zeroth-order estimators can be improved by increasing the number of randomly chosen vectors and averaging the results. For the biased zeroth-order estimators, we use the batch matrix multiplication to compute the function values for all the randomly chosen vectors using constant GPU calls. Moreover, for unbiased zeroth-order estimators, we simulate the forward-mode differentiation by computing the gradient followed by computing the dot products of the gradient and the randomly chosen vectors. For more details on the implementation, we encourage readers to look at the source code of our experiments.

B.3 Datasets and Models

We used Pytorch to manage the training process in our algorithm. We tested our algorithm on two sets of experiments, 1) Training a linear model, on the Year Prediction dataset Dua and Graff [2017], 2) Training the last (classification) layer of a ResNet50 deep neural network, on the CIFAR-10 Krizhevsky and Hinton [2009] and Flowers Nilsson and Zisserman [2008] datasets, and 3) Training a Convolutional Neural Network as a non-convex function on the Fashion MNIST Xiao et al. [2017] dataset. In the following, we describe the model architecture, the training hyper-parameters as well as the way that we prepared the datasets.

Training a linear model For this task, we used the Year Prediction dataset, a collection of songs each of which with 90 features mapped to their release year, an integer from 1922 to 2011. We applied the standardization on the samples’ input and min-max normalization on their outputs. We used \( 2^{17}/10000 \) random split for the
train/validation datasets. As the model, we used a linear layer of size \((90,1)\). To train the model, we used MSELoss with batch size \(= 128\). The learning rate was usually set to 0.001; it is stated otherwise.

**Training the classification layer of a ResNet50 deep neural network** We did this task on two of the well-known image classification datasets, CIFAR-10 and Flowers. For each dataset, we trained a raw ResNet50 model with output features equal to the number of classes of the dataset: 10 for the CIFAR-10 and 102 for the Flowers. We trained the models for 150 epochs using SGD optimizer and initial learning rate of 0.001. We decayed the learning rate by a factor of 10 every 50 epochs. After the training process, passed each dataset through its trained model and stored the inputs of the last layer paired with their labels as a new dataset. Then to run our experiments, we used these extracted datasets to train a randomly initialized linear model of size \((2048, \text{number of classes})\), as if we were tuning the classification layer of our ResNet50 model. We used the default train/test data split of the dataset for our training/validation datasets. We always used CrossEntropyLoss as the criterion and batch size \(= 128\) for these datasets. During the linear tuning, the learning was set to 0.001; it is stated otherwise.

**Training a Convolutional Neural Network as a Non-Convex** For this task we used the well-known Fashion MNIST dataset with its default train/test split as our training/validation datasets, with no augmentation. As the model we used a full CNN architecture with two basic blocks followed by three fully connected layers and a dropout layer; for the details of the architecture we encourage readers to look at our source code. We used CrossEntropyLoss as the criterion and batch size is set to 128.

### B.4 Experimental results

In the following there are additional experiments validating our earlier results, presented in the paper. Alignment of our results on different datasets ensures the robustness of our statements.

Figure 5: The impact of number of random vectors on the biased and unbiased zeroth-order estimators on CIFAR-10.

Figure 6: The effect of \(n_1/n_0\) on the validation accuracy for the fixed population size of 16 nodes on CIFAR-10.

Figure 7: Training loss versus number of nodes for all biased population on CIFAR-10 dataset.
C Zeroth-order Stochastic Gradient Properties.

Lemma 11. Let $G^i_u(x, u, \xi^i)$ be computed by $5$. Then, under Assumptions 2 and 4 we have:

$$\mathbb{E}_{u, \xi^i} \|G^i_u(x, u, \xi^i)\|^2 \leq \frac{1}{2} \nu^2 L^2 (d + 6)^3 + 2(d + 4) \left[ \|\nabla f^i(x)\|^2 + s_i^2 \right],$$  \hspace{1cm} (8)

$$\mathbb{E}_{u, \xi^i} \|G^i_u(x, u, \xi) - \nabla f^i(x)\|^2 \leq \frac{3\nu^2}{2} L^2 (d + 6)^3 + 4(d + 4) \left[ \|\nabla f^i(x)\|^2 + s_i^2 \right].$$  \hspace{1cm} (9)

Proof. Firstly, by plugging in $F^i(x, \xi^i)$ in 7 under Assumptions 2 and 4 we obtain

$$\mathbb{E} \|G^i_u(x, u, \xi)\|^2 \leq \frac{1}{2} \nu^2 L^2 (d + 6)^3 + 2(d + 4) \|\nabla F^i(x, \xi^i)\|^2$$

Then by getting an expectation and eliminating the randomness of the right-hand side with respect to $\xi^i$, we get

$$\mathbb{E}_{u, \xi^i} \|G^i_u(x, u, \xi^i)\|^2 \leq \frac{1}{2} \nu^2 L^2 (d + 6)^3 + 2(d + 4) \|\nabla F^i(x, \xi^i)\|^2$$  \hspace{1cm} (Assumption 4)

$$\leq \frac{1}{2} \nu^2 L^2 (d + 6)^3 + 2(d + 4) \left[ \|f^i(x)\|^2 + s_i^2 \right].$$

Secondly, using 3 we have:

$$\mathbb{E} \|G^i_u(x, u, \xi) - \nabla f^i(x)\|^2 = \mathbb{E} \|G^i_u(x, u, \xi)\|^2 + \|\nabla f^i(x)\|^2 - 2\mathbb{E} \langle G^i_u(x, u, \xi), \nabla f^i(x) \rangle \leq \mathbb{E} \|G^i_u(x, u, \xi)\|^2$$

$$- \|\nabla f^i(x)\|^2 \leq 0$$

$$\leq \frac{1}{2} \nu^2 L^2 (d + 6)^3 + 2(d + 4) \left[ \|f^i(x)\|^2 + s_i^2 \right].$$

Finally, together with 5 and the inequality above we can deduce:

$$\mathbb{E}_{u, \xi^i} \|G^i_u(x, u, \xi^i) - \nabla f^i(x)\|^2 \leq 2 \mathbb{E}_{u, \xi^i} \|G^i_u(x, u, \xi^i) - \nabla f^i_u(x)\|^2 + 2\|\nabla f^i_u(x) - \nabla f^i(x)\|^2 \leq \nu^2 L^2 (d + 6)^3 + 4(d + 4) \left[ \|f^i(x)\|^2 + s_i^2 \right] + 2\|\nabla f^i_u(x) - \nabla f^i(x)\|^2$$

$$\leq \frac{3\nu^2}{2} L^2 (d + 6)^3 + 4(d + 4) \left[ \|f^i(x)\|^2 + s_i^2 \right].$$
D Definitions

For the sake of simplicity, we now define some notations for the frequently-used expressions in the proof.

**Definition 12 (Gamma).**
\[ \Gamma_t := \frac{1}{n} \sum_i \| X_t^i - \mu_t \|^2. \]  

**Definition 13 (Average second-moment of estimator).**
\[ M_f^G := \frac{1}{n} \sum_i \| G^i(X_t^i) \|^2. \]  

**Definition 14 (Expectation conditioned step).**
\[ E_t[Y] := E[Y|X_t^1, X_t^2, ..., X_t^n]. \]  

**Definition 15 (Biasedness of estimators).** For node \( i \), using \( G^i(x) \) as its gradient estimator we define \( b_i \) as the upper bound for its biasedness, i.e.
\[ \| \nabla f^i(x) - E[G^i(x)] \| \leq b_i. \]  

Note that for an unbiased estimator we have \( b_i = 0 \). Moreover, for zeroth-order estimators \( E[G^i(x)] = \nabla f^i(x) \).

Hence, according to \( \mathbb{E}[\| \nabla f^i(x) - E[G^i(x)] \|] \) is bounded for a fixed \( \nu \). Therefore, \( b_i \) is well-defined in our setup.

We further define the average biasedness of estimators as
\[ B := \frac{1}{n} \sum_i b_i. \]  

**Definition 16 (Variance of estimators).** For node \( i \), using \( G^i(X_t^i) \) as its gradient estimator at step \( t \) we define \( (\sigma_i^t)^2 \) as the upper-bound of its variance, i.e.
\[ E[\| \nabla f^i(X_t^i) - G^i(X_t^i) \|^2 \] \[ \leq (\sigma_i^t)^2. \]  

Note that for the first-order nodes, i.e. \( G^i(x) = \nabla F^i(x) \), using \( \mathbb{E}[\| \nabla f^i(X_t^i) \|^2 = 3L^2L^2(d+6)^3 + 4(d+4)[\| \nabla f^i(X_t^i) \|^2 + s_t^2] \), which is well-defined considering that \( \nu \) is fixed in our setup.

We further define the average variance of estimators as
\[ (\bar{\sigma}_t)^2 := \frac{1}{n} \sum_i (\sigma_i^t)^2. \]  

E Useful Inequalities

**Lemma 17 (Young).** For any pair of vectors \( x, y \) and \( \alpha > 0 \) we have
\[ \langle x, y \rangle \leq \frac{\| x \|^2}{2\alpha} + \frac{\alpha \| y \|^2}{2}. \]  

**Lemma 18 (Cauchy-Schwarz).** For any vectors \( x_1, x_2, ..., x_n \in \mathbb{R}^d \) we have
\[ \| \sum_{i=1}^n x_i \|^2 \leq n \sum_{i=1}^n \| x_i \|^2. \]  

F The Complete Convergence Proof

In this part, we assume that there exist \( n_0 \) zeroth-order nodes and \( n_1 \) first-order nodes, all having access to a shared dataset, hence a shared objective function \( f \) that they want to minimize.

**Lemma 19.** For any time step \( t \) and constants \( \alpha_0 \geq \alpha_1 > 0 \) let \( M_{f}^t(\alpha_0, \alpha_1) = \frac{\alpha_0}{n} \sum_{i \in N_0} \mathbb{E}[\| \nabla f^i(X_t^i) \|^2 + \frac{\alpha_1}{n} \sum_{i \in N_1} \mathbb{E}[\| \nabla f^i(X_t^i) \|^2]. \) We have that:
\[ \mathbb{E}[M_{f}^t(\alpha_0, \alpha_1)] \leq 3L^2\alpha_0 \mathbb{E}[\Gamma_t] + \frac{3\alpha_0n_0\sigma_t^2 + 3\alpha_1n_1\sigma_t^2}{n} + 6(\alpha_0 + \alpha_1)L \mathbb{E}[f(\mu_t) - f(x^*)]. \]
Proof.
\[
\frac{\alpha_0}{n} \sum_{i \in N_0} \mathbb{E} \left\| \nabla f^i(X^i_t) \right\|^2 = \frac{\alpha_0}{n} \sum_{i \in N_0} \mathbb{E} \left\| \nabla f(X^i_t) - \nabla f^i(\mu_t) + \nabla f^i(\mu_t) - \nabla f(\mu_t) + \nabla f(\mu_t) - \nabla f(x^*) \right\|^2
\]

\[
\leq 3L^2 \alpha_0 \sum_{i \in N_0} \mathbb{E} \left\| X^i_t - \mu_t \right\|^2 + \frac{3\alpha_0 n_0 s_0^2}{n} + 6\alpha_0 \mathbb{E} \left[ f(\mu_t) - f(x^*) \right].
\]

Similarly, in the case of first-order nodes we get:
\[
\frac{\alpha_1}{n} \sum_{i \in N_1} \mathbb{E} \left\| \nabla f^i(X^i_t) \right\|^2 \leq 3L^2 \alpha_1 \sum_{i \in N_1} \mathbb{E} \left\| X^i_t - \mu_t \right\|^2 + \frac{3\alpha_1 n_1 s_1^2}{n} + 6\alpha_1 \mathbb{E} \left[ f(\mu_t) - f(x^*) \right].
\]

By summing up the above inequalities and using the fact that $\alpha_1 \geq \alpha_2$ (together with the definition of $\Gamma_t$), we get the proof of the lemma.

Lemma 9. Assume $\nu := \frac{\nu c}{\alpha}$ is fixed, where $\eta$ and $c$ are the learning rate and a constant respectively. Then, for any time step $t$ we have:
\[
\mathbb{E} \left[ M_t^G \right] \leq 6(d + 4)L^2 \mathbb{E}[\Gamma_t] + \frac{6(d + 4)n_0^2 + 3n_1 s_1^2}{n} + 6(2d + 9)L \mathbb{E} \left[ f(\mu_t) - f(x^*) \right] + \frac{2(d + 4)}{n} \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} \mathbb{E} \left[ \nabla f^i(X^i_t) \right]\right\|^2 + s_i^2)
\]

Next, we take expectation with respect to $X^1_t, X^2_t, ..., X^n_t$ and use Lemma 19 to get:
\[
\mathbb{E} \left[ M_t^G \right] \leq 6(d + 4)L^2 \mathbb{E}[\Gamma_t] + \frac{6(d + 4)n_0^2 + 3n_1 s_1^2}{n} + 6(2d + 9)L \mathbb{E} \left[ f(\mu_t) - f(x^*) \right] + \frac{2(d + 4)}{n} \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} \mathbb{E} \left[ \nabla f^i(X^i_t) \right]\right\|^2 + s_i^2)
\]

Which finishes the proof of the lemma.

Lemma 20. Assume $\nu := \frac{\nu c}{\alpha}$ is fixed, where $\eta$ and $c$ are the learning rate and a constant respectively. Then, for any time step $t$ we have
\[
\mathbb{E} \left[ (\sigma_t)^2 \right] \leq 12(d + 4)L^2 \mathbb{E}[\Gamma_t] + \frac{12n_0(d + 4)s_0^2 + 3n_1 s_1^2}{n} + 6(4d + 17)L \mathbb{E} \left[ f(\mu_t) - f(x^*) \right] + \frac{4(d + 4)}{n} \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} \mathbb{E} \left[ \nabla f(X^i_t) \right]\right\|^2 + s_i^2)
\]

Proof.
\[
\mathbb{E} \left[ (\sigma_t)^2 \right] = \frac{1}{n} \sum_i \mathbb{E} \left[ (\sigma_t^i)^2 \right] \leq \frac{1}{n} \sum_i \left( \frac{3\nu}{2} L^2 (d + 6)^3 + 4(d + 4) \left[ \mathbb{E} \left\| \nabla f(X^i_t) \right\|^2 + s_i^2 \right] \right) + \frac{1}{n} \sum_i \left( \mathbb{E} \left\| \nabla f^i(X^i_t) \right\|^2 + s_i^2 \right)
\]

\[
= \mathbb{E}[M_t^G (4(d + 4), 1)] + \frac{4(d + 4)}{n} \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} \mathbb{E} \left[ \nabla f(X^i_t) \right]\right\|^2 + s_i^2) + \mathbb{E} \left[ \nabla f^i(X^i_t) \right]\right\|^2 + s_i^2) + \frac{3n_0}{2nc^2} L^2 (d + 6)^3.
\]
Next, we take expectation with respect to $X_1^t, X_2^t, ..., X_n^t$ and use Lemma 19 to get:

$$
\mathbb{E}_t[(\sigma_t)^2] \leq 12(d + 4)L^2 \mathbb{E}[\Gamma_t] + \frac{12n_0(d + 4)s_i^2}{n} + 6(4d + 17)L(f(\mu_t) - f(x^*)) \\
+ \frac{4(d + 4)\sum_{i \in N_0} s_i^2}{n} + \sum_{i \in N_1} s_i^2 + \eta^2 \frac{3n_0}{2nc^2}L^2(d + 6)^3.
$$

Which finishes the proof of the lemma. □

**Lemma 21.** Assume $\nu := \frac{\eta}{c}$ is fixed, where $\eta$ and $c$ are the learning rate and a constant respectively. Then, for any time step $t$ we have

$$
B \leq \eta \frac{n_0}{2cn} L(d + 3)^{\frac{3}{2}}.
$$

**Proof.**

$$
B = \frac{1}{n} \sum_i b_i = \frac{1}{n} \sum_i \|\nabla f^i(X_i^t) - \mathbb{E}[G^i(X_i^t)]\| = \frac{1}{n} \sum_{i \in N_0} \|\nabla f^i(X_i^t) - \nabla f^\nu(X_i^t)\|
$$

\begin{equation}
\leq \frac{\nu n_0}{2n} L(d + 3)^{\frac{3}{2}} = \eta \frac{n_0}{2cn} L(d + 3)^{\frac{3}{2}}.
\end{equation}

□

**Lemma 8.** For any time step $t$ :

$$
\mathbb{E} [\Gamma_{t+1}] \leq (1 - \frac{1}{2n}) \mathbb{E} [\Gamma_t] + \frac{4}{n} \eta^2 \mathbb{E} [M^G_t].
$$

**Proof.** First we can open $\mathbb{E}_t [\Gamma_{t+1}]$ as

$$
\mathbb{E}_t [\Gamma_{t+1}] = \mathbb{E}_t \left[ \frac{1}{n} \sum_i \|X_{i+1}^t - \mu_{t+1}\|^2 \right].
$$

Observe that in this case $\mu_{t+1} = \mu_t - \eta(G^i_t(X_i^t) + G^j_t(X_j^t))/n$ and $X_{i+1}^t = X_{i+1}^j = (X_i^t + X_j^t)/2 - \eta(G^i_t(X_i^t) + G^j_t(X_j^t))/2$. 

17
Hence,

\[
E_t[\Gamma_{t+1}] = \frac{1}{n^2(n-1)} \sum_{i \neq j} E_t \left[ 2 \left\| (X_t^i + X_t^j)/2 - \left( \frac{n-2}{2n} \right) \eta(G^i(X_t^i) + G^j(X_t^j)) - \mu_t \right\|^2 + \sum_{k \neq i,j} \left\| X_t^k - \mu_t + \frac{\eta}{n} (G^i(X_t^i) + G^j(X_t^j)) \right\|^2 \right]
\]

\[
= \frac{1}{n^2(n-1)} \sum_{i \neq j} E_t \left[ 2 \left( \left\| (X_t^i + X_t^j)/2 - \mu_t \right\|^2 + \left( \frac{n-2}{2n} \right)^2 \eta^2 \left\| G^i(X_t^i) + G^j(X_t^j) \right\|^2 \right.ight.
\]
\[
- \left( \frac{n-2}{n} \right) \eta \langle G^i(X_t^i) + G^j(X_t^j), (X_t^i + X_t^j)/2 - \mu_t \rangle
\] + \sum_{k \neq i,j} \left( \left\| X_t^k - \mu_t \right\|^2 + \left( \frac{1}{n} \right)^2 \eta^2 \left\| G^i(X_t^i) + G^j(X_t^j) \right\|^2
\]
\[
+ \frac{2}{n} \eta \langle G^i(X_t^i) + G^j(X_t^j), X_t^k - \mu_t \rangle \right) \right]
\]

\[
\leq \left( 1 - \frac{1}{n} \right) E_t[\Gamma_t] - \frac{1}{n^2(n-1)} \eta \sum_{i \neq j} \sum E_t \left( G^i(X_t^i) + G^j(X_t^j), X_t^i + X_t^j - 2\mu_t \right)
\]

\[
+ \frac{1}{n^2(n-1)} \sum_{i \neq j} E_t(X_t^i - \mu_t, X_t^j - \mu_t) + \frac{1}{2n^2(n-1)} \eta^2 \sum_{i \neq j} \sum E_t \left\| G^i(X_t^i) + G^j(X_t^j) \right\|^2
\]

\[
\leq \left( 1 - \frac{1}{n} \right) E_t[\Gamma_t] + \frac{1}{n^2(n-1)} \sum_{i \neq j} E_t(X_t^i - \mu_t, X_t^j - \mu_t) + \frac{2}{n^2} \eta^2 \sum_{i} E_t \left\| G^i(X_t^i) \right\|^2.
\]

\[
P_1 := \frac{1}{n^2(n-1)} \eta \sum_{i \neq j} E_t \left( G^i(X_t^i) + G^j(X_t^j), X_t^i + X_t^j - 2\mu_t \right)
\]

\[
P_2 := \frac{2}{n^2(n-1)} \eta \left( \sum_{i \neq j} E_t \left( G^i(X_t^i), X_t^j - \mu_t \right) + (n-1) \sum E_t \left( G^i(X_t^i), X_t^i - \mu_t \right) \right)
\]

\[
= \frac{2(n-2)}{n^2(n-1)} \sum_{i} E_t \left( \eta G^i(X_t^i), X_t^i - \mu_t \right) \leq \frac{1}{n^2} \sum \left( 2\eta^2 E_t \left\| G^i(X_t^i) \right\|^2 + \frac{1}{2} E_t \left\| X_t^i - \mu_t \right\|^2 \right)
\]

\[
= \frac{2}{n} \eta^2 E_t[M^G_t] + \frac{1}{2n} E_t[\Gamma_t].
\]
By using (25) and (26) in inequality (24), we get
\[
E_t \left[ \Gamma_{t+1} \right] \leq (1 - \frac{1}{n}) E_t \left[ \Gamma_t \right] - \frac{1}{n(n-1)} E_t \left[ \Gamma_t \right] + \frac{2}{n} \eta^2 E_t \left[ M_t^G \right] + \frac{2}{n} \eta^2 E_t \left[ M_t^G \right] + \frac{1}{2n} E_t \left[ \Gamma_t \right]
\]
\[
\leq (1 - \frac{1}{2n}) E_t \left[ \Gamma_t \right] + \frac{4}{n} \eta^2 E_t \left[ M_t^G \right].
\]
(27)

Finally, by taking the expectation with respect to \(X_t^1, X_t^2, ..., X_t^n\) we will have
\[
E \left[ \Gamma_{t+1} \right] \leq (1 - \frac{1}{2n}) E \left[ \Gamma_t \right] + \frac{4}{n} \eta^2 E \left[ M_t^G \right]
\]
(28)

\[\square\]

**Lemma 22.** For any time step \(t\) and fixed learning rate \(\eta \leq \frac{1}{14L(d+4)^2}\)

\[
E \left[ \Gamma_{t+1} \right] \leq (1 - \frac{1}{4n}) E \left[ \Gamma_t \right] + \frac{12\eta^2 (2d + 4)n_0s_0^2 + n_1s_1^2}{n^2} + \frac{24\eta^2 (2d + 9)L E[f(\mu_t)] - f(x^*)}{n}
\]
\[
+ \frac{4\eta^2 (2d + 4)}{n^2} \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2 + \frac{n_0}{2nc^2} L^2 (d + 6)^3/2
\]

\[
= (1 - \frac{1}{2n}) \left( \frac{6(d + 4)L^2 r_t}{n} + \frac{6(d + 4)n_0s_0^2 + 3n_1s_1^2}{n} + \frac{6(2d + 9)L (f(\mu_t) - f(x^*))}{n}
\]
\[
+ \frac{2(d + 4)}{n} \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2 + \frac{\eta^2 n_0}{2nc^2} L^2 (d + 6)^3/2
\]

Now, by using Lemma 8 in the inequality above we have

\[
E \left[ \Gamma_{t+1} \right] \leq (1 - \frac{1}{2n}) E \left[ \Gamma_t \right] + \frac{4\eta^2}{n} \left( \frac{6(d + 4)L^2 r_t}{n} + \frac{6(d + 4)n_0s_0^2 + 3n_1s_1^2}{n} + \frac{6(2d + 9)L (f(\mu_t) - f(x^*))}{n}
\]
\[
+ \frac{2(d + 4)}{n} \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2 + \frac{n_0}{2nc^2} L^2 (d + 6)^3/2
\]

We get the proof of the lemma by using \(\eta \leq \frac{1}{14L(d+4)^2}\) in the above inequality. \[\square\]

Next, we define the following weights: for any step \(t \geq 0\), let \(w_t = (1 - \frac{\eta L}{2n})^{-t}\). This allows us to prove the following lemma:

**Lemma 23.** For any \(T \geq 0\) and \(\eta \leq \frac{1}{10T}\):

\[
\sum_{t=1}^{T} w_t E[\Gamma_{t-1}] \leq 120\eta^2 (2d + 9)L \sum_{t=1}^{T-1} w_t E[f(\mu_{t-1}) - f(x^*)]
\]
\[
+ \left( \frac{60\eta^2 (2d + 4)n_0s_0^2 + n_1s_1^2}{n} + \frac{20\eta^2 (2d + 4)}{n} \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2 + \frac{10\eta^4 n_0L^2 (d + 6)^3}{nc^2} \right) \sum_{t=1}^{T-1} w_t.
\]

\[\text{Proof.}\] Let \(P_t = \frac{24\eta^2 (2d + 9)L E[f(\mu_t) - f(x^*)]}{n}\)
and let \(Q = \frac{12\eta^2 (2d + 4)n_0s_0^2 + n_1s_1^2}{n^2} + \frac{4\eta^2 (2d + 4)}{n^2} \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2 + \frac{2\eta^4 n_0L^2 (d + 6)^3}{nc^2}\). Then the above lemma gives
us that for any $t \geq 0$: $\mathbb{E}[\Gamma_{t+1}] \leq (1 - \frac{1}{4n}) \mathbb{E}[\Gamma_t] + P_t + Q$. After unrolling the recursion, we get that for any $t > 1$, $\mathbb{E}[\Gamma_t] \leq \sum_{i=0}^{t-1} (P_t + Q)(1 - \frac{1}{4n})^{t-1-i}$. Hence,

$$
\sum_{t=1}^{T} w_t \mathbb{E}[\Gamma_{t-1}] \leq \sum_{t=1}^{T} w_t \left( \sum_{i=0}^{t-2} (P_t + Q)(1 - \frac{1}{4n})^{t-2-i} \right) = \sum_{t=0}^{T-2} (P_t + Q) \sum_{i=t+2}^{T} w_t (1 - \frac{1}{4n})^{i-t} = (1 - \frac{\eta T}{2n})^{-1} \sum_{t=0}^{T-2} (P_t + Q) \sum_{i=t+2}^{T} w_t (1 - \frac{\eta T}{2n})^{-(i-(t+2))} (1 - \frac{1}{4n})^{i-(t+2)}
$$

(29)

$$
= (1 - \frac{\eta T}{2n})^{-1} \sum_{t=0}^{T-2} w_{t+1} (P_t + Q) \sum_{j=0}^{T-(t+2)} \left( \frac{1}{1 - \frac{1}{4n}} \right)^j.
$$

For $\frac{1}{10T} \geq \eta$, we have $r := \frac{1 - \frac{1}{4n}}{1 - \frac{1}{4n}} \leq 1$. Hence, we can write

$$
\sum_{j=0}^{T-(t+2)} \left( \frac{1}{1 - \frac{1}{4n}} \right)^j = \sum_{j=0}^{T-(t+2)} r^j = \frac{1 - r^{T-(t+1)}}{1 - r} \leq 1 - r.
$$

(30)

By using the above inequality in (29) we have

$$
\sum_{t=1}^{T} w_t \mathbb{E}[\Gamma_{t-1}] \leq (1 - \frac{\eta T}{2n})^{-1} \sum_{t=0}^{T-2} w_{t+1} (P_t + Q) = \frac{1}{4n} \sum_{t=0}^{T-2} w_t (P_{t} + Q)
$$

Finally, since $\frac{1}{10T} \geq \eta$ we get $\frac{1}{4n} \sum_{t=0}^{T-2} w_t (P_{t} + Q) \leq 5n$ and the proof of lemma is finished.

**Lemma 24.** For $\eta \leq \frac{\sqrt{\eta T \log(d+3)}}{2\sqrt{Tn_0d}}$, we have that

$$
\mathbb{E}\left[\|\mu_{t+1} - x^*\|^2\right] \leq \frac{2}{n} \mathbb{E}\left[\|\mu_t - x^*\|^2\right] - 4\frac{\eta}{n} \mathbb{E}\left[\|\mu_t - x^*\|^2\right] - \frac{16L(12d + 52)}{n^2} \mathbb{E}\left[\|\mu_t - x^*\|^2\right] + \eta^2 \frac{64BL}{n^3} \mathbb{E}\left[\|\mu_t - x^*\|^2\right]
$$

$$
+ \left(2 \frac{L + \ell}{n} \eta + \eta^2 \frac{L^2(96d + 456)}{n^2} + \eta^4 \frac{32BL^2}{n^3}\right) \mathbb{E}[\Gamma_t]
$$

$$
+ \eta^2 \frac{(96d + 448)s_0^2 + 88s_1^2}{n^3} + \frac{8\eta^2(\eta + 1) \sum_{i=0}^{N_0} s_i^2 + \sum_{i=N_1}^{N} s_i^2}{n^3} + \frac{12\eta^4n_0L^2(d + 6)^2}{n^5c^2} + \frac{\eta^22B}{n}
$$

**Proof.** Let $F_t$ be the amount by which $\mu_t$ decreases at step $t$. So, $F_t$ is a sum of $\frac{2}{n}G^i(X_t^i)$ and $\frac{2}{n}G^j(X_t^j)$ for agents $i$ and $j$, which interact at step $t$. Also, let $F'_t$ be the amount by which $\mu_t$ would decrease if all the agents were contributing at that step using their true local gradients. That is $F'_t = \frac{2}{n} \sum_i \nabla f^i(X_t^i)$.

To make the calculations more clear, let us define $E_t[Y] := E[Y | X_t^1, X_t^2, \ldots, X_t^n]$

$$
\mathbb{E}\left[\|\mu_{t+1} - x^*\|^2\right] = \mathbb{E}\left[\mu_t - F_t - x^*\right]\mathbb{E}\left[\mu_t - F_t - x^*\right] = \mathbb{E}\left[\mu_t - F_t - x^* - F'_t + F'_t\right]\mathbb{E}\left[\mu_t - F_t - x^* - F'_t + F'_t\right]
$$

$$
= \mathbb{E}\left[\mu_t - x^* + F'_t\right]\mathbb{E}\left[\mu_t - x^* + F'_t\right] + \mathbb{E}\left[\mu_t - x^* + F'_t\right]\mathbb{E}\left[\mu_t - x^* + F'_t\right] + 2 \mathbb{E}\left[\mu_t - x^* + F'_t\right]\mathbb{E}\left[\mu_t - x^* + F'_t\right]
$$

$$
+ 2 \mathbb{E}[X_t^1, X_t^2, \ldots, X_t^n] \left[\mathbb{E}_t \left[F'_t - F_t\right] \mathbb{E}_t \left[F'_t - F_t\right] \right]
$$

(31)

This means that in order to upper bound $\mathbb{E}\left[\|\mu_{t+1} - x^*\|^2\right]$, we need to upper bound $\mathbb{E}\left[\mu_t - x^* - F'_t\right]^2$, $\mathbb{E}_t \left[F'_t - F_t\right]^2$, and $\mathbb{E}_t \left[\mu_t - x^* - F'_t, F'_t - F_t\right]$.  

20
For the second one, when $X_1, X_2, \ldots, X_n$ are fixed, we have that

\[
\|\mu_t - x^* - F_t^i\|^2 = \|\mu_t - x^* - \frac{2\eta}{n^2} \sum_i \nabla f^i(X_i^t)\|^2
\]

\[
= \|\mu_t - x^*\|^2 + \frac{4\eta^2}{n^2} \left(\frac{1}{n} \sum_i \nabla f^i(X_i^t)\right)^2 - 4\frac{\eta}{n} \left(\frac{1}{n} \sum_i \nabla f^i(X_i^t)\right) \tag{32}
\]

\[
R_1 = \left\|\frac{1}{n} \sum_i \nabla f^i(X_i^t)\right\|^2 = \left\|\frac{1}{n} \sum_i \nabla f^i(X_i^t) - \nabla f^i(\mu_t) + \nabla f^i(\mu_t) - \nabla f^i(x^*)\right\|^2
\]

\[
\leq \frac{2L^2}{n} \sum_i \|X_i^t - \mu_t\|^2 + \frac{4L}{n} \sum_i (f^i(\mu_t) - f^i(x^*))
\]

\[
= \frac{2L^2}{n} \sum_i \|X_i^t - \mu_t\|^2 + 4L \left[f(\mu_t) - f(x^*)\right] \tag{33}
\]

\[
R_2 = \left\langle \mu_t - x^*, \frac{1}{n} \sum_i \nabla f^i(X_i^t)\right\rangle = \frac{1}{n} \sum_i \left\langle \mu_t - X_i^t + X_i^t - x^* , \nabla f^i(X_i^t)\right\rangle
\]

\[
= \frac{1}{n} \sum_i \left[ \left\langle \mu_t - X_i^t , \nabla f^i(X_i^t)\right\rangle + \left\langle X_i^t - x^* , \nabla f^i(X_i^t)\right\rangle \right] \tag{34a}
\]

Using L-smoothness property (Assumption\[\text{3}\]) with $y = X_i^t$ and $x = x^*$ we have

\[
\left\langle \mu_t - X_i^t , \nabla f^i(X_i^t)\right\rangle \geq f^i(\mu_t) - f^i(X_i^t) - \frac{L}{2} \|\nabla f^i(\mu_t) - \nabla f^i(X_i^t)\|^2.
\]

Additionally, we use the $\ell$-strong convexity (Assumption\[\text{1}\]), to get

\[
\left\langle X_i^t - x^* , \nabla f^i(X_i^t)\right\rangle \geq (f^i(X_i^t) - f^i(x^*)) + \frac{\ell}{2} \|X_i^t - x^*\|^2.
\]

Now by plugging (34b) and (34c) in inequality (34a) we get that

\[
R_2 \geq \frac{1}{n} \sum_i \left[ f(\mu_t) - f(x^*) - \frac{L}{2} \|\nabla f^i(\mu_t) - \nabla f^i(X_i^t)\|^2 + f^i(X_i^t) - f^i(x^*) + \frac{\ell}{2} \|X_i^t - x^*\|^2 \right]
\]

\[
= \left[ f(\mu_t) - f(x^*) \right] - \frac{L}{2n} \sum_i \|X_i^t - \mu_t\|^2 + \frac{\ell}{2n} \sum_i \|X_i^t - x^*\|^2 \tag{35}
\]

Now we plug (33) and (35) back into (32) and take expectation into the account to get

\[
\mathbb{E} \left\|\mu_t - x^* - F_t^i\right\|^2 \leq \mathbb{E} \left\|\mu_t - x^*\right\|^2 + \frac{4\eta^2}{n^2} \left(\frac{2L^2}{n} \sum_i \|X_i^t - \mu_t\|^2 + 4L \mathbb{E} \left[f(\mu_t) - f(x^*)\right]\right)
\]

\[
- 4\frac{\eta}{n} \left( f(\mu_t) - f(x^*) \right) - \frac{L + \ell}{2n} \sum_i \mathbb{E} \left\|X_i^t - \mu_t\right\|^2 + \frac{\ell}{4} \mathbb{E} \|\mu_t - x^*\|^2 \tag{36}
\]

For the second one we have that:
\[
\mathbb{E}_t \left\| F'_t - F_t \right\|^2 = \frac{1}{n(n-1)} \sum_i \sum_{i \neq j} \mathbb{E}_t \left\| \frac{2\eta}{n^3} \sum_r \nabla f'(X'_t) - \frac{\eta}{n} (G_i(X'_t) + G^j(X'_t)) \right\|^2 \\
\leq \frac{4\eta^2}{n^3} \sum_i \mathbb{E}_t \left\| \frac{1}{n} \sum_r \nabla f'(X'_t) - G_i(X'_t) \right\|^2 \leq \frac{4\eta^2}{n^3} \sum_i \mathbb{E}_t \left\| \frac{1}{n} \sum_r \nabla f'(X'_t) - \nabla f(X'_t) + \nabla f(X'_t) - G_i(X'_t) \right\|^2 \\
\leq \frac{8\eta^2}{n^3} \sum_i \left( \mathbb{E}_t \left\| \frac{1}{n-1} \sum_{r \neq i} \nabla f'(X'_t) - \nabla f(X'_t) \right\|^2 + \mathbb{E}_t(\sigma_i)^2 \right) \\
\leq \frac{8\eta^2}{n^3} \sum_i \left( \mathbb{E}_t \left\| \nabla f'(X'_t) - \nabla f(X'_t) \right\|^2 + \mathbb{E}_t(\sigma_i)^2 \right) \\
\leq \frac{8\eta^2}{n^3(n-1)} \sum_i \sum_{r \neq i} \mathbb{E}_t \left\| \nabla f'(X'_t) - \nabla f(X'_t) \right\|^2 + \frac{8\eta^2}{n^2} \mathbb{E}_t(\sigma_i)^2 \\
\leq \frac{8\eta^2}{n^3(n-1)} \sum_i \sum_{r \neq i} \mathbb{E}_t \left\| \nabla f'(X'_t) - \nabla f'(\mu_t) + [\nabla f'(\mu_t) - \nabla f(\mu_t)] \\
+ [\nabla f(\mu_t) - \nabla f_i(\mu_t)] + [\nabla f_i(\mu_t) - \nabla f(X'_t)] \right\|^2 + \frac{8\eta^2}{n^2} \mathbb{E}_t(\sigma_i)^2 \\
\leq \frac{8\eta^2}{n^3(n-1)} \sum_i 8(n-1) \left( \mathbb{E}_t \left\| \nabla f'(X'_t) - \nabla f'(\mu_t) \right\|^2 + \mathbb{E}_t \left\| \nabla f'(\mu_t) - \nabla f(\mu_t) \right\|^2 \right) + \frac{8\eta^2}{n^3} \mathbb{E}_t(\sigma_i)^2 \\
\leq \frac{64\eta^2}{n^3} \sum_i \mathbb{E}_t \left\| \nabla f'(X'_t) - \nabla f'(\mu_t) \right\|^2 + \frac{64\eta^2}{n^3} \sum_i \mathbb{E}_t \left\| \nabla f'(\mu_t) - \nabla f(\mu_t) \right\|^2 + \frac{8\eta^2}{n^2} \mathbb{E}_t(\sigma_i)^2 \\
\leq \frac{64L^2\eta^2}{n^3} \mathbb{E}_t(\Gamma_t) + \frac{64\eta^2(\varsigma^2 n_0 + \varsigma^2 n_1)}{n^3} + \frac{8\eta^2}{n^2} \mathbb{E}_t(\sigma_i)^2 \\
= \frac{64L^2\eta^2}{n^2} \mathbb{E}_t(\Gamma_t) + \frac{64\eta^2(\varsigma^2 n_0 + \varsigma^2 n_1)}{n^3} + \frac{8\eta^2}{n^2} \mathbb{E}_t(\sigma_i)^2. \\
(37)
\]

Next, we remove conditioning and use Lemma 20 to get
\[
\mathbb{E} \left\| F'_t - F_t \right\|^2 = \mathbb{E} \left[ \mathbb{E}_t \left\| F'_t - F_t \right\|^2 \right] \\
\leq \frac{64L^2\eta^2}{n^2} \mathbb{E}[\Gamma_t] + \frac{64\eta^2(\varsigma^2 n_0 + \varsigma^2 n_1)}{n^3} + \frac{8\eta^2}{n^2} \mathbb{E}(\sigma_i)^2 \\
\leq \frac{64L^2\eta^2}{n^2} \mathbb{E}[\Gamma_t] + \frac{64\eta^2(\varsigma^2 n_0 + \varsigma^2 n_1)}{n^3} \\
+ \frac{8\eta^2}{n^2} \left( 12(d+4)L^2 \mathbb{E}[\Gamma_t] + \frac{12n_0(d+4)\varsigma^2 + 3n_1\varsigma^2}{n} + 6(4d+17)L(f(\mu_t) - f(x^*)) \\
+ \frac{4(d+4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2}{n} + \eta^2 \frac{3n_0}{2nc^2} L^2(d+6)^3 \right) \\
= \frac{\eta^2 L^2(96d+448)}{n^2} \mathbb{E}[\Gamma_t] + \frac{\eta^2(96d+448)\varsigma^2 n_0 + 88\varsigma^2 n_1}{n^3} \\
+ \frac{48\eta^2(4d+17)L \mathbb{E}[f(\mu_t) - f(x^*)]}{n^2} + \frac{8\eta^2((d+4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2)}{n^3} + \frac{12\eta^4 n_0 L^2(d+6)^3}{n^3 c^2}. \\
(38)
\]

Now consider the last one. We have:
\[
\mathbb{E}_t \left( \mu_t - x^* - F'_t, F'_t - F_t \right) = \left\langle \mu_t - F'_t - x^*, \mathbb{E}_t(F'_t - F_t) \right\rangle \\
\leq \left\| \mu_t - x^* - F'_t \right\| \cdot \left\| \mathbb{E}_t(F'_t - F_t) \right\| \\
\leq \left\| \mu_t - x^* \right\| + \left\| F'_t \right\| \cdot \left\| \mathbb{E}_t(F'_t - F_t) \right\|. \\
(39)
\]
\[ R_3 = \| \mathbb{E}_t (F'_t - F_t) \| = \frac{2\eta}{n^2} \sum_i \nabla f^i (X^i_t) - \frac{2\eta}{n^2} \mathbb{E}_t \left[ G^i (X^i_t) \right] \]
\[ \leq \frac{2\eta}{n^2} \sum_i \| \nabla f^i (X^i_t) - \mathbb{E}_t \left[ G^i (X^i_t) \right] \| \leq \frac{2\eta^2}{n^2} \sum_i b_i = \frac{2\eta^2}{n^2} B \] (40)

By using the inequality above in (39) and taking expectation from both sides we get
\[ \mathbb{E} \left[ \mu_t - x^* - F'_t, F'_t - F_t \right] = \mathbb{E}_{X^1_t, X^2_t, \ldots, X^n_t} \left[ \mathbb{E}_t \left[ \mu_t - x^* - F'_t, F'_t - F_t \right] \right] \]
\[ \leq \frac{2\eta^2}{n} B \left( \mathbb{E} \| \mu_t - x^* \| + \mathbb{E} \| F'_t \| \right) \leq \frac{2\eta^2}{n} B \left( \mathbb{E} \| \mu_t - x^* \|^2 + \frac{1}{4} + \mathbb{E} \| F'_t \|^2 + \frac{1}{4} \right) \]
\[ \leq \frac{2\eta^2}{n} B \left( \mathbb{E} \| \mu_t - x^* \|^2 + \frac{4\eta^2}{n^2} \mathbb{E} \left[ \frac{2L^2}{n} \sum_i \| X^i_t - \mu_t \|^2 \right] + 4L \left[ f(\mu_t) - f(x^*) \right] \right) + 0.5 \]
\[ \leq \eta^2 \frac{2B}{n} \mathbb{E} \| \mu_t - x^* \|^2 + \eta^4 \frac{16BL^2}{n^3} \mathbb{E} \| \Gamma_t \| + \eta^4 \frac{32BL}{n^3} \mathbb{E} \left[ f(\mu_t) - f(x^*) \right] + \eta^2 \frac{B}{n} \] (41)

To get the (*) inequality, we first used Young’s inequality twice with \( \alpha = \frac{1}{2} \), to get that
\[ \mathbb{E} \| \mu_t - x^* \|^2 + \frac{1}{4} + (\mathbb{E} \| F'_t \|^2) \leq (\mathbb{E} \| \mu_t - x^* \|^2) + \frac{1}{4} + (\mathbb{E} \| F'_t \|^2) + \frac{1}{4} \]
and then applied Jensen’s inequality to get
\[ (\mathbb{E} \| \mu_t - x^* \|^2) + \frac{1}{4} + (\mathbb{E} \| F'_t \|^2) + \frac{1}{4} \leq \mathbb{E} \| \mu_t - x^* \|^2 + \frac{1}{4} + \mathbb{E} \| F'_t \|^2 + \frac{1}{4} \]

Then following by (36), (37) and (41), the latter inequality (31) would be:
\[ \mathbb{E} \left\| \mu_{t+1} - x^* \right\|^2 = \mathbb{E} \left\| \mu_t - x^* - F'_t, F'_t - F_t \right\|^2 + \mathbb{E}_{X^1_t, X^2_t, \ldots, X^n_t} \left[ \mathbb{E}_t \left( \mu_t - x^* - F'_t, F'_t - F_t \right) \right] \]
\[ \leq (1 - \frac{\ell \eta}{n}) \mathbb{E} \| \mu_t - x^* \|^2 - \left( \frac{4\eta}{n} - 16L \frac{\eta^2}{n^2} \right) \mathbb{E} \left[ f(\mu_t) - f(x^*) \right] + \left( \frac{2L}{n} + \frac{\ell}{n} \right) \mathbb{E} \| \Gamma_t \| \]
\[ + \frac{\eta^2 (96d + 448)}{n^2} \mathbb{E} \| \Gamma_t \| + \frac{\eta^2 (96d + 448)^2}{n^3} n_0 + 88L^2 n_1 \]
\[ + \frac{48\eta^2 (4d + 17)L}{n^2} \mathbb{E} \left[ f(\mu_t) - f(x^*) \right] + \frac{8\eta^2 ((d + 4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2) + 12\eta^4 n_0 L^2 (d + 6)^3}{n^3 c^2} \]
\[ + \frac{\eta^2 (4d + 17)^2 L^2}{n^3} \mathbb{E} \left[ f(\mu_t) - f(x^*) \right] + \frac{44BL}{n^3} \mathbb{E} \left[ f(\mu_t) - f(x^*) \right] + \eta^2 \frac{2B}{n} \]

Hence:
\[ \mathbb{E} \left\| \mu_{t+1} - x^* \right\|^2 \leq (1 - \frac{\ell \eta}{n} + \frac{4B}{n^3}) \mathbb{E} \| \mu_t - x^* \|^2 - \left( \frac{4\eta}{n} - \eta^2 \frac{16L(12d + 52)}{n^2} - \eta^4 \frac{64BL}{n^3} \right) \mathbb{E} \left[ f(\mu_t) - f(x^*) \right] \]
\[ + \left( \frac{2L + \ell}{n} \left( \frac{4B}{n^3} \right) + \frac{\eta^2 (32BL^2)}{n^3} \right) \mathbb{E} \| \Gamma_t \| \]
\[ + \frac{\eta^2 (96d + 448)^2}{n^3} n_0 + 88L^2 n_1 + \frac{8\eta^2 ((d + 4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2) + 12\eta^4 n_0 L^2 (d + 6)^3}{n^3 c^2} + \eta^2 \frac{2B}{n} \]

We get the proof of the Lemma by plugging \( \eta \leq \frac{\sqrt{\gamma n}}{2\sqrt{L}n_0 (d+3)^2} \) and \( B \leq \frac{\eta^3 n_0 L(d + 3)^2}{\gamma n} \) (Lemma 21) in the above inequality.
Theorem 7. Assume an objective function \( f : \mathbb{R}^d \to \mathbb{R} \), equal to the average loss over all data samples, whose optimum \( x^* \) we are trying to find using Algorithm 7. Let \( n_0 \) be the number of zeroth-order nodes, and \( n_1 \) be the number of first-order agents. Given the data split described in the previous section, let \( f_i \) be local objective function of node \( i \).

Assume that the functions \( f \) and \( f_i \) satisfy Assumptions 1, 2, 3 and 4. Let the total number of steps in the algorithm \( T \), be large enough such that \( \frac{T}{\log T} = \Omega \left( \frac{n(d+\nu)(L+1)(\frac{1}{2}+\nu)}{\nu} \right) \), and let the learning rate be \( \eta = \frac{4n_0 \log T}{T} \). Assume that zeroth-order nodes use estimators with \( \nu = \frac{n}{\sqrt{d}} \). For \( 1 \leq t \leq T \), let the sequence of weights \( w_t \) be given by \( w_t = (1 - \frac{n t}{2n})^{-t} \) and let \( S_T = \sum_{t=1}^{T} w_t \). Finally, define \( \mu_t = \sum_{i=1}^{n} x_i^t / n \) and \( y_T = \sum_{t=1}^{T} \frac{w_t \mu_t}{S_T} \) to be the mean over local model parameters. Then, we can show that HDO provides the following convergence rate:

\[
\mathbb{E}[f(y_T) - f(x^*)] + \frac{\ell \mathbb{E}[\|\mu_T - x^*\|^2]}{8} = O \left( \frac{L \|\mu_0 - x^*\|^2}{T \log T} + \frac{\log(T) (dn_0 \sigma^2 + n_1 \sigma^2_1)}{T \bar{n}} + \frac{\log(T) (dn_0 \sigma^2_2 + n_1 \sigma^2_{12})}{T \bar{n}} + \frac{\log(T) dn_0}{T \bar{n}} \right).
\]

Proof. Let \( a_t^2 := \mathbb{E}[\|\mu_t - x^*\|^2], e_t := \mathbb{E}[f(\mu_t) - f(x^*)] \) and

\[
C_1 := 4 \frac{n}{\eta} - \eta^2 \frac{16L(12d + 52)}{n^2} - \eta^2 \frac{64BL}{n^3} \geq 4 \frac{n}{\eta} - \eta^2 \frac{16L(12d + 52)}{n^2} - \eta^2 \frac{32n_0 L^2(d + 3)^{\frac{3}{2}}}{d^{\frac{3}{2}} n^4},
\]

\[
C_2 := \frac{2L + \ell}{n} \eta + \eta^2 \frac{L^2 96d + 456}{n^2} + \eta^2 \frac{32BL^2}{n^3} \leq 2 \frac{L + \ell}{n} \eta + \eta^2 \frac{L^2 96d + 456}{n^2} + \eta^2 \frac{16n_0 L^3(d + 3)^{\frac{3}{2}}}{d^{\frac{3}{2}} n^4},
\]

\[
C_3 := \frac{2L + \ell}{n} \eta + \eta^2 \frac{(96d + 448) c_0^2 n_0 + 88s^2_1}{n^3} + \frac{8\eta^2 (d + 4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2}{n^3} + \frac{12\eta^4 n_0 L^2(d + 6)^3}{n^3 d} + \frac{2B \eta^2}{n} \leq \frac{2L + \ell}{n} \eta + \eta^2 \frac{(96d + 448) c_0^2 n_0 + 88s^2_1}{n^3} + \frac{8\eta^2 (d + 4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2}{n^3} + \frac{12\eta^4 n_0 L^2(d + 6)^3}{n^3 d} + \frac{L(d + 3)^{\frac{3}{2}} n_0 \eta^3}{d^{\frac{3}{2}} n^4}.
\]

Where, in the above inequalities, we used Lemma 21. Therefore, the recursion from Lemma 24 can be written as

\[
a_t^2 \leq (1 - \frac{\ell \eta}{2n}) a_{t-1}^2 - C_1 e_{t-1} + C_2 \mathbb{E}[\Gamma_{t-1}] + C_3.
\]

Then, we multiply the above recursion by \( w_t = (1 - \frac{\eta t}{2n})^{-t} \) and

\[
w_t a_t^2 \leq w_t \left( (1 - \frac{\ell \eta}{2n}) a_{t-1}^2 - C_1 e_{t-1} + C_2 \mathbb{E}[\Gamma_{t-1}] + C_3 \right) = w_{t-1} a_{t-1}^2 - w_t C_1 e_{t-1} + w_t C_2 \mathbb{E}[\Gamma_{t-1}] + w_t C_3.
\]

By summing the above inequality for \( t \in \{1, 2, ..., T\} \) and cancelling and rearrange terms we get:

\[
w_T a_T^2 \leq w_0 a_0^2 - C_1 \sum_{t=1}^{T} w_t e_{t-1} + C_2 \sum_{t=1}^{T} w_t \mathbb{E}[\Gamma_{t-1}] + C_3 \sum_{t=1}^{T} w_t.
\]
By using Lemma 23 in the inequality above we get that

$$w_T a_T^2 \leq w_0 a_0^2 - C_1 \sum_{t=1}^{T} w_t e_{t-1} + 120 C_2 \eta^2 (2d + 9)L \sum_{t=1}^{T-1} w_t E[f(\mu_{t-1}) - f(x^*)]$$

$$+ C_2 \left( \frac{60 \eta^2 (2d + 4)n_0 \sigma_0^2 + n_1 \sigma_1^2}{n} + \frac{20 \eta^2 (2d + 4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2}{n} + \frac{10 \eta^4 n_0 L^2 (d + 6)}{nc^2} \right)\sum_{t=1}^{T-1} w_t$$

$$+ C_3 \sum_{t=1}^{T} w_t$$

$$\leq w_0 a_0^2 - (C_1 - 120 C_2 \eta^2 (2d + 9)L) \sum_{t=1}^{D_1:=} w_t e_{t-1}$$

$$+ \left( C_2 \left( \frac{60 \eta^2 (2d + 4)n_0 \sigma_0^2 + n_1 \sigma_1^2}{n} + \frac{20 \eta^2 (2d + 4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2}{n} + \frac{10 \eta^4 n_0 L^2 (d + 6)}{nc^2} \right) + C_3 \right) \sum_{t=1}^{D_2:=} w_t.$$  \hspace{1cm} (42)

Let $S_T := \sum_{t=1}^{T} w_t$ and $y_T := \sum_{t=1}^{T} \frac{w_t \mu_{t-1}}{S_T}$, then by convexity of $f$ we have that

$$E[f(y_T) - f(x^*)] \leq \frac{1}{S_T} \sum_{t=1}^{T} w_t e_{t-1}.$$  \hspace{1cm} (43)

By choosing small enough $\eta$, so that have $D_1 \geq 0$, we can combine inequalities (42) and (43) to get

$$E[f(y_T) - f(x^*)] + \frac{w_T a_T^2}{S_T D_1} \leq \frac{w_0 a_0^2}{S_T D_1} + \frac{D_2}{D_1}.$$  \hspace{1cm} (44)

Our first goal is to lower bound $D_1$, we aim to choose upper bound on $\eta$ so that $D_1 = \Omega(\frac{n}{\eta})$. For this it will be enough to set $\eta = O(\frac{2}{n \log(T)})$. Since $C_1 \geq 4 \frac{2}{n} - \eta^2 16L(12d+52) \frac{1}{n^2} - \eta^4 32n_0 L^2 (d+3) \frac{3}{2cn^4}$, we have $C_1 = \Omega(\frac{n}{\eta})$. Plus, $C_2 = O(1/n)$ and hence $120 C_2 \eta^2 (2d + 9)L = O(\frac{2}{n})$, thus $D_1 = \Omega(\frac{n}{\eta})$, as desired. Also:

$$S_T = \sum_{t=1}^{T} w_t = \sum_{t=1}^{T} \left( 1 - \frac{\eta T}{2n} \right)^{-1} \geq \left( 1 - \frac{\eta T}{2n} \right)^{-T} \geq e^{\frac{\eta T}{2n}}.$$  \hspace{1cm} (45)

By setting $\eta = \frac{4n \log(T)}{T^2}$, we get $S_T \geq T^2$. Therefore, we have

$$\frac{w_0 a_0^2}{S_T D_1} = O \left( \frac{w_0 a_0^2 \ell}{T \log(T)} \right) = O \left( \frac{L \|x^*\|^2}{T \log(T)} \right).$$

Next, we upper bound $D_2$. Since $\eta = O(\frac{1}{\eta (L + \ell \eta / n)})$ we get that $C_2 = O((L + \ell) \eta / n)$. Additionally,

$$\frac{60 \eta^2 (2d + 4)n_0 \sigma_0^2 + n_1 \sigma_1^2}{n} + \frac{20 \eta^2 (2d + 4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2}{n} + \frac{10 \eta^4 n_0 L^2 (d + 6)}{nd}$$

$$= O \left( \frac{\eta^2 (dn_0 \sigma_0^2 + n_1 \sigma_1^2)}{n} + \frac{\eta^2 (dn_0 \sigma_0^2 + n_1 \sigma_1^2)}{n} + \frac{\eta^2 n_0}{n} \right).$$

By using $\eta = O(\frac{1}{(L + \ell + 1)n})$ in the equation above, we get

$$C_2 \left( \frac{60 \eta^2 (2d + 4)n_0 \sigma_0^2 + n_1 \sigma_1^2}{n} + \frac{20 \eta^2 (2d + 4) \sum_{i \in N_0} s_i^2 + \sum_{i \in N_1} s_i^2}{n} + \frac{10 \eta^4 n_0 L^2 (d + 6)}{nd} \right)$$

$$= O \left( \frac{\eta^2 (dn_0 \sigma_0^2 + n_1 \sigma_1^2)}{n^3} + \frac{\eta^2 (dn_0 \sigma_0^2 + n_1 \sigma_1^2)}{n^3} + \frac{\eta^2 n_0}{n^3} \right).$$
Finally, we have that 
\[ C_3 = O\left(\frac{\eta^2(dn_0\zeta_0^2 + n_1\zeta_1^2)}{n^3} + \frac{\eta^2(dn_0\sigma_0^2 + n_1\sigma_1^2)}{n^3} + \frac{\eta^3n_0}{n^3} + \frac{\eta^3dLn_0}{n^2}\right). \]

If we put together the above inequalities we get that 
\[ D_2 = O\left(\frac{\eta^2(dn_0\zeta_0^2 + n_1\zeta_1^2)}{n^2} + \frac{\eta^2(dn_0\sigma_0^2 + n_1\sigma_1^2)}{n^2} + \frac{\eta^3Ldn_0}{n}\right). \]

By plugging \( D_1 = \Omega\left(\frac{n}{T\ell}\right) \) and \( \eta = \frac{4n\log(T)}{T\ell} \), in the equation above we get that 
\[ \frac{D_2}{D_1} = O\left(\frac{\log(T)(dn_0\zeta_0^2 + n_1\zeta_1^2)}{T\ell n} + \frac{\log(T)(dn_0\sigma_0^2 + n_1\sigma_1^2)}{T\ell n} + \frac{\log(T)dn_0}{T\ell n}\right). \]

We also have that \( D_1 \leq C_1 \leq \frac{4\zeta}{n} \), and 
\[ \frac{w_T}{S_T} = \frac{(1 - \frac{\eta\ell}{2n})^{-T}}{\sum_{t=1}^{T}(1 - \frac{\eta\ell}{2n})^{-t}} \geq (1 - \frac{\eta\ell}{2n})(1 - \frac{\eta\ell}{2n})^{-1} = \frac{\eta\ell}{2n}. \]

Hence, \( \frac{w_T}{S_T D_1} \geq \frac{\eta\ell}{2n} \). By putting together the above inequalities we get the final convergence bound:
\[ \mathbb{E}[f(y_T) - f(x^*)] + \frac{\ell\mathbb{E}\|\mu_T - x^*\|^2}{8} = \mathbb{E}[f(y_T) - f(x^*)] + \frac{\ell a_T^2}{8} \leq \mathbb{E}[f(y_T) - f(x^*)] + \frac{w_T a_T^2}{S_T D_1} + \frac{D_2}{D_1} \]
\[ = O\left(\frac{L\|\mu_0 - x^*\|^2}{T\log(T)} + \frac{\log(T)(dn_0\zeta_0^2 + n_1\zeta_1^2)}{T\ell n} + \frac{\log(T)(dn_0\sigma_0^2 + n_1\sigma_1^2)}{T\ell n} + \frac{\log(T)dn_0}{T\ell n}\right). \]

Finally, we need to gather all upper bounds on \( \eta \) and compute lower bound on \( T \) so that the upper bounds are satisfied. We need \( \eta = O\left(\frac{1}{n(L + \epsilon + 1)}\right), \eta = O\left(\frac{1}{n(L + \epsilon + 1)}\right) \) in the proof of this theorem, the lemmas we used require \( \eta \leq \frac{1}{10n} \) and 
\[ \eta \leq \frac{\sqrt{T\ell n}}{2\sqrt{Ldn_0(d + 3)^2}} = O\left(\frac{\ell}{Ld}\right). \]

Considering \( \ell \leq L \), we get that we need \( \eta = O\left(\frac{1}{(d + n)(L + 1)\zeta^2}\right) \). Thus, we need \( \frac{T}{\log(T)} = \Omega\left(\frac{n(d + n)(L + 1)\zeta^2}{\epsilon} \right) \).

\( \square \)