OPT2022: 14th Annual Workshop on Optimization for Machine Learning

Bidirectional Adaptive Communication for Heterogeneous Distributed Learning

Dmitrii Avdiukhin DAVDYUKH@IU.EDU
Indiana University, IN

Vladimir Braverman VOVA@CS.JHU.EDU
Rice University, TX

Nikita Ivkin IVKIN@AMAZON.COM
Amazon, NY

Sebastian U. Stich STICH@CISPA.DE
CISPA, Germany

Abstract

Communication is a key bottleneck in distributed optimization, and, in particular, bandwidth and latency can be limiting factors when devices are connected over commodity networks, such as in Federated Learning. State-of-the-art techniques tackle these challenges by advanced compression techniques or delaying communication rounds according to predefined schedules. We present a new scheme that adaptively skips communication (broadcast and client uploads) by detecting slow-varying updates. The scheme automatically adjusts the communication frequency independently for each worker and the server. By utilizing an error-feedback mechanism – borrowed from the compression literature – we prove that the convergence rate is the same as for batch gradient descent in the convex and nonconvex smooth cases. We show that the total number of communication rounds between server and clients needed to achieve a targeted accuracy is reduced, even in the case when the data distribution is highly non-IID.

1. Introduction

With the data moving to the edge devices, training large scale machine learning models unavoidably shifts towards the distributed settings [24]. More and more applications require large number of workers cooperating in training one shared machine learning model, with each worker typically holding its small share of data and has very limited network bandwidth. Often such settings are driven by privacy concerns, so the data should not leave the device, among examples are personal smartphones, geo-destributed devices storing medical data, sensor networks, etc [10, 57].

In this paper, we address the problem of data parallel stochastic optimization with a central node coordinating computation of stochastic gradients on N edge nodes (or workers/clients):

$$\min_{x\in\mathbb{R}^d} f(x) \quad \text{with} \quad f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)\,.$$

Here $i \in [N]$ denotes the worker identifier, $f_i(x)\colon \mathbb{R}^d \to \mathbb{R}$ the loss function with respect to the model state $x \in \mathbb{R}^d$ on the worker $i \in [N]$. We assume that we can only query stochastic gradients for each $f_i(x)$, that is, we have access to a stochastic oracle $\mathbf{g}_i(\mathbf{x})$ with $\mathbb{E}[\mathbf{g}_i(\mathbf{x})] = \nabla f_i(\mathbf{x})$.

In the parameter server model [15, 16] stochastic optimization is performed via the following iterative steps: (1) at time step t, each worker node computes a stochastic gradient $\mathbf{g}_i^{(t)}(\mathbf{x}^{(t)})$ on a local mini-batch of data and (2) each worker communicates the estimated gradient to the server, (3) the server performs a gradient update step, $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \frac{\alpha}{N}\sum_{i=1}^{N}\mathbf{g}_i^{(t)}$, where α denotes the learning rate, and finally, (4) the server broadcasts $\mathbf{x}^{(t+1)}$ to all workers.

With the number of workers growing and with additional network resources getting more and more expensive, the speed of communicating local stochastic gradients and broadcasting the updates became one of the main performance bottlenecks [3]. Major attempts to ease that communication bottleneck fall into two categories: compressing the messages and reducing the communication frequency.

A rich variety of compressors, including diverse sparsification [4, 23, 54] and quantization [3, 8] techniques, can drastically speedup the communication by dropping the message size from $O(d)$ down to $O(\log d)$ [3] or even $O(1)$ [4, 54] per worker. Nevertheless, all the workers and the server are still required to communicate at every iteration, which can be infeasible in the Federated Learning setting when number of workers is in millions, and each worker have very limited network access (i.e. smartphones, or sensor networks) [24]. Reducing the communication frequency can be achieved by either fixed communication delay (a.k.a. LocalSGD [51, 64]) or via more adaptive communication protocols [11, 48]. Both can be effectively incorporated into the error accumulation framework [53] to guarantee the convergence. Our method falls into the class of adaptive communication protocols. Note that it is very well orthogonal to message compressing methods, thus can be efficiently combined with it.

While several compression methods [43, 56] address the cost of broadcasting the updates (downlink), current approaches reducing the communication frequency, such as e.g. [48], often neglect the cost of downlink and focus primarily on the cost of sending the gradients (uplink). In this paper we present a new communication protocol PROCRASTINATOR which detects slow-varying updates and gradients and choose to adaptively skip communication rounds—both server broadcasts and worker uploads, thus effectively reducing the latency for both.

1.1. Related Works

For training machine learning models in data centers, parameter-servers are used to aggregate the updates from participating devices and to orchestrate training [15, 28]. In Federated Learning this server-based approach has been extend to train over decentralized data at larger scale [24, 41]. To alleviate communication bottlenecks for exchanging updates between devices, several techniques have been proposed:

- **(i) Compressing updates:** Computed gradients are compressed using vector sparsification—communicate only top k or random k coordinates of the gradient vector [2, 54], value quantization—efficiently encode every gradient value to use smaller number of bits [3, 8] and random projections [23, 47, 58]. The downside of these approaches is an inevitable loss of information during compression, which can potentially increase the number of iterations.
- **(ii) Local SGD** [51, 64]**:** Every client performs multiple steps using the local gradients, and after that the results are averaged. Unfortunately, for some iteration, not all workers may be available for computation or communication, and a practical approach is to use updates from the clients which communicated first. **Federated Averaging** [35, 40] mimics this behavior by subsampling the workers whose updates are used at the current iterations. However, these

approaches require client communication and server broadcast even when the communicated data didn't significantly change. See [39] for more discussion on the local SGD and its variants.

(iii) Sparse communication on decentralized topologies: While relying on a parameter server to aggregate updates [15] or communication-heavy all-reduce, in decentralized training methods clients exchange model updates in a peer-to-peer fashion. This can alleviate communication bottlenecks in data-center training [5] and can be applied to arbitrary network topologies [36, 55].

(iv) Delayed and Asynchronous Updates: Methods that can tolerate delayed updates are often used for optimization in data centers or over unreliable channels [14]. While skipping updates may cause a short-term reduction in communication, such methods cannot guarantee a reduction in communication over the entire phase of training (standard convergence proofs depend on the property that all updates are communicated eventually [1, 53]).

These techniques have been refined in follow up works and to some extent are orthogonal to each other, i.e. they can be applied on top of each other, see e.g. [7].

Most of the papers introducing the compression techniques only focus on compressing the uplink messages sent from the workers to the server, but do not compress donwlink broadcast messages from the server to the workers [3, 4, 18, 42, 46, 52–54, 61]. A few recent works study bi-directional compression [13, 38, 43, 44, 56, 62, 63], however these works do not consider event triggered communication. Decentralized techniques alleviate the broadcast by design and only exchange compressed messages [30, 31, 55].

Distributed methods that use only intermittent communication most frequently communicate after a prescribed number of iterations (or epochs) on the local data [37, 40, 59], it is also possible to maintain a constant frequency only in expectation [32], to increase the frequency during training [60] or decrease the frequency [20]. However, for these methods, the communication frequency has to be fixed in advance.

In contrast, event triggered schemes do not follow prescribed communication patterns, but trigger communication events based on local (or global) decision rules, taking the problem data and algorithm state into account. Event triggered communication has been considered in the control community [17, 21] and optimization community [11, 12, 22, 25, 29].

The foundational results on delayed SGD method in [1] were derived under the simplifying assumption that the data on the worker nodes follows the same distribution (IID setting). While this may be a valid assumption in distributed optimization on a server (with frequent data-shuffling among nodes [19] in the Federated Learning setting, where the data is distributed in a fixed partition over the clients, it is paramount to study the non-IID setting. Distributed optimization under non-IID assumptions is a significantly harder problem than the IID case (where in principle, workers could solve the problem independently of each other, without ever communicating with others). A further distinguishing feature that sets our work apart from the standard literature on delayed communication is that we do not need to assume a fixed upper bound on the maximal encountered delay, as opposed in [1, 50, 53]. Instead, the communication frequency is automatically controlled by the algorithm. Very recently, Aviv et al. [6] present an algorithm that automatically adapts to the delays, but this methods does not guarantee a reduction in communication.

The most closely related work is the LENA [48] framework, which was the first to introduce the combination of event-triggered communication and drifting. Drift can be seen as expected gradient step from a silent worker: if at the iteration t, worker i decided not to communicate, the server will

assume the gradient of that worker is equal to the predefined drift value. Drift value can be updated when the next communication from that worker happens. While different triggering rules and drift strategies can be designed, it still requires broadcasting updated model parameters to all the workers every iteration, i.e. downlink stays the same. Thus even if all workers communicate infinitely rare, broadcasts will stay the same and effective reduction in latency is at most twice. In this paper, we challenge this limitation by introducing server triggers and global update drift (can be seen as a server drift), which is conceptually symmetric to the client drift. Main challenge in reducing the downlink is caused by dissynchronisation of workers drifts. In section C we compare our approach to the LENA framework [48] experimentally.

1.2. Our contributions

Our main contribution is the framework PROCRASTINATOR (Algorithm 1) which allows both the server and the workers to control their communication, sending updates only when necessary. In a nutshell, each worker monitors the norm of an accumulated difference between the current gradient and the last communicated gradient, and delays communication until the norm of the accumulated difference passes a certain threshold. The server computes the average of the last communicated local gradients as an estimate of the average gradient which is broadcast to the workers in the similar pattern. Our framework allows one to control the communication frequency by specifying the appropriate thresholds. Regardless of the choices of the thresholds, we show that, with the appropriate step size, the algorithm converges to a local minimum:

Theorem 1 (Informal, see Theorem 5) *For a Lipschitz function f, when the stochastic variance and the deviation between local gradients are bounded, after T iterations of Algorithm 2 we have:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 = O\left(\frac{1}{\sqrt{NT}} + \frac{1}{T^{\frac{2}{3}}}\right).$$

1.3. Preliminaries

For a function $f\colon \mathbb{R}^n \to \mathbb{R}$, we consider the minimization problem $\min_{\mathbf{x}} f(\mathbf{x})$. We make the following standard assumptions [9]:

Assumption 1 *$f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth, i.e. for all $\mathbf{x}, \mathbf{y}$: $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le L\|\mathbf{x} - \mathbf{y}\|$.*

In distributed settings, we have N clients, where each client corresponds to its local function f_i such that $f(\mathbf{x}) = \operatorname{avg}_i f_i(\mathbf{x})$, where $\operatorname{avg}_i a_i = \frac{1}{N}\sum_{i=1}^{N} a_i$. Furthermore, for each client, we assume that we have access to the stochastic gradient oracle $\mathbf{g}_i$:

Assumption 2 *For every client $i \in [N]$, the stochastic gradient is unbiased and has bounded variance, i.e. $\mathbb{E}[\mathbf{g}_i(\mathbf{x}) \mid \mathbf{x}] = \nabla f_i(\mathbf{x})$, $\quad \mathbb{E}[\|\mathbf{g}_i(\mathbf{x}) - \nabla f_i(\mathbf{x})\|^2 \mid \mathbf{x}] \le \sigma^2$ for all $\mathbf{x}$.*

Finally, we bound the deviation of local gradients from the global gradient:

Assumption 3 *For every client $i \in [N]$, we have $\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2 \le \alpha\|\nabla f(\mathbf{x})\|^2 + \beta$.*

Algorithm 1: Procrastinator (for more detailed pseudocode refer to Appendix B)

parameters: step size γ, number of iterations T, trigger parameters (A, B)

input: Initial point $\mathbf{x}^{(0)}$, stochastic gradient oracles $\mathbf{g}_i$

// drift and error accumulators for server and clients

$\mathbf{u}^{(0)} \leftarrow \mathbf{0}$, $\mathbf{r}^{(0)} \leftarrow \mathbf{0}$, $\forall i : \mathbf{d}_i^{(0)} \leftarrow \mathbf{0}$, $\mathbf{e}_i^{(0)} \leftarrow \mathbf{0}$

for $t = 0, 1, 2, \ldots, T-1$ **do**

For each client $i \in [N]$**:**

Compute local stochastic gradient $\mathbf{g}_i^{(t)}$

// Update local error: + gradient - local drift

$\mathbf{e}_i^{(t+1)} \leftarrow \mathbf{e}_i^{(t)} + \mathbf{g}_i^{(t)} - \mathbf{d}_i^{(t)}$

$\mathbf{d}_i^{(t+1)} \leftarrow \mathbf{d}_i^{(t)}$

// Check if the local error is large

if $\|\mathbf{e}_i^{(t+1)}\|^2 \geq A\|\mathbf{g}_i^{(t)}\|^2 + B$ **then**

// New local drift estimate

$\mathbf{d}_i^{(t+1)} = \mathbf{g}_i^{(t)}$

Send $(\mathbf{e}_i^{(t+1)}, \mathbf{d}_i^{(t+1)})$ to the server

$\mathbf{e}_i^{(t+1)} \leftarrow 0$

if *Didn't* ***receive*** *updated* $(\mathbf{x}^{(t+1)}, \mathbf{u}^{(t+1)})$ *from server* **then**

// Use $\mathbf{u}^{(t)}$ for local step

$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \gamma\mathbf{u}^{(t)}, \quad \mathbf{u}^{(t+1)} \leftarrow \mathbf{u}^{(t)}$

Server Update:

Let $\mathcal{C}^{(t)} \subset [N]$ be the set of clients that send updated $(\mathbf{e}_i^{(t+1)}, \mathbf{d}_i^{(t+1)})$ at iteration t

for $i \in \mathcal{C}^{(t)}$ **do**

Receive $(\mathbf{e}_i^{(t+1)}, \mathbf{d}_i^{(t+1)})$ from client i

for $i \notin \mathcal{C}^{(t)}$ **do** $\mathbf{d}_i^{(t+1)} \leftarrow \mathbf{d}_i^{(t)}$;

$\mathbf{r}^{(t+1)} \leftarrow \mathbf{r}^{(t)} + \frac{1}{N}\sum_{i=1}^{N}(\mathbf{d}_i^{(t)} - \mathbf{u}^{(t)}) + \frac{1}{N}\sum_{i\in\mathcal{C}^{(t)}} \mathbf{e}_i^{(t+1)}$

// Update server error: + local step estimates - global step + communicated errors

$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \gamma\mathbf{u}^{(t)}, \quad \mathbf{u}^{(t+1)} \leftarrow \mathbf{u}^{(t)}$

// Check if the server error is large

if $\|\mathbf{r}^{(t+1)}\|^2 \geq A\|\mathrm{avg}_i(\mathbf{d}_i^{(t)})\|^2 + B$ **then**

// Average drift is local update at next iterations

$\mathbf{u}^{(t+1)} \leftarrow \frac{1}{N}\sum_{i=1}^{N} \mathbf{d}_i^{(t+1)}$

// Propagate server error

$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \gamma\mathbf{u}^{(t)} - \gamma\mathbf{r}^{(t+1)}$

Broadcast $(\mathbf{x}^{(t+1)}, \mathbf{u}^{(t+1)})$

$\mathbf{r}^{(t+1)} \leftarrow 0$

2. Algorithm and Analysis

In distributed SGD, at every iteration, each client computes local gradient and communicates it to the server, which broadcasts the average gradient to all clients. After that, all clients perform a gradient descent step using this average. This approach requires each client and server to communicate at every iteration, which leads to extensive communication. The intuition behind our algorithm is to maintain estimates—of local gradient on the server and of average gradient on the clients—and communicate the actual values only when they significantly deviate from the estimates.

Our approach, PROCRASTINATOR, is presented as Algorithm 1. The server uses $\mathbf{d}_i^{(t)}$ to estimate local gradients (Lines 6 and 7) and clients use $\mathbf{u}^{(t)}$ to estimate the average gradient. Since $\mathbf{d}_i^{(t)}$ are the only estimates of local gradients available to the server, we compute $\mathbf{u}^{(t)}$ as an average of $\mathbf{d}_i^{(t)}$ (Line 17). Since $\mathbf{u}^{(t)}$ is the best estimate of the average gradient available to clients, the clients perform update $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \gamma \mathbf{u}^{(t)}$ instead of a gradient descent step (Line 10).

The key idea of our algorithm is to use *triggers* to control communication. Both client and server triggers have the following structure: they maintain an "error", which accumulates deviation of the actual value from the current estimate (Lines 3 and 14). When the accumulated error passes a certain threshold (Lines 5 and 16), a new estimate is communicated (Lines 7 and 19) and the error is reset. Note that, while the difference between the actual value and its estimate can be small, accumulated throughout multiple iterations it can substantially alter the algorithm behavior. To address this problem, the errors accumulate these differences since the last trigger activation.

2.1. Convergence Analysis

The main idea behind our analysis of Algorithm 2 is to use error-feedback framework [27, 45, 53, 54]. We introduce the sequence of *corrected iterates* $\{\mathbf{y}^{(t)}\}$ where $\mathbf{y}^{(0)} = \mathbf{x}^{(0)}$ and $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} - \gamma \operatorname{avg}_i \mathbf{g}_i^{(t)}$. Unlike $\{\mathbf{x}^{(t)}\}$, the sequence uses gradients for updates, and such a sequence commonly used in the analysis of SGD with error-feedback. The sequence $\{\mathbf{y}^{(t)}\}$ has the following relation with $\{\mathbf{x}^{(t)}\}$:

Lemma 2 *For any t, $\mathbf{y}^{(t)} = \mathbf{x}^{(t)} - \gamma(\mathbf{r}^{(t)} + \operatorname{avg}_i \mathbf{e}_i^{(t)})$.*

Therefore, $\xi^{(t)} = \mathbf{r}^{(t)} + \operatorname{avg}_i \mathbf{e}_i^{(t)}$ is the full error. Based on the proof of [27, Theorem II], we have the following intermediate result:

Lemma 3 *Let $\xi^{(t)} = \mathbf{r}^{(t)} + \operatorname{avg}_i \mathbf{e}_i^{(t)}$. Then under Assumptions 1 and 2, for every T we have*

$$\mathbb{E}[f(\mathbf{y}^{(T)})] \leq f(\mathbf{y}^{(0)}) - \sum_{t=0}^{T-1} \frac{\gamma}{2}(1 - L\gamma)\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + T\frac{L\gamma^2\sigma^2}{2N} + \frac{L^2\gamma^3}{2}\sum_{t=0}^{T-1}\mathbb{E}\|\xi^{(t)}\|^2 .$$

It remains to bound $\sum_{t=0}^{T-1}\mathbb{E}\|\xi^{(t)}\|^2$. The following lemma shows that it can be expressed in terms of $\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2$:

Lemma 4 *Under Assumption 3, for every T we have:*

$$\sum_{t=0}^{T-1}\mathbb{E}\|\xi^{(t)}\|^2 = O\Big(\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + T\Big) .$$

The proof of this lemma crucially exploits the main property of the Algorithm 1: the choice of our trigger condition allows to ensure that the accumulated error always remains bounded.

Selecting $\gamma^2 \leq \frac{1}{4c_1L^2}$ (and therefore $\gamma \leq \frac{1}{4L}$), from Lemma 3 we have:

$$\mathbb{E}[f(\mathbf{y}^{(T)})] \leq f(\mathbf{y}^{(0)}) - \frac{\gamma}{4}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + T\frac{L\gamma^2\sigma^2}{2N} + T\frac{L^2\gamma^3}{2}\,.$$

After regrouping the terms and using that $f(\mathbf{y}^{(0)}) - \mathbb{E}[f(\mathbf{y}^{(T)})] \leq f_{\max}$:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 = O(\frac{f_{\max}}{T\gamma} + \frac{L\gamma\sigma^2}{N} + 2L^2\gamma^2)\,.$$

Selecting the step size based on [32, Lemma 17], we have:

Theorem 5 *Under Assumptions 1–3, after T iterations of Algorithm 2, we have:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 = O\left(\frac{1}{\sqrt{NT}} + \frac{1}{T^{\frac{2}{3}}}\right)\,,$$

which matches the convergence rate $O(\frac{1}{\sqrt{NT}})$ of distributed SGD when $N = O(\sqrt[3]{T})$ or when the number of iterations $T = \Omega(N^3)$ is sufficiently large.

In the appendix, we present an additional convergence result for the case when the objective is convex.

2.2. Experiments and Hyperparameter Selection

Our experimental results in Appendix C show that PROCRASTINATOR outperforms local/distributed SGD with respect to communication. Compared with LENA, it shows similar client communication requirements, while requiring substantially less broadcasts.

The choice of parameter B in the trigger condition is primarily motivated by the desired communication gap. For a communication gap C, the expected accumulated stochastic variance after C iterations is $C \cdot \sigma^2$, and therefore setting B to $C \cdot \sigma^2$ offsets the variance.

Parameter A is important when gradients can be unbounded, e.g. in case of a strongly convex function with a starting point being far from optimum. A reasonable choice of A is 1, meaning that the client/server communicates when the corresponding error is comparable to the gradient norm. In all our experiments, we use $A = 1$ and $B = 10$, showing that the same choice of hyperparameters is applicable across various tasks.

3. Conclusion

We present a new method—PROCRASTINATOR—to address the communication bottlenecks and latency in distributed optimization. As a distinguishing novelty, our scheme can automatically suppress broadcast of model updates to clients if the updated model state on the server does not deviate much from predicted values. If the clients do not receive a broadcast, they update their state according to a predefined rule which ensures that clients stay in sync—avoiding client drift [24, 26]. This enables drastic savings in the total number of broadcasts, but also client uploads.

References

[1] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 873–881. Curran Associates, Inc., 2011.

[2] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445, Copenhagen, Denmark, 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/D17-1045.

[3] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

[4] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cedric Renggli. The convergence of sparsified gradient methods. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5977–5987. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7837-the-convergence-of-sparsified-gradient-methods.pdf.

[5] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

[6] Rotem Zamir Aviv, Ido Hakimi, Assaf Schuster, and Kfir Yehuda Levy. Asynchronous distributed learning : Adapting to gradient delays without prior knowledge. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 436–445. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/aviv21a.html.

[7] Debraj Basu, Deepesh Data, Can Karakus, and Suhas N. Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification, and local computations. *IEEE J. Sel. Areas Inf. Theory*, 1(1):217–226, 2020. doi: 10.1109/jsait.2020.2985917. URL https://doi.org/10.1109/jsait.2020.2985917.

[8] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018.

[9] L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. URL https://doi.org/10.1137/16M1080173.

[10] Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

[11] Tianyi Chen, Georgios Giannakis, Tao Sun, and Wotao Yin. LAG: Lazily aggregated gradient for communication-efficient distributed learning. In *Advances in Neural Information Processing Systems*, pages 5050–5060, 2018.

[12] Weisheng Chen and Wei Ren. Event-triggered zero-gradient-sum distributed consensus optimization over directed networks. *Automatica*, 65:90–97, 2016. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2015.11.015. URL https://www.sciencedirect.com/science/article/pii/S0005109815004793.

[13] Laurent Condat and Peter Richtárik. Murana: A generic framework for stochastic variance-reduced optimization. In *Mathematical and Scientific Machine Learning*, pages 81–96. PMLR, 2022.

[14] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc D'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc V. Le, and Andrew Y. Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems 25 (NeurIPS)*, pages 1223–1231. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf.

[15] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc V. Le, and Andrew Y. Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.

[16] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *J. Mach. Learn. Res.*, 13(1):165–202, January 2012. URL http://dl.acm.org/citation.cfm?id=2503308.2188391.

[17] Dimos V. Dimarogonas, Emilio Frazzoli, and Karl H. Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, 57(5):1291–1297, 2012. doi: 10.1109/TAC.2011.2174666.

[18] Eduard Gorbunov, Dmitry Kovalev, Dmitry Makarenko, and Peter Richtárik. Linearly converging error compensated SGD. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020. URL https://arxiv.org/abs/2010.12292.

[19] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[20] Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck R. Cadambe. Local SGD with periodic averaging: Tighter analysis and adaptive synchronization. In *Advances in Neural Information Processing Systems*, 2019.

[21] W. Heemels, K. H. Johansson, and Paulo Tabuada. An introduction to event triggered and self-triggered control. In *IEEE Conference on Decision and Contro*, pages 3270–3285, 2012.

[22] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. Gaia: Geo-distributed machine learning approaching LAN speeds. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 629–647, 2017.

[23] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-efficient distributed SGD with sketching. *arXiv preprint arXiv:1903.04488*, 2019.

[24] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[25] Michael Kamp, Mario Boley, Michael Mock, Daniel Keren, Assaf Schuster, and Izchak Sharfman. Adaptive communication bounds for distributed online learning. *arXiv preprint arXiv:1911.12896*, 2019.

[26] Sai P. Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda T. Suresh. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. In *ICML - Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020. URL https://arxiv.org/abs/1910.06378.

[27] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3252–3261. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/karimireddy19a.html.

[28] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

[29] Solmaz S. Kia, Jorge Cortés, and Sonia Martínez. Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication. *Automatica*, 55:254–264, 2015. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2015.03.001. URL https://www.sciencedirect.com/science/article/pii/S0005109815001053.

[30] Anastasia Koloskova, Sebastian U. Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3478–3487. PMLR, 2019. URL https://arxiv.org/abs/1902.00340.

[31] Anastasia Koloskova, Tao Lin, Sebastian U. Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1907.09356.

[32] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U. Stich. A unified theory of decentralized SGD with changing topology and local updates. *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL http://arxiv.org/abs/1602.05629.

[33] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[35] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[36] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems 30*, pages 5330–5340. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7117-can-decentralized-algorithms-outperform-centralized-algorithms-a-case-s pdf.

[37] Tao Lin, Sebastian U. Stich, Kumar K. Patel, and Martin Jaggi. Don't use large mini-batches, use local SGD. *International Conference on Learning Representations (ICLR)*, 2020. URL https://arxiv.org/abs/1808.07217.

[38] Xiaorui Liu, Yao Li, Jiliang Tang, and Ming Yan. A double residual compression algorithm for efficient distributed learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 133–143. PMLR, 2020. URL http://proceedings.mlr.press/v108/liu20a.html.

[39] Grigory Malinovsky, Kai Yi, and Peter Richtárik. Variance reduced proxskip: Algorithm, theory and application to federated learning. *arXiv preprint arXiv:2207.04338*, 2022.

[40] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[41] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2016. URL http://arxiv.org/abs/1602.05629.

[42] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

[43] Constantin Philippenko and Aymeric Dieuleveut. Bidirectional compression in heterogeneous settings for distributed or federated learning with partial participation: tight convergence guarantees. *arXiv preprint arXiv:2006.14591*, 2020.

[44] Constantin Philippenko and Aymeric Dieuleveut. Preserved central model for faster bidirectional compression in distributed settings. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2021.

[45] Peter Richtárik, Igor Sokolov, and Ilyas Fatkhullin. Ef21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems*, 34: 4384–4396, 2021.

[46] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. FetchSGD: Communication-efficient federated learning with sketching. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8253–8265. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/rothchild20a.html.

[47] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pages 8253–8265. PMLR, 2020.

[48] Hossein Shokri Ghadikolaei, Sebastian Stich, and Martin Jaggi. LENA: Communication-efficient distributed learning with self-triggered gradient uploads. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3943–3951. PMLR, 2021. URL http://proceedings.mlr.press/v130/shokri-ghadikolaei21a.html.

[49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[50] Sebastian Stich, Amirkeivan Mohtashami, and Martin Jaggi. Critical parameters for scalable distributed learning with large batches and asynchronous updates. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pages 4042–4050. PMLR, 2021.

[51] Sebastian U. Stich. Local SGD converges fast and communicates little. *International Conference on Learning Representations*, art. arXiv:1805.09767, 2019. URL https://arxiv.org/abs/1805.09767.

[52] Sebastian U. Stich. On communication compression for distributed optimization on heterogeneous data. *arXiv preprint arXiv:2009.02388*, 2020.

[53] Sebastian U. Stich and Sai Praneeth Karimireddy. The error-feedback framework: SGD with delayed gradients. *Journal of Machine Learning Research*, 21(237):1–36, 2020. URL http://jmlr.org/papers/v21/19-748.html.

[54] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems 31*, pages 4447–4458. Curran Associates, Inc., 2018.

[55] Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. Communication compression for decentralized training. In *Advances in Neural Information Processing Systems 31*, pages 7663–7673. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7992-communication-compression-for-decentralized-training.pdf.

[56] Hanlin Tang, Xiangru Lian, Tong Zhang, and Ji Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1905.05957*, 2019.

[57] Mark Tomlinson, Wesley Solomon, Yages Singh, Tanya Doherty, Mickey Chopra, Petrida Ijumba, Alexander C Tsai, and Debra Jackson. The use of mobile phones as a data collection tool: a report from a household survey in south africa. *BMC medical informatics and decision making*, 9(1):51, 2009.

[58] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. *arXiv preprint arXiv:1905.13727*, 2019.

[59] Jianyu Wang and Gauri Joshi. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. *arXiv preprint arXiv:1808.07576*, 2018. URL http://arxiv.org/abs/1808.07576.

[60] Jianyu Wang and Gauri Joshi. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD. In A. Talwalkar, V. Smith, and M. Zaharia, editors, *Proceedings of Machine Learning and Systems*, volume 1, pages 212–229, 2019. URL https://proceedings.mlsys.org/paper/2019/file/c8ffe9a587b126f152ed3d89a146b445-Paper.pdf.

[61] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized SGD and its applications to large-scale distributed optimization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5325–5333. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/wu18d.html.

[62] Yue Yu, Jiaxiang Wu, and Longbo Huang. Double quantization for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[63] Shuai Zheng, Ziyue Huang, and James T. Kwok. Communication-efficient distributed blockwise momentum SGD with error-feedback. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[64] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 2595–2603, 2010.

Appendix A. Algorithm

Algorithm 2 shows the complete version of the algorithm.

Appendix B. Convergence Proof

B.1. Non-convex Case

Recall that $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} - \gamma \operatorname{avg}_i(\mathbf{g}_i^{(t)})$. From Algorithm 2:

$$\mathbf{x}^{(t+1)} = \begin{cases} \mathbf{x}^{(t)} - \gamma \mathbf{u}^{(t)}, & \text{if broadcast doesn't happen at iteration } t \\ \mathbf{x}^{(t)} - \gamma \mathbf{u}^{(t)} - \gamma \mathbf{r}^{(t+1/2)}, & \text{if broadcast happens at iteration } t \end{cases}$$

We first show the relation between $\mathbf{x}^{(t)}$ and $\mathbf{y}^{(t)}$.

Lemma 6 *For any t, $\mathbf{y}^{(t)} = \mathbf{x}^{(t)} - \gamma(\mathbf{r}^{(t)} + \operatorname{avg}_i \mathbf{e}_i^{(t)})$.*

Proof Proof by induction. The equality holds for $t = 0$. Assume that for some t, $\mathbf{y}^{(t)} = \mathbf{x}^{(t)} - \gamma(\mathbf{r}^{(t)} + \operatorname{avg}_i \mathbf{e}_i^{(t)})$. Recall that $\mathcal{C}^{(t)}$ is the set of clients communicating at iteration t. By definition of $\mathbf{r}^{(t+1/2)}$ and $\mathbf{e}_i^{(t+1)}$, we have:

$$\begin{aligned}
&\mathbf{r}^{(t+1/2)} + \operatorname*{avg}_i(\mathbf{e}_i^{(t+1)}) \\
&= \left(\mathbf{r}^{(t)} + \operatorname*{avg}_i(\mathbf{d}_i^{(t)} - \mathbf{u}^{(t)}) + \frac{1}{N} \sum_{i \in \mathcal{C}^{(t)}} \mathbf{e}_i^{(t+1/2)} \right) + \frac{1}{N} \sum_{i \notin \mathcal{C}^{(t)}} \mathbf{e}_i^{(t+1/2)} && \text{(Def. of } \mathbf{r}^{(t+1/2)} \text{ and } \mathbf{e}_i^{(t+1)}) \\
&= \mathbf{r}^{(t)} + \operatorname*{avg}_i(\mathbf{d}_i^{(t)} - \mathbf{u}^{(t)}) + \operatorname*{avg}_i(\mathbf{e}_i^{(t+1/2)}) \\
&= \mathbf{r}^{(t)} + \operatorname*{avg}_i(\mathbf{d}_i^{(t)} - \mathbf{u}^{(t)}) + \operatorname*{avg}_i(\mathbf{e}_i^{(t)}) + \operatorname*{avg}_i(\mathbf{g}_i^{(t)} - \mathbf{d}_i^{(t)}) && \text{(Def. of } \mathbf{e}_i^{(t+1/2)}) \\
&= \mathbf{r}^{(t)} + \operatorname*{avg}_i(\mathbf{e}_i^{(t)}) + \operatorname*{avg}_i(\mathbf{g}_i^{(t)} - \mathbf{u}^{(t)}) && (1)
\end{aligned}$$

If the broadcast doesn't happen at iteration t, then:

$$\begin{aligned}
\mathbf{y}^{(t+1)} - \mathbf{x}^{(t+1)} &= (\mathbf{y}^{(t)} - \mathbf{x}^{(t)}) - \gamma \operatorname*{avg}_i(\mathbf{g}_i^{(t)} - \mathbf{u}^{(t)}) \\
&= -\gamma(\mathbf{r}^{(t)} + \operatorname*{avg}_i(\mathbf{e}_i^{(t)}) + \operatorname*{avg}_i(\mathbf{g}_i^{(t)} - \mathbf{u}^{(t)})) && \text{(IH)} \\
&= -\gamma(\mathbf{r}^{(t+1)} + \operatorname*{avg}_i(\mathbf{e}_i^{(t+1)})) && \text{(Equation (1))}
\end{aligned}$$

If the broadcast happens, we have:

$$\begin{aligned}
\mathbf{y}^{(t+1)} - \mathbf{x}^{(t+1)} &= (\mathbf{y}^{(t)} - \mathbf{x}^{(t)}) - \gamma \operatorname*{avg}_i(\mathbf{g}_i^{(t)} - \mathbf{u}^{(t)} - \mathbf{r}^{(t+1/2)}) \\
&= -\gamma(\mathbf{r}^{(t)} + \operatorname*{avg}_i(\mathbf{e}_i^{(t)}) + \operatorname*{avg}_i(\mathbf{g}_i^{(t)} - \mathbf{u}^{(t)}) - \mathbf{r}^{(t+1/2)}) && \text{(IH)} \\
&= -\gamma(\mathbf{r}^{(t+1/2)} + \operatorname*{avg}_i(\mathbf{e}_i^{(t+1)}) - \mathbf{r}^{(t+1/2)}) && \text{(Equation (1))} \\
&= -\gamma(\mathbf{r}^{(t+1)} + \operatorname*{avg}_i(\mathbf{e}_i^{(t+1)})) && (\mathbf{r}^{(t+1)} = 0) \qquad \blacksquare
\end{aligned}$$

Algorithm 2: Procrastinator (complete version)

parameters: step size γ, number of iterations T, client trigger parameters (A, B), server trigger parameters (C, D)

input: Initial point $\mathbf{x}^{(0)}$, stochastic gradient oracles $\mathbf{g}_i$

$\mathbf{r}^{(0)} \leftarrow \mathbf{0}$ // Server error

Broadcast $\mathbf{x}^{(0)}$ to all clients

for *each client* $i \in [N]$ **do**

  $\mathbf{d}_i^{(0)} \leftarrow \mathbf{0}$ // client drift — server's estimation of client's gradient

  $\mathbf{u}^{(0)} \leftarrow \mathbf{0}$ // server drift during last broadcast — client's estimation of the global update

  $\mathbf{e}_i^{(0)} \leftarrow \mathbf{0}$ // Local error

for $t = 0, 1, 2, \ldots$ **do**

For each client $i \in [N]$**:**

  $\mathbf{g}_i^{(t)} \leftarrow \nabla F_i(\mathbf{x}^{(t)})$ // Compute local stochastic gradient

  Client trigger:

  $\mathbf{e}_i^{(t+1/2)} \leftarrow \mathbf{e}_i^{(t)} + \mathbf{g}_i^{(t)} - \mathbf{d}_i^{(t)}$ // Update local error: + gradient - local drift

  if $\|\mathbf{e}_i^{(t+1/2)}\|^2 \geq A\|\mathbf{g}_i^{(t)}\|^2 + B$ **then** // Check if the local error is large

    $\mathbf{d}_i^{(t+1)} = \mathbf{g}_i^{(t)}$ // New local drift estimate

    Send $(\mathbf{e}_i^{(t+1/2)}, \mathbf{d}_i^{(t+1)})$ to the server

    $\mathbf{e}_i^{(t+1)} \leftarrow 0$

  else

    $\mathbf{d}_i^{(t+1)} \leftarrow \mathbf{d}_i^{(t)}$

    $\mathbf{e}_i^{(t+1)} \leftarrow \mathbf{e}_i^{(t+1/2)}$

  if *Didn't* ***receive*** *updated* $(\mathbf{x}^{(t+1)}, \mathbf{u}^{(t+1)})$ *from server* **then**

    $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \gamma\mathbf{u}^{(t)}, \quad \mathbf{u}^{(t+1)} \leftarrow \mathbf{u}^{(t)}$ // Use $\mathbf{u}^{(t)}$ for local step

Server Update:

Let $\mathcal{C}^{(t)} \subset [N]$ be the set of clients that send updated $(\mathbf{e}_i^{(t+1/2)}, \mathbf{d}_i^{(t+1)})$ at iteration t

for $i \in \mathcal{C}^{(t)}$ **do Receive** $(\mathbf{e}_i^{(t+1/2)}, \mathbf{d}_i^{(t+1)})$ from client i ;

for $i \notin \mathcal{C}^{(t)}$ **do** $\mathbf{d}_i^{(t+1)} \leftarrow \mathbf{d}_i^{(t)}$;

Server trigger:

$\mathbf{r}^{(t+1/2)} \leftarrow \mathbf{r}^{(t)} + \frac{1}{N}\sum_{i=1}^N(\mathbf{d}_i^{(t)} - \mathbf{u}^{(t)}) + \frac{1}{N}\sum_{i\in\mathcal{C}^{(t)}}\mathbf{e}_i^{(t+1/2)}$ // Update server error: + local step estimates - global step + communicated errors

if $\|\mathbf{r}^{(t+1/2)}\|^2 \geq C\|\mathrm{avg}_i(\mathbf{d}_i^{(t)})\|^2 + D$ **then** // Check if the server error is large

  $\mathbf{u}^{(t+1)} \leftarrow \frac{1}{N}\sum_{i=1}^N \mathbf{d}_i^{(t+1)}$ // Average drift is local update at next iterations

  $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \gamma\mathbf{u}^{(t)} - \gamma\mathbf{r}^{(t+1/2)}$ // Propagate server error

  Broadcast $(\mathbf{x}^{(t+1)}, \mathbf{u}^{(t+1)})$

  $\mathbf{r}^{(t+1)} \leftarrow 0$

else

  $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \gamma\mathbf{u}^{(t)}, \quad \mathbf{u}^{(t+1)} \leftarrow \mathbf{u}^{(t)}$ // Same as clients

  $\mathbf{r}^{(t+1)} \leftarrow \mathbf{r}^{(t+1/2)}$

Lemma 7 *Let $\xi^{(t)} = \mathbf{r}^{(t)} + \operatorname{avg}_i \mathbf{e}_i^{(t)}$. Then for every T we have*

$$\mathbb{E}[f(\mathbf{y}^{(T)})] \leq f(\mathbf{y}^{(0)}) - \sum_{t=0}^{T-1} \frac{\gamma}{2}(1 - L\gamma)\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + T\frac{L\gamma^2\sigma^2}{2N} + \frac{L^2\gamma^3}{2}\sum_{t=0}^{T-1}\mathbb{E}\|\xi^{(t)}\|^2$$

Proof By the Descent Lemma:

$$\begin{aligned}
&\mathbb{E}_t[f(\mathbf{y}^{(t+1)})] \\
&\leq f(\mathbf{y}^{(t)}) - \langle \nabla f(\mathbf{y}^{(t)}), \mathbb{E}_t[\mathbf{y}^{(t+1)} - \mathbf{y}^{(t)}]\rangle + \frac{L}{2}\mathbb{E}_t\|\mathbf{y}^{(t+1)} - \mathbf{y}^{(t)}\|^2 \\
&= f(\mathbf{y}^{(t)}) - \gamma\langle \nabla f(\mathbf{y}^{(t)}), \mathbb{E}_t[\mathbf{g}^{(t)}]\rangle + \frac{L\gamma^2}{2}\mathbb{E}_t\|\mathbf{g}^{(t)}\|^2 \\
&= f(\mathbf{y}^{(t)}) - \gamma\langle \nabla f(\mathbf{y}^{(t)}), \nabla f(\mathbf{x}^{(t)})\rangle + \frac{L\gamma^2}{2}(\|\nabla f(\mathbf{x}^{(t)})\|^2 + \mathbb{E}_t\|\frac{1}{N}\sum_{i=1}^{N}(\mathbf{g}_i^{(t)} - \nabla f_i(\mathbf{x}^{(t)}))\|^2) \\
&= f(\mathbf{y}^{(t)}) - \gamma\langle \nabla f(\mathbf{y}^{(t)}), \nabla f(\mathbf{x}^{(t)})\rangle + \frac{L\gamma^2}{2}(\|\nabla f(\mathbf{x}^{(t)})\|^2 + \frac{\sigma^2}{N}),
\end{aligned}$$

where we used the fact that $\mathbb{E}_t[\mathbf{g}^{(t)}] = \nabla f(\mathbf{x}^{(t)})$ stochastic noises are independent. Since $\langle \nabla f(\mathbf{x}^{(t)}), \nabla f(\mathbf{x}^{(t)})\rangle = \|\nabla f(\mathbf{x}^{(t)})\|^2$ and using inequality $\langle a, b\rangle \leq \frac{1}{2}\|a\|^2 + \frac{1}{2}\|b\|^2$:

$$\begin{aligned}
&\mathbb{E}_t[f(\mathbf{y}^{(t+1)})] \\
&\leq f(\mathbf{y}^{(t)}) - \gamma\|\nabla f(\mathbf{x}^{(t)})\|^2 + \frac{L\gamma^2}{2}(\|\nabla f(\mathbf{x}^{(t)})\|^2 + \frac{\sigma^2}{N}) + \gamma\langle \nabla f(\mathbf{x}^{(t)}) - \nabla f(\mathbf{y}^{(t)}), \nabla f(\mathbf{x}^{(t)})\rangle \\
&\leq f(\mathbf{y}^{(t)}) - \gamma(1 - \frac{L\gamma}{2})\|\nabla f(\mathbf{x}^{(t)})\|^2 + \frac{L\gamma^2\sigma^2}{2N} + \frac{\gamma}{2}\|\nabla f(\mathbf{x}^{(t)}) - \nabla f(\mathbf{y}^{(t)})\|^2 + \frac{\gamma}{2}\nabla\|f(\mathbf{x}^{(t)})\|^2 \\
&\leq f(\mathbf{y}^{(t)}) - \frac{\gamma}{2}(1 - L\gamma)\|\nabla f(\mathbf{x}^{(t)})\|^2 + \frac{L\gamma^2\sigma^2}{2N} + \frac{L^2\gamma}{2}\|\mathbf{x}^{(t)} - \mathbf{y}^{(t)}\|^2 \\
&= f(\mathbf{y}^{(t)}) - \frac{\gamma}{2}(1 - L\gamma)\|\nabla f(\mathbf{x}^{(t)})\|^2 + \frac{L\gamma^2\sigma^2}{2N} + \frac{L^2\gamma^3}{2}\|\xi^{(t)}\|^2
\end{aligned}$$

Taking the expectation:

$$\mathbb{E}[f(\mathbf{y}^{(t)})] \leq f(\mathbf{y}^{(0)}) - \sum_{\tau=0}^{t-1} \frac{\gamma}{2}(1 - L\gamma)\mathbb{E}\|\nabla f(\mathbf{x}^{(\tau)})\|^2 + t\frac{L\gamma^2\sigma^2}{2N} + \frac{L^2\gamma^3}{2}\sum_{\tau=0}^{t-1}\mathbb{E}\|\xi^{(\tau)}\|^2 .$$

■

It suffices to bound $\sum_{\tau=0}^{t-1}\mathbb{E}\|\xi^{(\tau)}\|^2$.

Lemma 8 *Under Assumption 3, for every T we have:*

$$\sum_{t=0}^{T-1}\mathbb{E}\|\xi^{(t)}\|^2 \leq c_1\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + c_2 T,$$

where $c_1 = 8(A + AC + C)(1 + \alpha)$ and $c_2 = 4((A + AC + C)(\beta + \sigma^2) + BC + D)T$.

Proof Since $\|\xi^{(t)}\|^2 \le 2(\|\mathbf{r}^{(t)}\|^2 + \|\operatorname{avg}_i \mathbf{e}_i^{(t)}\|^2)$, we bound $\sum_t \mathbb{E}\|\mathbf{e}_i^{(t)}\|^2$ and $\sum_t \mathbb{E}\|\mathbf{r}^{(t)}\|^2$ separately.

Bounding $\sum_t \mathbb{E}\|\mathbf{e}_i^{(t)}\|^2$. If the client communicates to the server at iteration t, then $\mathbf{e}^{(t+1)} = 0$. Otherwise, $\mathbf{e}^{(t+1)} = \mathbf{e}^{(t+1/2)}$, and by Line 8 of Algorithm 2, we have:

$$\mathbb{E}\|\mathbf{e}_i^{(t+1)}\|^2 \le A\mathbb{E}\|\mathbf{g}_i^{(t)}\|^2 + B$$

and therefore

$$\sum_{t=0}^{T-1} \mathbb{E}\|\mathbf{e}_i^{(t)}\|^2 \le A\sum_{t=0}^{T-1} \mathbb{E}\|\mathbf{g}_i^{(t)}\|^2 + BT$$

Bounding $\sum_t \mathbb{E}\|\mathbf{r}^{(t)}\|^2$. From Line 19 of Algorithm 2, we have:

$$\sum_{t=0}^{T-1} \|\mathbf{r}^{(t)}\|^2 \le C\sum_{t=0}^{T-1} \|\operatorname*{avg}_i(\mathbf{d}_i^{(t)})\|^2 + TD \le C\sum_{t=0}^{T-1} \operatorname*{avg}_i \|\mathbf{d}_i^{(t)}\|^2 + TD = C\operatorname*{avg}_i \sum_{t=0}^{T-1} \|\mathbf{d}_i^{(t)}\|^2 + TD,$$

and therefore it suffices to bound $\sum_{t=0}^{T-1} \mathbb{E}\|\mathbf{d}_i^{(t)}\|^2$ for all i. For a client i, let t_1 and t_2 be the iterations such that the client communicated at times t_1 and t_2 and didn't communicate for any $t \in (t_1, t_2)$ ($t_1 = -1$ if t_2 is the earliest communication, $t_2 = T-1$ if t_1 is the latest communication). If $t_2 = t_1 + 1$, then $\sum_{t=t_1+1}^{t_2-1} \|\mathbf{d}_i^{(t)}\|^2$ is trivially 0. Otherwise, since the trigger didn't activate at iteration $(t_2 - 1)$, by Line 8 of Algorithm 2 we have

$$\|\mathbf{e}_i^{(t_2)}\|^2 = \|\mathbf{e}_i^{(t_2-1+1/2)}\|^2 = \|\mathbf{e}_i^{(t_2-1)} + (\mathbf{g}_i^{(t_2-1)} - \mathbf{d}_i^{(t_2-1)})\|^2 = \cdots = \|\sum_{t=t_1+1}^{t_2-1} (\mathbf{g}_i^{(t)} - \mathbf{d}_i^{(t)})\|^2 \le A\|\mathbf{g}_i^{(t_2-1)}\|^2 + B$$

Using $\|a-b\|^2 \ge \frac{\|a\|^2}{2} - \|b\|^2$ and $\mathbf{d}_i^{(t_1+1)} = \mathbf{d}_i^{(t_1+2)} = \cdots = \mathbf{d}_i^{(t_2-1)}$:

$$\frac{1}{2}(t_2 - t_1 - 1)\sum_{t=t_1+1}^{t_2-1} \|\mathbf{d}_i^{(t)}\|^2 \le \|\sum_{t=t_1+1}^{t_2-1} \mathbf{g}_i^{(t)}\|^2 + A\|\mathbf{g}_i^{(t_2-1)}\|^2 + B$$

By Cauchy-Schwarz, $\|\sum_{t=t_1+1}^{t_2-1} \mathbf{g}_i^{(t)}\|^2 \le (t_2 - t_1 - 1)\sum_{t=t_1+1}^{t_2-1} \|\mathbf{g}_i^{(t)}\|^2$. Dividing the above inequality by $(t_2 - t_1 - 1)$, we have:

$$\sum_{t=t_1+1}^{t_2-1} \|\mathbf{d}_i^{(t)}\|^2 \le 2\sum_{t=t_1+1}^{t_2-1} \|\mathbf{g}_i^{(t)}\|^2 + \frac{2A}{t_2 - t_1 - 1}\|\mathbf{g}_i^{(t_2-1)}\|^2 + \frac{2B}{t_2 - t_1 - 1}$$

Since $\mathbf{d}_i^{(t_2)} = \mathbf{g}_i^{(t_1)}$ (defining $\mathbf{g}_i^{(-1)} = 0$ for $t_1 = -1$) and $t_2 - t_1 - 1 \ge 1$:

$$\sum_{t=t_1+1}^{t_2} \|\mathbf{d}_i^{(t)}\|^2 \le 2\sum_{t=t_1}^{t_2-1} \|\mathbf{g}_i^{(t)}\|^2 + 2A\|\mathbf{g}_i^{(t_2-1)}\|^2 + 2B$$

Finally, splitting $[0:T-1]$ into $-1=t_0<t_1<\ldots<t_k=T$ such that the i-th client communicates at iterations t_j, we have

$$\begin{aligned}\sum_{t=0}^{T-1}\|\mathbf{d}_i^{(t)}\|^2 &= \sum_{j=0}^{k}\sum_{t=t_j+1}^{t_{j+1}}\|\mathbf{d}_i^{(t)}\|^2\\ &\le 2\sum_{j=0}^{k}\left(\sum_{t=t_j+1}^{t_{j+1}}\|\mathbf{g}_i^{(t)}\|^2+A\|\mathbf{g}_i^{(t_{j+1}-1)}\|^2+B\right)\\ &\le 2((1+A)\sum_{t=0}^{T-1}\|\mathbf{g}_i^{(t)}\|^2+BT),\end{aligned}$$

Bounding $\mathbf{g}_i^{(t)}$ in terms of $\nabla f(\mathbf{x}^{(t)})$. By Assumption 3, we have

$$\|\nabla f_i(\mathbf{x})-\nabla f(\mathbf{x})\|^2\le\alpha\|\nabla f(\mathbf{x})\|^2+\beta$$

Using inequality $\|a-b\|^2\ge\frac{1}{2}\|a\|^2-\|b\|^2$, we have:

$$\frac{1}{2}\|\nabla f_i(\mathbf{x})\|^2-\|\nabla f(\mathbf{x})\|^2\le\alpha\|\nabla f(\mathbf{x})\|^2+\beta\implies\|\nabla f_i(\mathbf{x})\|^2\le 2(1+\alpha)\|\nabla f(\mathbf{x})\|^2+2\beta$$

Using $\mathbb{E}\|\mathbf{g}_i^{(t)}\|^2=\mathbb{E}\|\nabla f_i(\mathbf{x}^{(t)})\|^2+\mathbb{E}\|\mathbf{g}_i^{(t)}-\nabla f_i(\mathbf{x}^{(t)})\|^2=\mathbb{E}\|\nabla f_i(\mathbf{x}^{(t)})\|^2+\sigma^2$, for $\mathbf{g}_i^{(t)}$ we have

$$\mathbb{E}\|\mathbf{g}_i^{(t)}(\mathbf{x})\|^2\le 2(1+\alpha)\mathbb{E}\|\nabla f(\mathbf{x})\|^2+2\beta+\sigma^2$$

Bounding $\sum_t\mathbb{E}\|\xi^{(t)}\|^2$. Putting the above bounds together, we have

$$\begin{aligned}\sum_{t=0}^{T-1}\mathbb{E}\|\xi^{(t)}\|^2 &\le 2\sum_{t=0}^{T-1}\mathbb{E}[\|\mathbf{r}^{(t)}\|^2+\operatorname*{avg}_i\|\mathbf{e}_i^{(t)}\|^2]\\ &\le 2(C\sum_{t=0}^{T-1}\operatorname*{avg}_i\mathbb{E}[\|\mathbf{d}_i^{(t)}\|^2+DT+A\sum_{t=0}^{T-1}\mathbb{E}\|\mathbf{g}_i^{(t)}\|^2+BT)\\ &\le 2(2C((1+A)\sum_{t=0}^{T-1}\mathbb{E}\|\mathbf{g}_i^{(t)}\|^2+BT)+DT+A\sum_{t=0}^{T-1}\mathbb{E}\|\mathbf{g}_i^{(t)}\|^2+BT)\\ &\le 4(A+AC+C)\sum_{t=0}^{T-1}\mathbb{E}\|\mathbf{g}_i^{(t)}\|^2+4(BC+D)T\\ &\le 8(A+AC+C)(1+\alpha)\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})_i\|^2+4((A+AC+C)(\beta+\sigma^2)+BC+D)T.\end{aligned}$$

■

Proof [Proof of the main theorem] By Lemma 7 and using bound on $\sum_{t=0}^{T-1}\mathbb{E}\|\xi^{(t)}\|^2$:

$$\begin{aligned}
&\mathbb{E}[f(\mathbf{y}^{(T)})] \\
&\leq f(\mathbf{y}^{(0)}) - \sum_{t=0}^{T-1}\frac{\gamma}{2}(1-L\gamma)\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + T\frac{L\gamma^2\sigma^2}{2N} + \frac{L^2\gamma^3}{2}\sum_{t=0}^{T-1}\mathbb{E}\|\xi^{(t)}\|^2 \\
&\leq f(\mathbf{y}^{(0)}) - \sum_{t=0}^{T-1}\frac{\gamma}{2}(1-L\gamma)\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + T\frac{L\gamma^2\sigma^2}{2N} + L^2\gamma^3(c_1\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + Tc_2) \\
&= f(\mathbf{y}^{(0)}) - \sum_{t=0}^{T-1}\frac{\gamma}{2}(1-L\gamma-c_1L^2\gamma^2)\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + T\frac{L\gamma^2\sigma^2}{2N} + c_2TL^2\gamma^3 \\
&= f(\mathbf{y}^{(0)}) - \sum_{t=0}^{T-1}\frac{\gamma}{4}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 + T\frac{L\gamma^2\sigma^2}{2N} + c_2TL^2\gamma^3,
\end{aligned}$$

where the last inequality obtained by selecting $\gamma \leq \frac{1}{4L}$ and $\gamma^2 \leq \frac{1}{4c_1L^2}$. Rearranging the terms and dividing by $\frac{\gamma T}{4}$, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 \leq \frac{4(f(\mathbf{y}^{(0)}) - \mathbb{E}[f(\mathbf{y}^{(T)})])}{\gamma T} + \frac{2L\gamma\sigma^2}{N} + 4c_2L^2\gamma^2.$$

The rest of the proof follows that of [32, Lemma 17]. Let $F = f(\mathbf{y}^{(0)}) - f^\star \geq f(\mathbf{y}^{(0)}) - \mathbb{E}[f(\mathbf{y}^{(T)})]$. Balancing the first two terms:

$$\frac{4F}{\gamma T} = \frac{2L\gamma\sigma^2}{N} \implies \gamma = \frac{\sqrt{2FN}}{\sqrt{L}\sigma}$$

Balancing the first and the last term:

$$\frac{4F}{\gamma T} = 4c_2L^2\gamma^2 \implies \gamma = \left(\frac{F}{c_2L^2T}\right)^{1/3}$$

Therefore, by selecting $\gamma = \min\left(\frac{\sqrt{2FN}}{\sqrt{L}\sigma}, \left(\frac{F}{c_2L^2T}\right)^{1/3}, \frac{1}{2L\sqrt{c_1}}\right)$:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(t)})\|^2 = O\left(\frac{\sqrt{LF}\sigma}{\sqrt{NT}} + \left(\frac{FL\sqrt{c_2}}{T}\right)^{2/3} + \frac{FL\sqrt{c_1}}{T}\right).$$

■

B.2. Convex Case

Theorem 9 *Let f be a convex function satisfying Assumptions 1-3. Let c_1 and c_2 be as defined in Lemma 8. Let $\{\mathbf{x}^{(t)}\}$ be the sequence from Algorithm 2 with $\gamma \leq \min\{\frac{1}{4(1+\alpha)}, \frac{1}{8\sqrt{c_1}L}\}$. Then*

$$\mathbb{E}[f(\frac{1}{T}\sum_{\tau=0}^{T-1}\mathbf{x}^{(\tau)}) - f(\mathbf{x}^\star)] \leq 16\gamma^2c_2L + \frac{2\|\mathbf{x}^{(0)} - \mathbf{x}^\star\|^2}{\gamma T} + 2\gamma(2\beta + \frac{\sigma^2}{N})$$

Therefore, for $\gamma = \Theta(1/\sqrt{T})$ we achieve $O(1/\sqrt{T})$ convergence rate.

Proof Let $\mathbb{E}_t$ denote expectation conditioned on $\mathbf{x}^{(t)}, \mathbf{e}_i^{(t)}, \mathbf{r}^{(t)}$. Since $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} - \gamma \operatorname{avg}_{i\in[N]}(\mathbf{g}_i^{(t)})$:

$$\begin{aligned}
\mathbb{E}_t\|\mathbf{y}^{(t+1)} - \mathbf{x}^\star\|^2 &= \|\mathbf{y}^{(t)} - \gamma \operatorname*{avg}_{i\in[N]}(\mathbf{g}_i^{(t)}) - \mathbf{x}^\star\|^2 \\
&= \|\mathbf{y}^{(t)} - \mathbf{x}^\star\|^2 + \gamma^2\mathbb{E}_t\|\operatorname*{avg}_{i\in[N]}(\mathbf{g}_i^{(t)})\|^2 - 2\gamma\mathbb{E}_t\langle \operatorname*{avg}_{i\in[N]}(\mathbf{g}_i^{(t)}), \mathbf{y}^{(t)} - \mathbf{x}^\star\rangle \\
&= \|\mathbf{y}^{(t)} - \mathbf{x}^\star\|^2 + \gamma^2\mathbb{E}_t\|\operatorname*{avg}_{i\in[N]}(\mathbf{g}_i^{(t)})\|^2 - 2\gamma\langle \nabla f(\mathbf{x}^{(t)}), \mathbf{y}^{(t)} - \mathbf{x}^\star\rangle
\end{aligned}$$

For the last term, we have:

$$\begin{aligned}
-2\gamma\langle \nabla f(\mathbf{x}^{(t)}), \mathbf{y}^{(t)} - \mathbf{x}^\star\rangle &= -2\gamma\langle \nabla f(\mathbf{x}^{(t)}), \mathbf{y}^{(t)} - \mathbf{x}^{(t)}\rangle - 2\gamma\langle \nabla f(\mathbf{x}^{(t)}), \mathbf{x}^{(t)} - \mathbf{x}^\star\rangle \\
&\le 2\gamma^2\|\nabla f(\mathbf{x}^{(t)})\| \cdot \|\xi^{(t)}\| - 2\gamma\langle \nabla f(\mathbf{x}^{(t)}), \mathbf{x}^{(t)} - \mathbf{x}^\star\rangle
\end{aligned}$$

Substituting this into the inequality above, by taking expectation and using telescoping, we have:

$$\begin{aligned}
\mathbb{E}\|\mathbf{y}^{(t)} - \mathbf{x}^\star\|^2 \le \|\mathbf{x}^{(0)} - \mathbf{x}^\star\|^2 &+ \gamma^2\sum_{\tau=0}^{t-1}\mathbb{E}\|\operatorname*{avg}_{i\in[N]}(\mathbf{g}_i^{(\tau)})\|^2 \\
&+ 2\gamma^2\sum_{\tau=0}^{t-1}\mathbb{E}[\|\nabla f(\mathbf{x}^{(\tau)})\| \cdot \|\xi^{(\tau)}\|] - 2\gamma\sum_{\tau=0}^{t-1}\mathbb{E}\langle \nabla f(\mathbf{x}^{(\tau)}), \mathbf{x}^{(\tau)} - \mathbf{x}^\star\rangle \qquad (2)
\end{aligned}$$

We'll simplify the terms on the right-hand side. Using the fact that stochastic noises are independent, as shown in Lemma 8:

$$\sum_{\tau=0}^{t-1}\mathbb{E}\|\mathbf{g}_i^{(\tau)}(\mathbf{x}^{(\tau)})\|^2 \le 2(1+\alpha)\sum_{\tau=0}^{t-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(\tau)})\|^2 + t(2\beta + \frac{\sigma^2}{N})$$

By Cauchy-Schwarz inequality and by Lemma 8:

$$\begin{aligned}
\sum_{\tau=0}^{t-1}\mathbb{E}[\|\nabla f(\mathbf{x}^{(\tau)})\| \cdot \|\xi^{(\tau)}\|] &\le \sqrt{\mathbb{E}[\sum_{\tau=0}^{t-1}\|\nabla f(\mathbf{x}^{(\tau)})\|^2] \cdot \mathbb{E}[\|\sum_{\tau=0}^{t-1}\xi^{(\tau)}\|^2]} \\
&\le \sqrt{\mathbb{E}[\sum_{\tau=0}^{t-1}\|\nabla f(\mathbf{x}^{(\tau)})\|^2] \cdot \mathbb{E}[c_1\sum_{\tau=0}^{t-1}\|\nabla f(\mathbf{x}^{(\tau)})\|^2 + c_2 t]} \\
&\le \sqrt{c_1}\sum_{\tau=0}^{t-1}\mathbb{E}\|\nabla f(\mathbf{x}^{(\tau)})\|^2 + \sqrt{c_2}\sqrt{t}\sqrt{\mathbb{E}[\sum_{\tau=0}^{t-1}\|\nabla f(\mathbf{x}^{(\tau)})\|^2]}.
\end{aligned}$$

By convexity:

$$-2\gamma\sum_{\tau=0}^{t-1}\mathbb{E}\langle \nabla f(\mathbf{x}^{(\tau)}), \mathbf{x}^{(\tau)} - \mathbf{x}^\star\rangle \le -2\gamma\sum_{\tau=0}^{t-1}\mathbb{E}[f(\mathbf{x}^{(\tau)}) - f(\mathbf{x}^\star)]$$

and for smooth convex functions we have:

$$\mathbb{E}\|\nabla f(\mathbf{x}^{(\tau)})\|^2 = \mathbb{E}\|\nabla f(\mathbf{x}^{(\tau)}) - \nabla f(\mathbf{x}^\star)\|^2 \leq 2L\mathbb{E}[f(\mathbf{x}^{(\tau)}) - f(\mathbf{x}^\star)]$$

Let's denote $\sqrt{\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}[f(\mathbf{x}^{(\tau)}) - f(\mathbf{x}^\star)]}$ as S_t. Substituting the bounds above into Equality (2) and dividing it by t:

$$0 \leq \frac{\|\mathbf{x}^{(0)} - \mathbf{x}^\star\|^2}{t} + 2\gamma^2(1+\alpha)S_t^2 + \gamma^2(2\beta + \frac{\sigma^2}{N}) + 2\gamma^2\sqrt{c_1}LS_t^2 + 4\gamma^2\sqrt{c_2 L}S_t - 2\gamma S_t^2$$

When $\gamma \leq \min\{\frac{1}{4(1+\alpha)}, \frac{1}{8\sqrt{c_1}L}\}$, it follows:

$$S_t^2 - 4\gamma\sqrt{c_2 L}S_t \leq \frac{\|\mathbf{x}^{(0)} - \mathbf{x}^\star\|^2}{\gamma t} + \gamma(2\beta + \frac{\sigma^2}{N})$$

By inequality $ab \leq \frac{a^2}{2} + \frac{b^2}{2}$, we have $4\gamma\sqrt{c_2 L}S_t \leq \frac{S_t^2}{2} + 8\gamma^2 c_2 L$, and therefore:

$$S_t^2 \leq 16\gamma^2 c_2 L + \frac{2\|\mathbf{x}^{(0)} - \mathbf{x}^\star\|^2}{\gamma t} + 2\gamma(2\beta + \frac{\sigma^2}{N})$$

Finally, by convexity:

$$S_t^2 = \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}[f(\mathbf{x}^{(\tau)}) - f(\mathbf{x}^\star)] \geq \mathbb{E}[f(\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbf{x}^{(\tau)}) - f(\mathbf{x}^\star)]\,.$$

■

Appendix C. Experiments

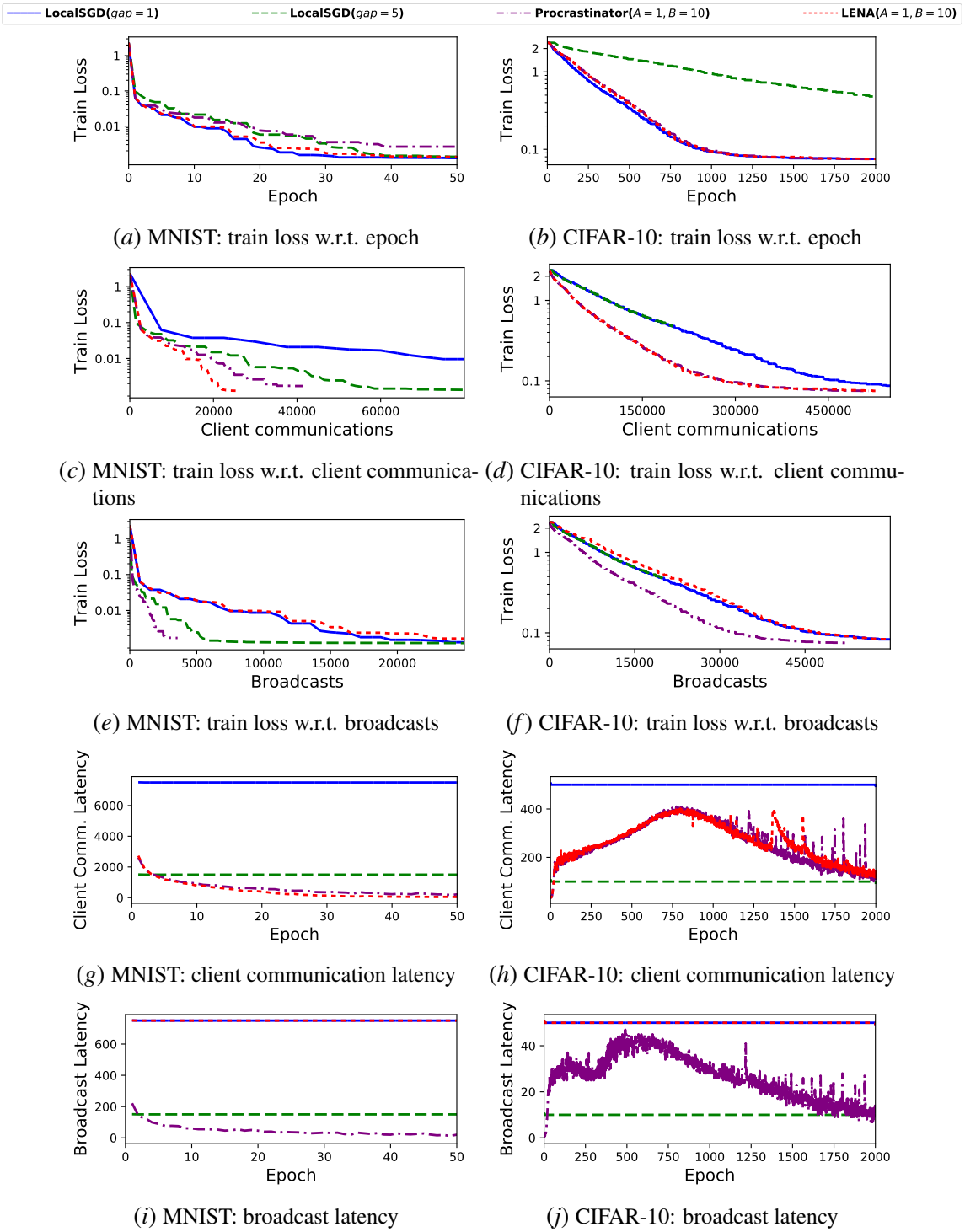
In this section, we empirically show convergence and communication improvements of PROCRASTINATOR. We perform experiments on three datasets: MNIST [34], CIFAR-10 [33] and CIFAR-100 [33]. For MNIST, we train a deep convolutional model with $\gamma = 0.1$ and batch size 8, while for CIFAR-10 we train the VGG neural network [49] with $\gamma = 0.01$ and batch size 100.

In our experiments, we consider the following approaches:

- **PROCRASTINATOR** with parameters (A, B) as in Algorithm 1.
- **LENA** [48] with parameters (A, B).
- **Local SGD** with parameter gap. Each worker makes gradient descent step (i.e. $\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{x}_i^{(t)} - \gamma\mathbf{g}_i^{(t)}$), and every gap iterations all worker synchronize their parameters.

For PROCRASTINATOR and LENA, we selected $A = 1$ and $B = 10$ as parameters which work well across multiple tasks. For local SGD, we considered $gap = 1$ as the most basic baseline (the algorithm becomes the distributed SGD), and $gap = 5$, since with such a gap, communication of Local SGD is comparable to communication of PROCRASTINATOR and LENA.

Our results are shown in Figure 1. For each dataset, we report the following (additional results, including test loss, are shown in the Appendix):



(*a*) MNIST: train loss w.r.t. epoch

(*b*) CIFAR-10: train loss w.r.t. epoch

(*c*) MNIST: train loss w.r.t. client communications

(*d*) CIFAR-10: train loss w.r.t. client communications

(*e*) MNIST: train loss w.r.t. broadcasts

(*f*) CIFAR-10: train loss w.r.t. broadcasts

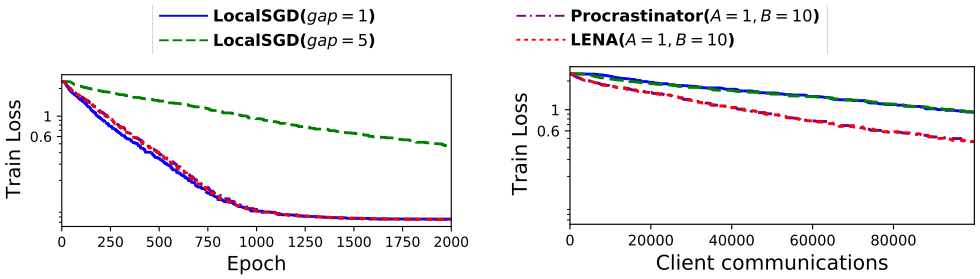
(*g*) MNIST: client communication latency

(*h*) CIFAR-10: client communication latency

(*i*) MNIST: broadcast latency

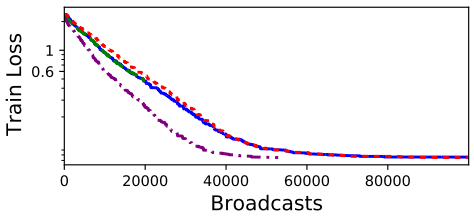
(*j*) CIFAR-10: broadcast latency

Figure 1: Convergence of distributed SGD (local SGD with $gap = 1$), local SGD with $gap = 5$, PROCRASTINATOR and LENA on MNIST (left) and CIFAR-10 datasets. For each dataset, we show train loss with respect to: the number of iterations, the number of communications from clients and the number of broadcasts. Additionally, we show the number of client communications and the number of broadcasts per epoch. The results show that PROCRASTINATOR and LENA have the best convergence w.r.t. client communication, while PROCRASTINATOR additionally has the best convergence w.r.t. broadcasts.

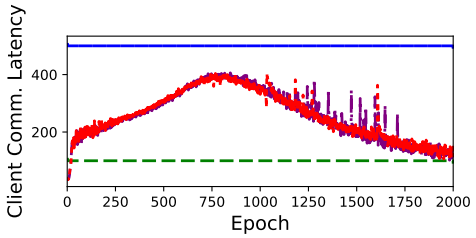


(*a*) CIFAR-100: train loss w.r.t. epoch

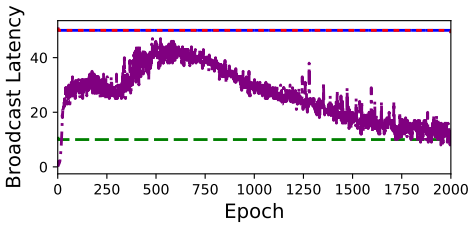
(*b*) CIFAR-100: train loss w.r.t. client communications



(*c*) CIFAR-100: train loss w.r.t. broadcasts



(*d*) CIFAR-100: client communication latency



(*e*) CIFAR-100: broadcast latency

Figure 2: Convergence of distributed SGD (local SGD with $gap = 1$), local SGD with $gap = 5$, PROCRASTINATOR and LENA on CIFAR-100 dataset. Similarly to Figure 1, PROCRASTINATOR and LENA have the best convergence w.r.t. client communication, while PROCRASTINATOR additionally has the best convergence w.r.t. broadcasts.

- Train loss with respect to the number of epochs, the number of client communications and the number of broadcasts.
- Client communication latency and broadcast latency. Namely, for each epoch we report the number of communications/broadcasts during that epoch.

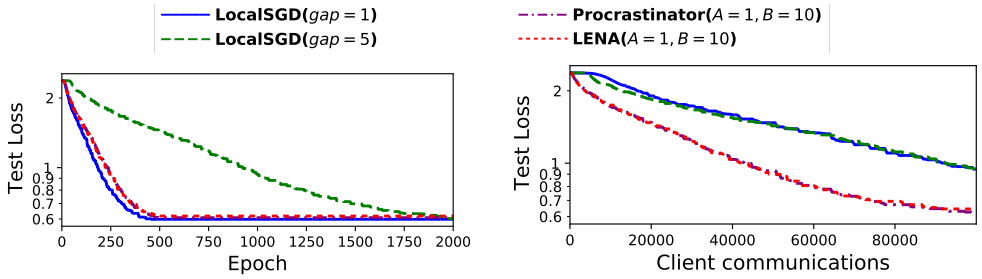
For the MNIST dataset, Figure 1(a) shows that all algorithms have comparable convergence rates, and therefore, communication becomes the main deciding performance factor. In Figure 1(c), LENA requires the least amount of communication, followed by PROCRASTINATOR and substantially outperforming local SGD and distributed SGD in terms of communication required to reach the final accuracy. However, LENA, similarly to distributed SGD, requires broadcasts at every iteration, and therefore is significantly outperformed by PROCRASTINATOR in this aspect.

On CIFAR-10 and CIFAR-100 datasets, algorithm behavior is noticeably different. In Figure 1(b), all algorithms except Local SGD have similar performance. However, with respect to client communication, PROCRASTINATOR and LENA have the best performance, requiring 2x-5x times less communication compared with distributed/local SGD. And due to broadcast requirements of LENA, PROCRASTINATOR shows the best communication performance: 1.5x-2x times less broadcasts compared with distributed/local SGD.

Overall, PROCRASTINATOR outperforms local/distributed SGD with respect to communication. Compared with LENA, it shows similar client communication requirements, while requiring substantially less broadcasts.

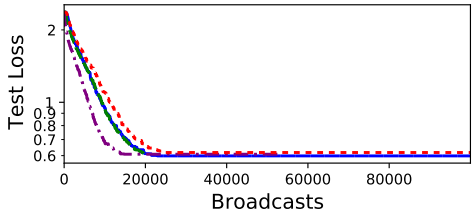
C.1. Additional experimental results

In this section, we show the test loss convergence for datasets under study.



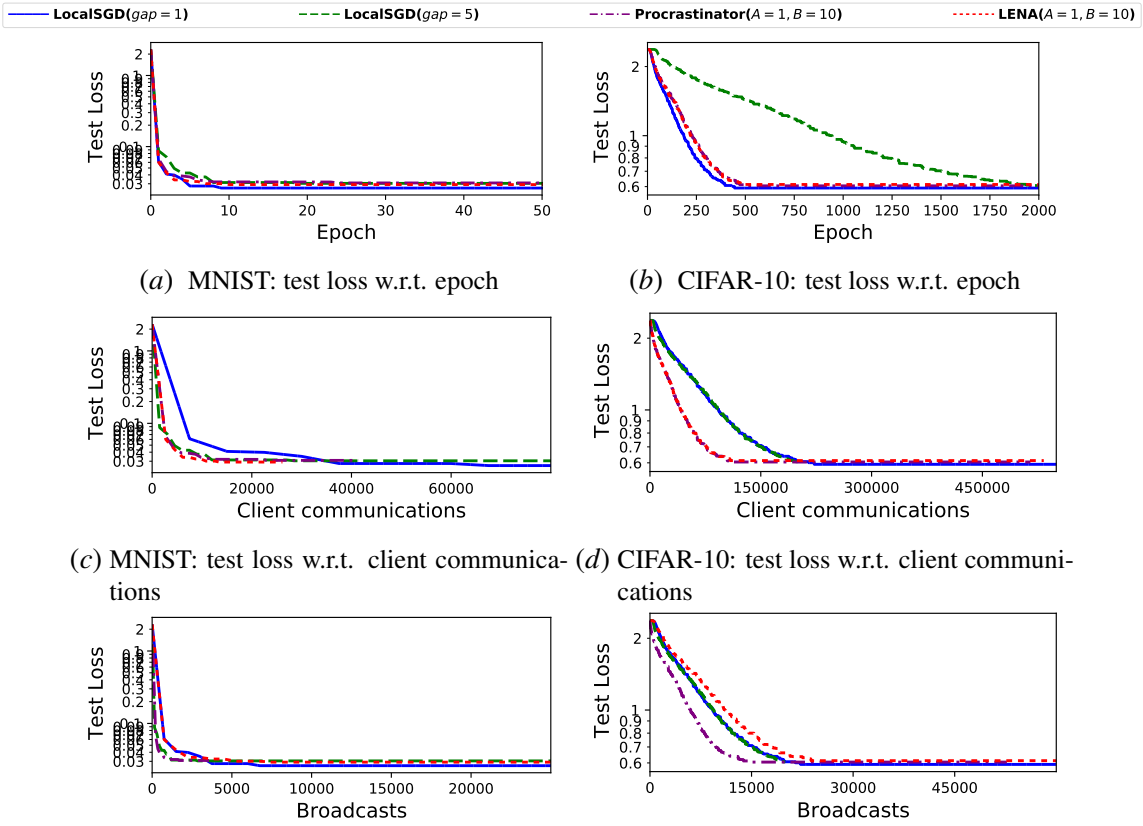
(*a*) CIFAR-100: test loss w.r.t. epoch

(*b*) CIFAR-100: test loss w.r.t. client communications



(*c*) CIFAR-100: test loss w.r.t. broadcasts

Figure 3: Test loss convergence of distributed SGD (local SGD with $gap = 1$), local SGD with $gap = 5$, PROCRASTINATOR and LENA on CIFAR-100 dataset.



(*a*) MNIST: test loss w.r.t. epoch

(*b*) CIFAR-10: test loss w.r.t. epoch

(*c*) MNIST: test loss w.r.t. client communications

(*d*) CIFAR-10: test loss w.r.t. client communications

(*e*) MNIST: test loss w.r.t. broadcasts

(*f*) CIFAR-10: test loss w.r.t. broadcasts

Figure 4: Test loss convergence of distributed SGD (local SGD with $gap = 1$), local SGD with $gap = 5$, PROCRASTINATOR and LENA on MNIST (left) and CIFAR-10 datasets. For each dataset, we show test loss with respect to: the number of iterations, the number of communications from clients and the number of broadcasts.