Accelerated Methods with Compressed Communications for Distributed Optimization Problems Under Data Similarity

Dmitry Bylinkin[1,2], **Aleksandr Beznosikov**[2, 1, 3, 4]
[1]Moscow Institute of Physics and Technology
[2]Ivannikov Institute for System Programming of the Russian Academy of Sciences
[3]Sber AI Lab
[4]Skoltech
da.bylinkin@gmail.com, anbeznosikov@gmail.com

Abstract

In recent years, as data and problem sizes have increased, distributed learning has become an essential tool for training high-performance models. However, the communication bottleneck, especially for high-dimensional data, is a challenge. Several techniques have been developed to overcome this problem. These include communication compression and implementation of local steps, which work particularly well when there is similarity of local data samples. In this paper, we study the synergy of these approaches for efficient distributed optimization. We propose the first theoretically grounded accelerated algorithms utilizing unbiased and biased compression under data similarity, leveraging variance reduction and error feedback frameworks. In terms of communication time our theory gives $\tilde{\mathcal{O}}\left(1+\left[M^{-1/4}+\omega^{-1/2}\right]\sqrt{\delta/\mu}\right)$ complexity for unbiased compressors and $\tilde{\mathcal{O}}\left(1+\beta^{1/4}\sqrt{\delta/\mu}\right)$ for biased ones, where M is the number of computational nodes, β is the compression power, δ is the similarity measure and μ is the parameter of strong convexity of the objective. Our theoretical results are of record and confirmed by experiments on different average losses and datasets.

1 Introduction

Conventional machine/deep learning algorithms often struggle to handle the scale and complexity of modern datasets, resulting in long training times and limited scalability. Distributed optimization (Verbraeken et al. 2020) has witnessed remarkable progress, driven by rising demand for efficient methods across diverse applications such as medical image analysis (Wu et al. 2022), chemical physics (Zhu, Luo, and White 2022), predictive maintenance (Bidollahkhani and Kunkel 2024) and natural language processing (Bai 2022). Distributed learning addresses emerging challenges by spreading the training process across multiple nodes, allowing scientists and engineers to process and analyze data that would be impractical to handle on a single machine (Alqahtani and Demirbas 2019). In terms of optimization, we have the following problem statement:

$$\min_{x\in\mathbb{R}^d}\left[f(x)=\frac{1}{M}\sum_{m=1}^{M}f_m(x)\right] \tag{1}$$

$$\text{with}\quad f_m(x)=\frac{1}{n_m}\sum_{j=1}^{n_m}\ell(x,z_j^m),$$

where M refers to the number of nodes/devices/clients/agents/machines, n_m is the size of the local dataset on m-th machine, x is the vector representation of the model using d features, z_j^m is the j-th data point on the m-th node and ℓ is the loss function. z_j^m is an ordered pair of feature description a_j^i and label b_j^m. We consider an architecture with a star-network topology, that is, a server, represented by f_1, plays a crucial role as the main computational hub. The other nodes act as users that can communicate with the server but not directly with each other. This hierarchical structure allows for easy coordination through the first node within the system.

When training modern distributed models, the bottleneck is often the cost of communicating information (Konečnỳ et al. 2016; Jordan, Lee, and Yang 2019). It is a known fact that client-to-server communication is much more resource-intensive than server-to-client (Mishchenko et al. 2019; Kairouz et al. 2021). As a result, considerable effort has been devoted to development of distributed optimization methods that would be efficient in terms of nodes to server communication. Significant success has been achieved with the development of compression techniques, that help to reduce the amount of data transferred between nodes during training. In this paper, we deal with two main classes of compressors: unbiased and biased (Beznosikov et al. 2023).

Definition 1. *We call the mapping $Q:\mathbb{R}^d\to\mathbb{R}^d$ an unbiased compressor, if there exists a constant $\omega>1$ such that*

$$\mathbb{E}_Q\left[Q(x)\right]=x,\ \mathbb{E}_Q\left[\|Q(x)-x\|^2\right]\le\omega\|x\|^2,\ \forall x\in\mathbb{R}^d.$$

Definition 2. *We call the mapping $C:\mathbb{R}^d\to\mathbb{R}^d$ a biased compressor, if there exists a constant $\beta>1$ such that*

$$\mathbb{E}_C\left[\|C(x)-x\|^2\right]\le\left(1-\frac{1}{\beta}\right)\|x\|^2,\ \forall x\in\mathbb{R}^d.$$

Let us denote by γ_ω the value showing how much the operator $Q(z)$ compresses the input vector on average. Let b be the number of bits needed to represent a single float, and $\|\cdot\|_{bits}$ be the number of bits needed to represent the input vector. We define $\gamma_\omega^{-1}=\frac{1}{bd}\mathbb{E}\|Q(z)\|_{bits}$. Similarly, we introduce the compressive force γ_β for $C(z)$. For practical compressors,

we have $\gamma_\omega \geq \omega$ and $\gamma_\beta \geq \beta$ (Vogels, Karimireddy, and Jaggi 2019; Beznosikov et al. 2023). While methods with unbiased compressors may theoretically provide more optimistic estimates (Gorbunov, Hanzely, and Richtárik 2020; Gorbunov et al. 2021; Li et al. 2020), biased ones show a notable advantage in practical applications (Sun et al. 2019). Consequently, there is a growing interest in biased compression techniques. However, analyzing biased compressors is challenging, and there is limited understanding of their behavior (Beznosikov et al. 2023; Richtárik, Sokolov, and Fatkhullin 2021; Stich and Karimireddy 2019).

Another technique that addresses the communication bottleneck in distributed optimization is utilizing local steps under the Hessian similarity condition (Shamir, Srebro, and Zhang 2014; Hendrikx et al. 2020; Kovalev et al. 2022). In this scenario, a single node represents the average nature of the data across all ones.

Definition 3. *We say that there is the Hessian similarity (δ-relatedness) between f_i and f, if there exists a constant $\delta > 0$ such that*

$$\|\nabla^2 f_i(x) - \nabla^2 f(x)\| \leq \delta, \quad \forall x \in \mathbb{R}^d.$$

Consider f to be L-smooth. As the size n of data, distributed uniformly across the nodes, increases, the losses become more statistically similar. In the quadratic case, it is shown that $\delta \sim L/n$. Otherwise, we have $\delta \sim L/\sqrt{n}$ for any non-quadratic losses (Hendrikx et al. 2020). As a result, the server gets to contact the clients less often to compute the full gradient. Despite the fact that research on distributed optimization via compression or similarity has been going on for quite some time, there are still open questions:

1. *How to build accelerated methods that use both compression and similarity?*
2. *Would such methods be superior to existing SOTAs in terms of communication efficiency?*

2 Notation

When we talk about the communication efficiency of an algorithm, it is important to choose the right definition of its communication complexity. This can be done in several ways.

1. *Number of communication rounds (CC-1).* The number of times the server initiates communication with clients is used as a complexity measure. This does not take into account the number of involved machines. Even if the server has communicated with all the devices within a communication round, this counts the same as if it only has communicated with one.
2. *Number of client-server communications (CC-2).* It arises when we recognize that the number of rounds of communication is not sufficient to adequately compare distributed methods. For example, if the nodes operate asynchronously. In this case, the more appropriate metric is the total number of communications rather than the number of rounds. Using this approach, we begin to experience the superiority of methods that turn out to be bad in the sense of *CC-1*.
3. *Communication time (CC-3).* We consider a synchronous setup. Let all the devices and their communication channels to the server be equivalent. Let transmission time of one unit of information from the devices to the server take τ time units. Also, if the clients start communicating with the server at same time, the channel between them is initialized in negligible time. Then, the total time of one such round of communication is τK, where K is the amount of information transmitted from each machine to the server. This definition allows us to see the strengths of methods with compressed communications, since they can change K.

3 Related Works

3.1 Distributed Learning via Hessian Similarity

Now that the communication complexity is suitably defined, let us move on to the survey on effective communication techniques. The first work around similarity was the Newton-type `DANE` method, developed for quadratic strongly convex functions (Shamir, Srebro, and Zhang 2014). For this class of problems, a *CC-1* lower bound was proved in (Arjevani and Shamir 2015). `DANE` did not reach it. This raised the question of how to fill this gap. History of work on this problem counts many papers. Nevertheless, all of them either did not reach exactly the bound or considered special cases (Zhang and Lin 2015; Lu, Freund, and Nesterov 2018; Yuan and Li 2020; Beznosikov et al. 2021; Tian et al. 2022). Recently, `Accelerated ExtraGradient` enjoying optimal *CC-1* communication complexity under the Hessian similarity condition was constructed in (Kovalev et al. 2022).

Exploiting the similarity is not the only approach to effective communication. There are also the compression techniques, to which we devote the next subsection.

3.2 Distributed Learning via Compression

There has been a significant amount of research done on communication compression. Especially on unbiased operators, due to their ease of analysis. The first family of compression schemes with convergence guarantees was proposed in one-device setup by Alistarh et al. (2017). It was suggested to perform gradient descent steps using compressed gradients. An extension of the proposed approach to multiple nodes was made a year later, when the first method `DQGD` for the distributed setup appeared (Khirirat, Feyzmahdavian, and Johansson 2018). The following was suggested: the node gets the current point, calculates the local gradient, then compresses and sends it to the server to perform the gradient step. This scheme is quite simple and easy to analyze, but it has a number of drawbacks, which is discussed below. A comprehensive review of unbiased compression can be found in (He, Huang, and Yuan 2024). As noted before, the nature of unbiased compression is quite simple and one can design methods simply by replacing the stochastic gradient with a compressed one. At the same time, biased compression shows better empirical results (Vogels, Karimireddy, and Jaggi 2019), but its nature is non-trivial and requires a special approach. In 2014, a framework for constructing methods with biased compression was proposed (Seide et al. 2014).

The key idea is that each node remembers the "error" it made when compressing the local gradient, and then takes it into account in a special way in the next rounds. This approach is called Error Feedback. There were no theoretically proven results without unnatural assumptions until (Stich and Karimireddy 2019; Beznosikov et al. 2023).

In all mentioned papers both for unbiased and biased compression a similar problem arisen – none of the named algorithms converged to the true optimum. The reason is uncontrolled variance of compressed gradient difference, which results in its approximation not tending to zero as the algorithm runs. After reaching some neighborhood of the solution, the method loses the ability to take small enough steps to avoid "overshooting" the optimum. This could be solved by the variance reduction (VR) technique.

3.3 Variance Reduction

Originally, variance reduction was designed to solve the convergence problem of SGD (Robbins and Monro 1951). Classical stochastic optimization methods such as SGD face the same problem as the simplest distributed methods with compression: approximation of the gradient does not tend to zero when searching for the optimum. Thus, there is convergence to its neighborhood only. The variance reduction framework allows to correct this drawback of naive methods. There are two main approaches: SAGA (Defazio, Bach, and Lacoste-Julien 2014) and SVRG (Johnson and Zhang 2013). The first stores the history of evoked gradients by mini-batches, resulting in a more accurate approximation of the full gradient. The second has no "memory", but occasionally recalculates the full gradient. The optimal stochastic optimization algorithm with variance reduction is KATYUSHA (Allen-Zhu 2018a). Its various modifications are of great interest (Kovalev, Horváth, and Richtárik 2020; Allen-Zhu 2018b). These ideas could be developed to solve problems arising in the analysis of methods utilizing unbiased compression. For example, see DIANA (Mishchenko et al. 2019) and its accelerated version ADIANA (Li et al. 2020). In the non-convex case, the current state-of-the-art MARINA is also based on the VR idea (Gorbunov et al. 2021). Variance reduction can be exploited in biased compression methods as well, see EF21 (Richtárik, Sokolov, and Fatkhullin 2021), ECLK (Qian, Richtárik, and Zhang 2021). It is also known how to construct variance reduction schemes for a more general class of problems than minimization – variational inequalities (VIs) (Alacaoglu and Malitsky 2022). This approach is useful for developing methods that combine compression with other approaches, such as local steps and similarity (Beznosikov, Takác, and Gasnikov 2024; Beznosikov et al. 2022).

Since our goal is to utilize the synergy of multiple techniques to develop methods for effective communication, we devote the next subsection to a brief overview and comparison of the best existing methods.

3.4 Effective Communication via Combination of Techniques

One of the modern methods combining different techniques to communicate efficiently is LoCoDL (Condat, Maranjyan, and Richtárik 2024). Using compression and local steps, it is possible to construct a method that, in the case of $\omega > M$, repeats the result of ADIANA in the sense of *CC-1*, and outperforms it in the sense of *CC-2* and *CC-3*. The authors of (Condat et al. 2023) managed to add client sampling to this combination of techniques. Their TAMUNA gives stronger results in some special cases. The main weakness of the above methods is that they do not exploit data similarity. Thus, if the ratio δ/L is small enough, these methods lose significantly to SOTAs that use similarity. Accelerated Extragradient (Kovalev et al. 2022) is unbeatable in the sense of *CC-1* and superior to LoCoDL and TAMUNA in the sense of *CC-2* and *CC-3*. These results are achieved by combining similarity and local steps. AccSVRS (Lin et al. 2024), which uses client sampling in addition to these techniques, loses in the sense of *CC-1*. However, in the sense of *CC-2*, this method is optimal. One can note the current best methods in terms of *CC-3* are Accelerated ExtraGradient and Three Pillars Algorithm. The first one is accelerated, but does not use compression. The second one uses unbiased compression but does not use acceleration. Thus, which of these two algorithms performs better depends on the ratio of the similarity constant to the number of machines.

4 Our Contributions

In this paper, we investigate whether it is possible to construct communication-efficient (in terms of *CC-3*) algorithms for distributed optimization problems. By taking state-of-the-art techniques for handling similarity, compression and local steps into a single method, we bridge the existing gap and design a SOTA in the sense of *CC-3*.

1. **Combination of compression and similarity.** It can be seen from Table 1 that there are methods that combine similarity with either compression or acceleration – but not both techniques at the same time. However, there was no success in combining all three approaches into a single algorithm. *We propose **the first** accelerated method for **both** unbiased and biased compression under similarity condition*.
2. **Best communication time**. To the best of our knowledge, in the case of $\delta \ll L$, depending on the number of machines, either Accelerated ExtraGradient or Three Pillars Algorithm (Beznosikov, Takác, and Gasnikov 2024) gives the top *CC-3* result. It can be seen from Table 1 that our OLGA *significantly **dominates** both methods*.
3. **Numerical experiments**. The constructed theory is verified experimentally on different problems. Experiments confirm the superiority of our method and show its robustness to changes in the number of computational nodes and different values of the problem constants.

5 Problem Formulation and Assumptions

Real problems arising in practice are known to behave better than in theory. Therefore, constructing methods for idealized setups can be useful (Woodworth, Mishchenko, and Bach 2023). Our work relies on the strong convexity of the mean risk and the homogeneity of the data on all nodes.

Method	Approach	Communication time (*CC-3*)	Weaknesses
ADIANA	Unbiased compression Acceleration	$\mathcal{O}\left(\left(1+\left(\omega^{-1}+M^{-1/2}\right)\sqrt{\frac{L}{\mu}}\right)\log\frac{1}{\varepsilon}\right)$	Only unbiased compressor All nodes are involved in communication round Does not account for similarity
ECLK	Biased compression Acceleration	$\mathcal{O}\left(\left(1+\beta^{1/2}\sqrt{\frac{L}{\mu}}\right)\log\frac{1}{\varepsilon}\right)$	Bad constants in estimation All nodes are involved in communication round Does not account for similarity
LoCoDL	Unbiased compression Local steps Acceleration	$\mathcal{O}\left(\left(1+(\omega^{-1}+\omega^{-1/2})\sqrt{\frac{L}{\mu}}\right)\log\frac{1}{\varepsilon}\right)$	Does not account for similarity All nodes are involved in communication round
TAMUNA	Unbiased compression Client sampling Local steps	$\mathcal{O}\left(\left(M+\sqrt{\frac{L}{\mu}}\right)\log\frac{1}{\varepsilon}\right)$	Does not account for similarity
AccExtraGradient	Similarity Local steps Acceleration	$\mathcal{O}\left(\sqrt{\frac{\delta}{\mu}}\log\frac{1}{\varepsilon}\right)$	All nodes are involved in communication round Communication is inefficient
AccSVRS	Similarity Client sampling Local steps Acceleration	$\mathcal{O}\left(\left(M+M^{3/4}\sqrt{\frac{\delta}{\mu}}\right)\log\frac{1}{\varepsilon}\right)$	Communication is inefficient
Optimistic MASHA	Similarity Unbiased compression	$\mathcal{O}\left(1+\left(M^{-1}\frac{L}{\mu}+M^{-1/2}\frac{\delta}{\mu}\right)\log\frac{1}{\varepsilon}\right)$	All nodes are involved in communication round Lipschitz constant is included in the estimation No acceleration Only permutation compressor
Three Pillars Algorithm	Unbiased compression Similarity Local steps	$\mathcal{O}\left(\left(1+M^{-1/2}\frac{\delta}{\mu}\right)\log\frac{1}{\varepsilon}\right)$	All nodes are involved in communication round No acceleration Only permutation compressor
OLGA	Similarity Unbiased compression Local steps Acceleration	$\mathcal{O}\left(\left(1+\left[M^{-1/4}+\omega^{-1/2}\right]\sqrt{\frac{\delta}{\mu}}\right)\log\frac{1}{\varepsilon}\right)$	Only unbiased compressor All nodes are involved in communication round
EF-OLGA	Similarity Biased compression Local steps Acceleration	$\mathcal{O}\left(\left(1+\beta^{1/4}\sqrt{\frac{\delta}{\mu}}\right)\log\frac{1}{\varepsilon}\right)$	All nodes are involved in communication round

Table 1: Summary of different communication complexities of SOTAs for distributed optimization.
Notation: $\omega, \beta =$ compression constants, $M =$ number of computational nodes, $\delta =$ similarity (relatedness) constant, $\mu =$ constant of strong convexity of the objective f, $L =$ Lipschitz constant of the gradient of f.

Assumption 1. *Every f_m is δ-related to f (Definition 3) and $f\colon \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex on $\mathbb{R}^d$:*

$$f(x) \geq f(y) + \langle \nabla f(y), x-y\rangle + \frac{\mu}{2}\|x-y\|^2, \quad \forall x, y \in \mathbb{R}^d. \tag{2}$$

Note that Assumption 1 allows local functions to be nonconvex. The only requirement imposed on them is the Hessian similarity. We consider the distributed optimization problem of the form (1) , where the data is drawn from a single distribution. It is also worth noting that Definition 3 implies δ-smoothness of $f_m - f$ for all $m \in [1, M]$:

$$\|\nabla(f_m - f)(x) - \nabla(f_m - f)(y)\|^2 \leq \delta^2\|x-y\|^2, \quad \forall x, y \in \mathbb{R}^d. \tag{3}$$

In fact, the Hessian similarity assumption does not dramatically reduce the generality of our analysis. Indeed, if the data is heterogeneous, it is sufficient to put $\delta = L$ in our results.

6 Unbiased Compression via OLGA

As mentioned above, compression can be implemented through variance reduction (Qian, Richtárik, and Zhang 2021; Gorbunov et al. 2021; Beznosikov et al. 2022). The approach proposed in (Lin et al. 2024) provides a way to construct an optimal method for the variance reduction technique under the similarity condition. The idea is to try to naturally generalize this approach to schemes with compression. We first consider an unbiased compressor (Definition 1).

In Algorithm 1, the full gradient is called only once per iteration before entering the loop (Line 4). At each iteration, it is required to communicate with each node and solve the subproblem (Line 12 of Algorithm 1). The number of iterations of Algorithm 1 is a random variable (see Line 2) that is not bounded from above, and hence the effectiveness of communication is important. Algorithm 1 has no acceleration. We build it to use as an integral part of more efficient Algorithm 2. From the point of view of theory, we are only interested in the descent lemma for Algorithm 1.

Lemma 1. *Consider an epoch of Algorithm 1. Let $h(x) = f_1(x) - f(x) + \frac{1}{2\theta}\|x\|^2$, where $\theta \leq \min\left\{\frac{\sqrt{p}\sqrt{M}}{8\delta\sqrt{\omega}}, \frac{1}{2\delta}\right\}$. Then the following inequality holds for every $x \in \mathbb{R}^d$:*

$$\mathbb{E}\left[f(x_N) - f(x)\right] \leq \mathbb{E}\Big[\langle x - x_0, \nabla h(x_N) - \nabla h(x_0)\rangle - \frac{p}{2}D_h(x_0, x_N) - \frac{\mu}{2}\|x_N - x\|^2\Big]. \tag{4}$$

Algorithm 1

Input: $x_0 \in \mathbb{R}^d, p \in (0,1), \theta > 0$
Set $N \sim \text{Geom}(p)$
Send x_0 and $\nabla f_1(x_0)$ to each device
Collect $\nabla f(x_0) = \frac{1}{M}\sum_{m=1}^{M} \nabla f_m(x_0)$ on server
for $k = 0, 1, 2, \ldots, N-1$ **do**:
    for each device m in parallel **do**:
        Calculate $\hat{g}_k^m$ using formula:

$$\hat{g}_k^m = \nabla f_m(x_k) - \nabla f_1(x_k) - \nabla f_m(x_0) + \nabla f_1(x_0)$$

        Send $g_k^m = Q\left(\hat{g}_k^m\right)$ to server
    end for
    Collect $g_k = \frac{1}{M}\sum_{m=1}^{M} g_k^m$ on server
    Calculate $t_k = g_k - \nabla f_1(x_0) + \nabla f(x_0)$
    Update $x_{k+1} = \arg\min_{x \in \mathbb{R}^d} q(x)$, where

$$q(x) = \langle t_k, x - x_k \rangle + \frac{1}{2\theta}\|x - x_k\|^2 + f_1(x)$$

    Send x_{k+1} and $\nabla f_1(x_{k+1})$ to each device
end for
Output: x_N

Next, we employ interpolation framework motivated by KatyushaX (Allen-Zhu 2018b) to obtain the final version of proposed algorithm. The key difference between our approach and KatyushaX is the choice of a suitable Bregman divergence as a metric function instead of the Euclidean distance used in the mentioned method.

Note that Line 9 of Algorithm 2 can be solved analytically and does not require expensive computations. The full gradient is computed twice per iteration, whereas without compression it could be invoked potentially infinitely often, resulting in significant communication costs. Algorithm 2 calls the full gradient a second time on the outer iteration. OLGA calls the full gradient a constant number of times.

Algorithm 2: OLGA

Input: $z_0 = y_0 \in \mathbb{R}^d, p \in (0,1), \theta > 0, \tau \in (0,1), \alpha > 0$
for $k = 0, 1, 2, \ldots, K-1$ **do**:
    Update $x_{k+1} = \tau z_k + (1-\tau) y_k$
    Update $y_{k+1} = \text{Alg.1}(x_{k+1}, \theta, p)$
    Send y_{k+1} to each device
    Collect $\nabla f(y_{k+1}) = \frac{1}{M}\sum_{m=1}^{M} \nabla f_m(y_{k+1})$ on server
    Calculate $t_k = \nabla(f_1 - f)(x_{k+1}) - \nabla(f_1 - f)(y_{k+1})$
    Calculate $G_{k+1} = p\left(t_k + \frac{x_{k+1} - y_{k+1}}{\theta}\right)$
    Update $z_{k+1} = \arg\min_{z \in \mathbb{R}^d} q(z)$, where

$$q(z) = \frac{1}{2\alpha}\|z - z_k\|^2 + \langle G_{k+1}, z \rangle + \frac{\mu}{2}\|z - y_{k+1}\|^2$$

end for
Output: y_K

For the sake of brevity of description, we introduce two potential functions:

$$Y_k = \frac{\alpha}{\tau}[f(y_k) - f(x_*)], \quad Z_k = \frac{1 + \mu\alpha}{2}\|z_k - x_*\|^2,$$

where x_* is the solution of the problem (1).

Theorem 1. *Let the problem (1) be solved by Algorithm 2 with* $\theta \leq \min\left\{\frac{\sqrt{p}\sqrt{M}}{8\delta\sqrt{\omega}}, \frac{1}{2\delta}\right\}$ *and tuning parameters such that* $4\alpha p\tau \leq \theta$. *Then the following inequality holds:*

$$\mathbb{E}\left[Y_{k+1} + Z_{k+1}\right] \leq \mathbb{E}\left[(1-\tau)Y_k + (1+\mu\alpha)^{-1} Z_k\right].$$

As discussed above, the iteration of Algorithm 2 invokes the full gradient twice and the compressed gradient another N times. N is a random variable depending on the parameter p. On average, we have $\mathcal{O}\left(1/\gamma_\omega + p\right)$ *CC-3* for a single iteration of OLGA. It is obvious that one should choose $p = 1/\gamma_\omega$. It has been discussed above that for practical compressors $\gamma_\omega \geq \omega$ holds. Let us select α, τ values more carefully and formulate the following corollary.

Corollary 1. *Let the problem (1) be solved by Algorithm 2. Choose*

$$p = \frac{1}{\gamma_\omega}, \quad \theta \leq \min\left\{\frac{\sqrt{p}\sqrt{M}}{8\delta\sqrt{\omega}}, \frac{1}{2\delta}\right\},$$

$$\tau = \min\left\{\frac{\sqrt{\mu}\theta^{1/2}p^{-1/2}}{4}, \frac{1}{4}\right\}, \quad \alpha = \frac{\theta p^{-1}}{8\tau},$$

then Algorithm 2 has

$$\tilde{\mathcal{O}}\left(\gamma_\omega + \sqrt{\frac{\delta}{\mu}}\left[\gamma_\omega M^{-1/4} + \gamma_\omega^{1/2}\right]\right) \text{ CC-1},$$

$$\tilde{\mathcal{O}}\left(M\gamma_\omega + \sqrt{\frac{\delta}{\mu}}\left[\gamma_\omega M^{3/4} + M\gamma_\omega^{1/2}\right]\right) \text{ CC-2},$$

and

$$\tilde{\mathcal{O}}\left(1 + \sqrt{\frac{\delta}{\mu}}\left[M^{-1/4} + \gamma_\omega^{-1/2}\right]\right) \text{ CC-3}.$$

It is worth noting that in *CC-3* if γ_ω is too large, the term with M dominates and the communication time result stops improving. If γ_ω is too small, the compression effect is not as strong as it could be. Therefore, the best effect is given by $\gamma_\omega = \Theta\left(\sqrt{M}\right)$. To simplify the appearance of the obtained results, the complexities are shown in Table 1 under consideration that $\gamma_\omega \sim \omega$. This holds for a number of common used compressors (Beznosikov et al. 2023; Alistarh et al. 2018).

6.1 Discussion

Let us compare Algorithm 2 with distributed optimization SOTAs. It makes sense to make comparisons only with methods that utilize similarity, because superiority over other ones depends mainly on how much δ is less than L. The optimal choice $\gamma_\omega = \Theta\left(\sqrt{M}\right)$ is assumed below.

1. **`Accelerated ExtraGradient`** outperforms our method in the sense of *CC-1* and *CC-2*. However, `OLGA` has $\tilde{\mathcal{O}}\left(1 + M^{-1/4}\sqrt{\delta/\mu}\right)$ *CC-3* vs. $\tilde{\mathcal{O}}\left(\sqrt{\delta/\mu}\right)$ for its competitor and hence turns out to be better by a factor $M^{1/4}$. Thus, the difference in the running time of methods with a substantially large number of machines is enormous.
2. **`AccSVRS`** loses to our method in terms of *CC-1*: $\tilde{\mathcal{O}}\left(M^{1/2} + M^{1/4}\sqrt{\delta/\mu}\right)$ vs. $\tilde{\mathcal{O}}\left(M + M^{3/4}\sqrt{\delta/\mu}\right)$; and *CC-3*: $\tilde{\mathcal{O}}\left(1 + M^{-1/4}\sqrt{\delta/\mu}\right)$ vs. $\tilde{\mathcal{O}}\left(M + M^{3/4}\sqrt{\delta/\mu}\right)$; but wins by *CC-2* due to client sampling.
3. **`Three Pillars Algorithm`** loses to our method by *CC-1*: $\tilde{\mathcal{O}}\left(M^{1/2} + M^{1/4}\sqrt{\delta/\mu}\right)$ vs. $\mathcal{O}\left(M + M^{1/2}\cdot\delta/\mu\right)$; and *CC-2*: $\tilde{\mathcal{O}}\left(M^{3/2} + M^{5/4}\sqrt{\delta/\mu}\right)$ vs. $\mathcal{O}\left(M^2 + M^{3/2}\cdot\delta/\mu\right)$. In terms of *CC-3*, it has a better $M^{-1/2}$ factor. However, `Three Pillars Algorithm` is not accelerated method. Thus, it loses to `OLGA` in a wide range of practical problems.

7 Biased Compression via `EF-OLGA`

In this section, we consider a biased compressor (Definition 2). As noted above, biased compressors are difficult to analyze and hence compressing gradient differences without introducing additional sequences will not yield results. Here we have to add "error" terms e_k^m.

Algorithm 3

Input: $x_0 \in \mathbb{R}^d, p \in (0,1), \theta > 0, e_0^m = 0$
Set $N \in \text{Geom}(p)$
Send x_0 and $\nabla f_1(x_0)$ to each device
Collect $\nabla f(x_0) = \frac{1}{M}\sum_{m=1}^M \nabla f_m(x_0)$ on server
for $k = 0, 1, 2, \ldots, N-1$ **do**:
  for each device m in parallel **do**:
    Calculate $\hat{g}_k^m$ using formula:

$$\hat{g}_k^m = \nabla f_m(x_k) - \nabla f_1(x_k) - \nabla f_m(x_0) + \nabla f_1(x_0)$$

    Send $g_k^m = C\{e_k^m + \theta\hat{g}_k^m\}$ to server
    Update $e_{k+1}^m = e_k^m - g_k^m + \theta\hat{g}_k^m$
    if $k = N-1$ **then**:
      Send $e_{k+1}^m = e_N^m$ to server
    end if
  end for
  Collect $g_k = \frac{1}{M}\sum_{m=1}^M g_k^m$ on server
  Calculate $t_k = \frac{1}{\theta}g_k - \nabla f_1(x_0) + \nabla f(x_0)$
  Update $x_{k+1} = \arg\min_{x\in\mathbb{R}^d} q(x)$, where

$$q(x) = \langle t_k, x - x_k\rangle + \frac{1}{2\theta}\|x - x_k\|^2 + f_1(x)$$

  Send x_{k+1} and $\nabla f_1(x_{k+1})$ to each device
end for
Output: x_N, $\frac{1}{M}\sum_{m=1}^M e_N^m$

Algorithm 4: `EF-OLGA`

Input: $z_0 = y_0 \in \mathbb{R}^d, p \in (0,1), \theta > 0, \tau \in (0,1), \alpha > 0$ and $e_m^0 \in \mathbb{R}^d$ for every $m \in [1, M]$
for $k = 0, 1, 2, \ldots, K-1$ **do**:
  Update $x_{k+1} = \tau z_k + (1-\tau)y_k$
  Update $y_{k+1}, e_{k+1} = \text{Alg.3}(x_{k+1}, \theta, p)$
  Send y_{k+1} to each device
  Collect $\nabla f(y_{k+1}) = \frac{1}{M}\sum_{m=1}^M \nabla f(y_{k+1})$ on server
  Calculate $t_k = \nabla(f_1 - f)(x_{k+1}) - \nabla(f_1 - f)(y_{k+1})$
  Calculate $G_{k+1} = p\left(t_k + \frac{x_{k+1} - \tilde{y}_{k+1}}{\theta}\right)$, where

$$\tilde{y}_{k+1} = y_{k+1} - e_{k+1}$$

  Update $z_{k+1} = \arg\min_{z\in\mathbb{R}^d} q(z)$, where

$$q(z) = \frac{1}{2\alpha}\|z - z_k\|^2 + \langle G_{k+1}, z\rangle + \frac{\mu}{2}\|z - y_{k+1}\|^2$$

end for
Output: y_K

Algorithm 3 is obtained by combining Algorithm 1 and error feedback framework (Seide et al. 2014). It is proposed to introduce an additional sequence e_k^m at each node m that will "remember" how much the gradient sent to the server differs from the true gradient computed on the machine (Line 9). Unlike the analysis in the previous section, it is not possible to introduce the appropriate metric immediately. After extensive analysis of virtual sequences, the Bregman divergence of the smooth function on "real" arguments and the Euclidean distance on "virtual" ones appear independently. This requires some manipulation to analyze correctly.

Note that Line 8 of Algorithm 4 utilizes "error" terms from the last iteration of Algorithm 3. Without such a modification, variance reduction can not be performed. Thus, in the biased case, a stronger connection between the inner and outer algorithms is formed. As in the case of Algorithm 2, we also do not sample the stochastic gradient at the outer iteration because it does not affect the result and complicates the analysis. Therefore, `EF-OLGA` calls the full gradient twice per iteration.

Theorem 2. *Let the problem (1) be solved by Algorithm 4 with $\theta \le \frac{p^{3/2}}{24\delta} \le \frac{1}{6\delta}$ and tuning parameters such that $28\alpha p\tau \le \theta$. Then the following inequality holds:*

$$\mathbb{E}\left[Y_{k+1} + Z_{k+1}\right] \le \mathbb{E}\left[(1-\tau)Y_k + (1+\mu\alpha)^{-1}Z_k\right].$$

Again, the *CC-3* of the iteration is $\mathcal{O}\left(\frac{1}{\gamma_\beta} + p\right)$. Thus, $p = 1/\gamma_\beta$.

Corollary 2. *Let the problem (1) be solved by Algorithm 4. Choose*

$$p = \frac{1}{\gamma_\beta}, \quad \tau = \min\left\{\frac{\theta^{1/2}p^{-1/2}}{18}, \frac{1}{18}\right\},$$

$$\theta \le \frac{p^{3/2}}{12\delta}, \quad \alpha = \frac{\theta p^{-1}}{36\tau},$$

then Algorithm 4 has

$$\tilde{\mathcal{O}}\left(\gamma_\beta + \gamma_\beta^{5/4}\sqrt{\frac{\delta}{\mu}}\right), \quad \tilde{\mathcal{O}}\left(M\gamma_\beta + M\gamma_\beta^{5/4}\sqrt{\frac{\delta}{\mu}}\right),$$

and

$$\tilde{\mathcal{O}}\left(1 + \gamma_\beta^{1/4}\sqrt{\frac{\delta}{\mu}}\right)$$

CC-1, CC-2 and CC-3, respectively.

This corollary repeats the proof of Corollary 1 with other constants.

8 Numerical Experiments

Our theoretical findings are confirmed numerically on various tasks. In particular, we consider the ridge regression problem (McDonald 2009):

$$f(x) = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{n}\sum_{j=1}^{n}\left(\langle x, a_j^m\rangle - b_j^m\right)^2 + \lambda\|x\|^2, \quad (5)$$

and the logistic regression problem:

$$f(x) = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{n}\sum_{j=1}^{n}\ln\left(1 + e^{-b_j^m\langle x, a_j^m\rangle}\right) + \lambda\|x\|^2. \quad (6)$$

We set the penalty parameter λ to $L/100$, where L is a Lipschitz constant of the gradient of the main objective. Also, we consider different datasets from LibSVM (Chang and Lin 2011): `mushrooms` and `a9a` (Appendix). Since we consider illustrative experiments with linear models, it is not difficult to calculate μ, L, δ. Therefore, the parameters of algorithms are chosen in the same way as in the theory without any tuning. We also vary the number of workers M. As competitors we take state-of-the-art methods from Table 1: `AccSVRS`, `Accelerated ExtraGradient`, `ADIANA`, `LoCoDL`. For algorithms with compression, we use a random sparsification operators Rand$K$, where we vary the number of coordinates K (and hence ω since $\omega = d/K$).

See the experiments with `OLGA` on `mushrooms` dataset on Figure 1. The rest ones can be found in the supplementary material.

9 Conclusion

In this paper, we pioneered two algorithms, `OLGA` and `EF-OLGA`, which close the gap between accelerated methods utilizing similarity and accelerated methods exploiting unbiased/biased compression. In the case of unbiased compressors, constructed theory guarantees an advantage over SOTAs in terms of communicated time. Numerical experiments supports our theoretical insights. Nevertheless, a number of questions remain for further research. It is also worth noting that for biased compressor, the communication complexity lower bound under similarity condition is unknown: can we design an accelerated algorithm that enjoys a better complexity, or is the estimate we received a lower bound?

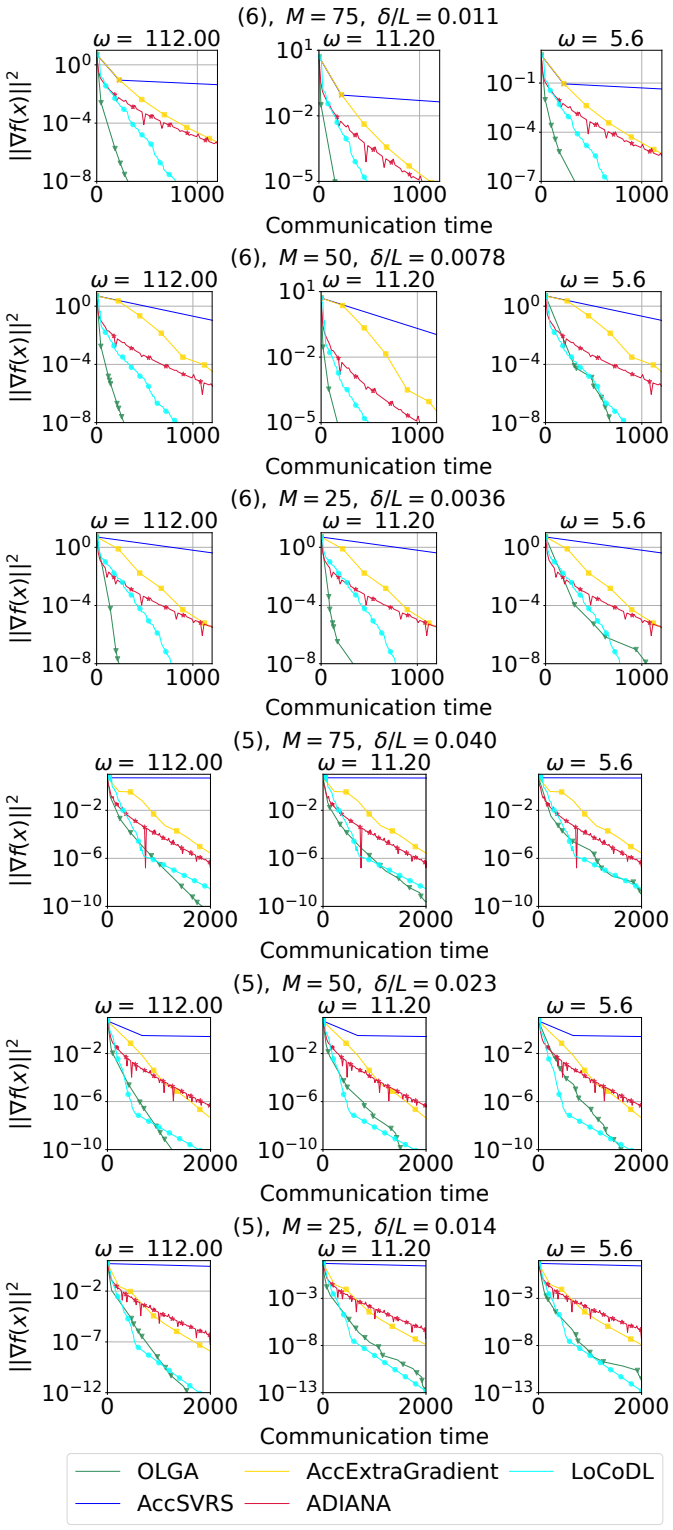

Figure 1: Comparison of state-of-the-art distributed methods. The comparison is made on `mushrooms` dataset. The criterion is the communication time (*CC-3*). We vary the ratio δ/L, the number of nodes M and the power of compression ω.

Acknowledgments

The work was done in the Laboratory of Federated Learning Problems of the ISP RAS (Supported by Grant App. No. 2 to Agreement No. 075-03-2024-214).

References

Alacaoglu, A.; and Malitsky, Y. 2022. Stochastic variance reduction for variational inequality methods. In *Conference on Learning Theory*, 778–816. PMLR.

Alistarh, D.; Grubic, D.; Li, J.; Tomioka, R.; and Vojnovic, M. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. *Advances in neural information processing systems*, 30.

Alistarh, D.; Hoefler, T.; Johansson, M.; Konstantinov, N.; Khirirat, S.; and Renggli, C. 2018. The convergence of sparsified gradient methods. *Advances in Neural Information Processing Systems*, 31.

Allen-Zhu, Z. 2018a. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(221): 1–51.

Allen-Zhu, Z. 2018b. Katyusha x: Practical momentum method for stochastic sum-of-nonconvex optimization. *arXiv preprint arXiv:1802.03866*.

Alqahtani, S.; and Demirbas, M. 2019. Performance analysis and comparison of distributed machine learning systems. *arXiv preprint arXiv:1909.02061*.

Arjevani, Y.; and Shamir, O. 2015. Communication complexity of distributed convex learning and optimization. *Advances in neural information processing systems*, 28.

Bai, H. 2022. Modern distributed data-parallel large-scale pre-training strategies for nlp models. In *Proceedings of the 6th International Conference on High Performance Compilation, Computing and Communications*, 44–53.

Beznosikov, A.; Horváth, S.; Richtárik, P.; and Safaryan, M. 2023. On biased compression for distributed learning. *Journal of Machine Learning Research*, 24(276): 1–50.

Beznosikov, A.; Richtárik, P.; Diskin, M.; Ryabinin, M.; and Gasnikov, A. 2022. Distributed methods with compressed communication for solving variational inequalities, with theoretical guarantees. *Advances in Neural Information Processing Systems*, 35: 14013–14029.

Beznosikov, A.; Scutari, G.; Rogozin, A.; and Gasnikov, A. 2021. Distributed saddle-point problems under data similarity. *Advances in Neural Information Processing Systems*, 34: 8172–8184.

Beznosikov, A.; Takác, M.; and Gasnikov, A. 2024. Similarity, compression and local steps: three pillars of efficient communications for distributed variational inequalities. *Advances in Neural Information Processing Systems*, 36.

Bidollahkhani, M.; and Kunkel, J. M. 2024. Revolutionizing System Reliability: The Role of AI in Predictive Maintenance Strategies. *arXiv preprint arXiv:2404.13454*.

Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3): 1–27.

Condat, L.; Agarskỳ, I.; Malinovsky, G.; and Richtárik, P. 2023. Tamuna: Doubly accelerated federated learning with local training, compression, and partial participation. *arXiv preprint arXiv:2302.09832*.

Condat, L.; Maranjyan, A.; and Richtárik, P. 2024. LoCoDL: Communication-Efficient Distributed Learning with Local Training and Compression. *arXiv preprint arXiv:2403.04348*.

Defazio, A.; Bach, F.; and Lacoste-Julien, S. 2014. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27.

Gorbunov, E.; Burlachenko, K. P.; Li, Z.; and Richtárik, P. 2021. MARINA: Faster non-convex distributed learning with compression. In *International Conference on Machine Learning*, 3788–3798. PMLR.

Gorbunov, E.; Hanzely, F.; and Richtárik, P. 2020. A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, 680–690. PMLR.

He, Y.; Huang, X.; and Yuan, K. 2024. Unbiased compression saves communication in distributed optimization: when and how much? *Advances in Neural Information Processing Systems*, 36.

Hendrikx, H.; Xiao, L.; Bubeck, S.; Bach, F.; and Massoulie, L. 2020. Statistically preconditioned accelerated gradient method for distributed optimization. In *International conference on machine learning*, 4203–4227. PMLR.

Johnson, R.; and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26.

Jordan, M. I.; Lee, J. D.; and Yang, Y. 2019. Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2): 1–210.

Khirirat, S.; Feyzmahdavian, H. R.; and Johansson, M. 2018. Distributed learning with compressed gradients. *arXiv preprint arXiv:1806.06573*.

Konečnỳ, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Kovalev, D.; Beznosikov, A.; Borodich, E.; Gasnikov, A.; and Scutari, G. 2022. Optimal gradient sliding and its application to optimal distributed optimization under similarity. *Advances in Neural Information Processing Systems*, 35: 33494–33507.

Kovalev, D.; Horváth, S.; and Richtárik, P. 2020. Don't jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop. In *Algorithmic Learning Theory*, 451–467. PMLR.

Li, Z.; Kovalev, D.; Qian, X.; and Richtárik, P. 2020. Acceleration for compressed gradient descent in distributed and federated optimization. *arXiv preprint arXiv:2002.11364*.

Lin, D.; Han, Y.; Ye, H.; and Zhang, Z. 2024. Stochastic distributed optimization under average second-order similarity: Algorithms and analysis. *Advances in Neural Information Processing Systems*, 36.

Lu, H.; Freund, R. M.; and Nesterov, Y. 2018. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1): 333–354.

McDonald, G. C. 2009. Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1): 93–100.

Mishchenko, K.; Gorbunov, E.; Takáč, M.; and Richtárik, P. 2019. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*.

Qian, X.; Richtárik, P.; and Zhang, T. 2021. Error compensated distributed SGD can be accelerated. *Advances in Neural Information Processing Systems*, 34: 30401–30413.

Richtárik, P.; Sokolov, I.; and Fatkhullin, I. 2021. EF21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems*, 34: 4384–4396.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Seide, F.; Fu, H.; Droppo, J.; Li, G.; and Yu, D. 2014. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Interspeech*, volume 2014, 1058–1062. Singapore.

Shamir, O.; Srebro, N.; and Zhang, T. 2014. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, 1000–1008. PMLR.

Stich, S. U.; and Karimireddy, S. P. 2019. The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*.

Sun, H.; Shao, Y.; Jiang, J.; Cui, B.; Lei, K.; Xu, Y.; and Wang, J. 2019. Sparse gradient compression for distributed SGD. In *International Conference on Database Systems for Advanced Applications*, 139–155. Springer.

Tian, Y.; Scutari, G.; Cao, T.; and Gasnikov, A. 2022. Acceleration in distributed optimization under similarity. In *International Conference on Artificial Intelligence and Statistics*, 5721–5756. PMLR.

Verbraeken, J.; Wolting, M.; Katzy, J.; Kloppenburg, J.; Verbelen, T.; and Rellermeyer, J. S. 2020. A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2): 1–33.

Vogels, T.; Karimireddy, S. P.; and Jaggi, M. 2019. PowerSGD: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32.

Woodworth, B.; Mishchenko, K.; and Bach, F. 2023. Two losses are better than one: Faster optimization using a cheaper proxy. In *International Conference on Machine Learning*, 37273–37292. PMLR.

Wu, Y.; Zeng, D.; Wang, Z.; Shi, Y.; and Hu, J. 2022. Distributed contrastive learning for medical image segmentation. *Medical Image Analysis*, 81: 102564.

Yuan, X.-T.; and Li, P. 2020. On convergence of distributed approximate newton methods: Globalization, sharper bounds and beyond. *Journal of Machine Learning Research*, 21(206): 1–51.

Zhang, Y.; and Lin, X. 2015. Disco: Distributed optimization for self-concordant empirical loss. In *International conference on machine learning*, 362–370. PMLR.

Zhu, W.; Luo, J.; and White, A. D. 2022. Federated learning of molecular properties with graph neural networks in a heterogeneous setting. *Patterns*, 3(6).