Learning Under Delayed Feedback: Implicitly Adapting to Gradient Delays

Rotem Zamir Aviv [1] Ido Hakimi [2] Assaf Schuster [2] Kfir Y. Levy [1 3]

Abstract

We consider stochastic convex optimization problems, where several machines act asynchronously in parallel while sharing a common memory. We propose a robust training method for the constrained setting and derive non asymptotic convergence guarantees that do not depend on prior knowledge of update delays, objective smoothness, and gradient variance. Conversely, existing methods for this setting crucially rely on this prior knowledge, which render them unsuitable for essentially all shared-resources computational environments, such as clouds and data centers. Concretely, existing approaches are unable to accommodate changes in the delays which result from dynamic allocation of the machines, while our method implicitly adapts to such changes.

1. Introduction

The past decade has witnessed a wide adoption of machine learning (ML) techniques in various fields. However, the underlying model complexity and the vast amount of data required to train modern ML models may lead to impractical prolonged training time. Parallelizing the learning process has the potential to greatly decrease the learning duration, by exploiting more computation power at every step.

Stochastic Gradient Descent (SGD) and its variants are amongst the most popular training methods in ML, and are known to work well even for modern large scale problems. However, due to the sequential nature of SGD, extending its usage for distributed learning is not straightforward.

Distributed learning methods with parameter-servercan be divided into synchronous Vs. asynchronous approaches. In the synchronous setting, all of the machines communicate at the same time and depend on one another to proceed. Such methods are easier to analyze, yet their performance depends on the slowest machine and they require large communication overheads. Conversely, in the asynchronous case, machines may communicate independently of other machines, which allows more flexibility and reduces communication. It is well known that the performance of asynchronous methods degrades due to the staleness of the gradient updates, i.e., *the delay between the current model, and the (outdated) model for which the gradient feedback is computed.*

The asynchronous setting was extensively investigated in the context of stochastic convex optimization, which captures fundamental learning problems like linear regression, logistic regression and SVMs. It was shown that if the learning objective is non-smooth, delays in the computed gradient feedback necessarily degrade the performance of SGD compared to the non-delayed setting, see e.g. (Joulani et al., 2013; Nedić et al., 2001).

Nevertheless, it was recently shown that the latter does not apply for smooth objectives. Arjevani et al. (2020) were the first to obtain the optimal rates for SGD with delays in the smooth convex case. They have provided a version of delayed SGD that incurs no degradation compared to the non-delayed case as long as the maximal delay $\tau_{\max}$ is small enough: $\tau_{\max} \leq O(\sqrt{T})$, where T is the total number of SGD updates. In the strongly-convex case they have shown that there is no degradation as long as $\tau_{\max} \leq O(T/\log T)$. While the result of Arjevani et al. (2020) was limited to quadratic objectives, in a very recent work, Stich & Karimireddy (2020) have generalized their result for the general convex smooth case, while obtaining the same optimal guarantees.

Unfortunately, the optimal methods of Arjevani et al. (2020) and Stich & Karimireddy (2020) **(i)** do not hold for constrained problems, **(ii)** degrade with the maximal delay, which might be substantially higher than the average delay, and are not able to accommodate changes in the delays throughout the training process, and **(iii)** require prior knowledge of the maximal delay and smoothness parameter. In practice, when working with shared resources computational systems, delays might vary with time due to dynamic allocation of machines. Such scalable systems are central to the entire discipline of distributed learning,

[1]Viterby Faculty of Electrical and Computer Engineering, Technion, Haifa, Israel [2]Taub Faculty of Computer Science, Technion, Haifa, Israel [3]A Viterbi Fellow. Correspondence to: Rotem Zamir Aviv <ratume@campus.technion.ac.il>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

when more and more computations are carried on remote machines.

Thus, we propose delay-adaptive and delay-robust algorithms for asynchronous stochastic convex optimization, with arbitrary time varying delays. *Delay-robust* means that one can tune a baseline (online) algorithm for a given model without delays, yet apply it for a delay incorporated model, without further tuning, while achieving the optimal delay dependency (or vice versa). By *delay-adaptive* we mean that our methods do not require any information regarding the delays, and are able to accommodate non-stationary changes in the delays.

Contributions: We summarize our contributions below,

- For the general convex smooth case we utilize a simple yet general approach to develop a *delay-adaptive* algorithm that obtains the optimal rates for the *delayed constrained* setting, thus resolving an open question posed by Stich & Karimireddy (2020). Our algorithm implicitly adapts to the objective smoothness and the gradient variance.
- For the strongly-convex smooth case we develop a *delay-adaptive* algorithm which obtains meaningful yet suboptimal guarantees for the delayed constrained setting, without prior knowledge of the smoothness parameter or the gradient variance. This is the first method for the delayed strongly-convex and *constrained* setting with non-trivial guarantees.
- We allow for arbitrary delays that may vary with time, with no further assumptions. Our algorithm implicitly adapts to changes in delays, which enhances its robustness and enables usage in scalable systems. In contrast to previous works on this topic, the performance of our method degrades *proportionally to the average delay rather than to the maximal one.*
- We validate the performance of our algorithm on real-world data. The experiments demonstrate our algorithm robustness and adaptivity. Concretely, when tuning SGD to train a given model for a specific delay regime and then changing the delay regime, its performance might degrade. Conversely, our algorithm maintains its high performance.

On the technical side, our work builds on a recent online to batch technique (Cutkosky, 2019; Kavis et al., 2019), that we combine with optimistic online learning techniques (Mohri & Yang, 2016; Rakhlin & Sridharan, 2013).

1.1. Related Work

There is a large volume of published studies considering distributed learning; we provide few closely related examples. A recent survey on this topic can be viewed e.g. in (Verbraeken et al., 2020; Ben-Nun & Hoefler, 2019).

Our focus here is on centralized distributed setting, where there is a single parameter-vector that is updated using information received from several parallel machines. The centralized case further divides into synchronous Vs. asynchronous approaches. Synchronous training methods usually employ large batches to compute gradient estimates, when the batch computation is distributed between the machines. This approach was extensively investigated e.g. in (Dekel et al., 2012; Cotter et al., 2011; Shalev-Shwartz & Zhang, 2013; Li et al., 2014; Takáč et al., 2015; Jain et al., 2016).

The centralized asynchronous case was investigated e.g. in (Bertsekas & Tsitsiklis, 1989; Agarwal & Duchi, 2012; Shamir & Srebro, 2014; McMahan & Streeter, 2014; Sra et al., 2015; Dutta et al., 2018). One line of work that has gained much interest in the asynchronous case is a model where the updates in the parameter-vector are performed with individual coordinate granularity (Recht et al., 2011; Leblond et al., 2018). It was shown that this approach is beneficial when data features are sparse.

Another line of work in the context of asynchronous training is to analyze SGD with delayed gradient feedbacks. This study was initiated by (Agarwal & Duchi, 2012), and followed by (McMahan & Streeter, 2014; Sra et al., 2015; Lian et al., 2015; Feyzmahdavian et al., 2016; Zheng et al., 2017; Dutta et al., 2018; Arjevani et al., 2020; Stich & Karimireddy, 2020) amongst others.

An optimal SGD variant for the delayed convex smooth case was first suggested by (Arjevani et al., 2020), providing an optimal convergence rate of $O(1/\sqrt{T} + \tau/T)$, where τ is some constant delay and T is the total number of gradient updates. They also proved an optimal rate of $O\left(1/T + \exp\left(\frac{-HT}{L\tau}\right)\right)$ for the strongly-convex and smooth case, where L and H are the smoothness and strong-convexity parameters of the objective. These bounds imply that we suffer no degradation compared to non-delayed SGD, as long as $\tau \le O(\sqrt{T})$ in the convex case, and $\tau \le O(T/\log T)$ in the strongly-convex case. While the method of (Arjevani et al., 2020) only applies for quadratic losses, the very recent work of (Stich & Karimireddy, 2020) has generalized these results to the general convex case and upper bounded delays $\tau \in [0, \tau_{\max}]$.

Finally, while previous papers assume upper bounded delays (Lian et al., 2015), (Agarwal & Duchi, 2012), (Zheng et al., 2017), in environments such as clouds and data centers this assumptions is often violated. To ensure scalability, the algorithm must accommodate changes in delays. Sra et al. (2015); McMahan & Streeter (2014); Dutta et al.

(2018); Ren et al. (2020); Zhou et al. (2018) have addressed this issue, though with either some additional assumptions on the delay distribution, suboptimal guarantees, or asymptotic guarantees.

The rest of the paper is organized as follows. The next section describes our problem formulation and introduces the concept of Online Convex Optimization. Section 3 offers our main Theorem for smooth non strongly convex objectives. Section 4 presents the main Theorem for smooth strongly convex objectives, and in Section 5 we detail our experimental setup and results. Complete proofs can be found in the supplement.

2. Problem Formulation

We focus on solving the following optimization problem,

$$\min_{\mathbf{w}\in\mathcal{K}} f(\mathbf{w}) \ ,$$

where $f : \mathcal{K} \to \mathbb{R}$ is a convex and smooth function, and $\mathcal{K}$ is a compact convex subset of $\mathbb{R}^d$.

We consider first-order iterative optimization algorithms with access to a stochastic gradient oracle that returns unbiased estimates of the objective gradients. In the standard non-delayed setup, in each iteration t the algorithm queries a noisy gradient oracle with a point $\mathbf{x}_t \in \mathcal{K}$, and in return it receives an unbiased estimate,

$$\mathbf{g}_t = \nabla f(\mathbf{x}_t) + \boldsymbol{\xi}_t \ \ \forall t \ ,$$

where $\mathbb{E}[\boldsymbol{\xi}_t|\mathbf{x}_t] = 0$. After T iterations the algorithm outputs an estimate $\mathbf{x}_T$ of the optimal solution, and its accuracy is measured by the expected excess loss,

$$\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{w}^*)] \ ,$$

where $\mathbf{w}^* \in \arg\min_{\mathbf{w}\in\mathcal{K}} f(\mathbf{w})$.

In this paper we consider centralized distributed asynchronous learning problems. Concretely, we assume a central server which maintains a global parameter-vector, and employs several workers in parallel to update that vector using first order information. Each worker queries the central server for the most updated parameter-vector, and computes its gradients with respect to that vector. The gradients are sent back to the central server, which then updates its parameter-vector accordingly. Since every worker communicates with the central server independently of the others, by the time a certain worker has restored the gradients to the server, another may have updated the parameter-vector. Thus, the server updates the model based on stale gradients.

This setting can be described as a first order stochastic optimization problem, yet with a *delayed noisy gradient oracle*. Now, when we query this oracle with $\mathbf{x}_t$, a stale gradient estimate of the objective at a previous query point $\mathbf{x}_{t-\tau_t}$ is received. We assume that the delays $\tau_t \in [0, t-1]$ may change arbitrarily and are unknown in advance. Similarly to (Arjevani et al., 2020), we describe the *delayed noisy gradient oracle* as follows,

$$\mathbf{g}_{t-\tau_t} = \nabla f(\mathbf{x}_{t-\tau_t}) + \boldsymbol{\xi}_t \ ,$$

where $\mathbb{E}[\boldsymbol{\xi}_t|\mathbf{x}_t] = 0$. In Appendix A we explain why $\mathbb{E}[\boldsymbol{\xi}_t|\mathbf{x}_t] = 0$ makes sense in the context of the asynchronous distributed setting. Note that in order for this assumption to hold for asynchronous stochastic optimization, all workers must have equal access to the data and the delays are assumed to be data-independent.

We make the following standard assumptions throughout the paper:

1. Bounded gradients. There exists a constant $G > 0$ such that
$$\|\nabla f(\mathbf{x})\| \le G \quad \forall \mathbf{x} \in \mathcal{K} \ .$$
2. Bounded variance. There exist a constant σ^2 such that
$$\mathbb{E}[\|\boldsymbol{\xi}_t\|^2|\mathbf{x}_t] \le \sigma^2 \quad \forall t \ .$$
3. Compact domain. There exist a constant D such that
$$\|\mathbf{x} - \mathbf{y}\|^2 \le D^2 \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{K} \ .$$
4. We assume $\tau_t \le t-1$, since the first computed gradient is $\mathbf{g}_1$.

Online Convex Optimization. Our results rely on Online Convex Optimization (OCO) techniques that we use as a mechanism for solving the aforementioned stochastic optimization problem. Next, we describe this setting and necessary definitions.

OCO problems can be depicted as a repeated game of T rounds. At every round $t \in [T]$ a learner makes a prediction $\mathbf{w}_t \in \mathcal{K}$, after which a convex loss function $f_t : \mathcal{K} \mapsto \mathbb{R}$ is chosen. Then, the learner incurs a loss $f_t(\mathbf{w}_t)$, and receives $f_t(\cdot)$ as a feedback. We assume that the losses $f_t(\cdot)$ may change arbitrarily, and may depend on the choices of the learner up to round t. Now, given a sequence of non-negative weights $\{\alpha_t\}_{t\in[T]}$, the goal of the learner is to minimize the (weighted) regret, which is defined below,

$$\mathrm{Reg}_T(\mathbf{w}^*) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{w}_t) - \min_{\mathbf{w}\in\mathcal{K}} \sum_{t=1}^{T} \alpha_t f_t(\mathbf{w}) \ . \qquad (1)$$

Note that if $\alpha_t = 1 \forall t$ this is the standard definition of regret. Oftentimes we assume that the learner can access

$f_t(\cdot)$ through a first order oracle, i.e., he may query a gradient oracle of $f_t(\cdot)$.

There is a strong connection between OCO and stochastic optimization. Concretely, if the losses $\{f_t(\cdot)\}_{t\in[T]}$ received by the OCO algorithm are unbiased estimates of a fixed function $f(\cdot)$, i.e., $f_t(\mathbf{w}) := f(\mathbf{w}; z_t)$ where z_t's are i.i.d. samples from some unknown distribution $\mathcal{D}$, then one can show the following online to batch conversion between regret guarantees and excess loss (Cesa-Bianchi et al., 2004),

$$\mathbb{E}\left[f\left(\frac{\sum_{t=1}^T \alpha_t \mathbf{w}_t}{\sum_{t=1}^T \alpha_t}\right) - f(\mathbf{w}^*)\right] \le \frac{\mathrm{Reg}_T(\mathbf{w}^*)}{\sum_{t=1}^T \alpha_t} \,,$$

where $f(\mathbf{w}) := \mathbb{E}_{z\sim\mathcal{D}}[f(\mathbf{w}; z)]$. This conversion allows usage of OCO methods for offline stochastic optimization, while maintaining the powerful framework of OCO and its strong guarantees. The next section shows how our results build on a recent novel online to batch conversion that is different than the standard one that we have just described.

Preliminaries

Definition 2.1. *Smoothness.* Function f is said to be smooth if and only if its gradient is L Lipschitz with some $L > 0$ over $\mathcal{K}$. Meaning,

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le L\|\mathbf{x} - \mathbf{y}\| \;\; \forall \mathbf{x}, \mathbf{y} \in \mathcal{K} \,.$$

Definition 2.2. *Strong convexity.* Function f is said to be H strongly-convex if and only if

$$f(\mathbf{x}) - f(\mathbf{y}) \le \nabla f(\mathbf{x})^\top(\mathbf{x}-\mathbf{y}) - \frac{H}{2}\|\mathbf{x}-\mathbf{y}\|^2 \;\; \forall \mathbf{x}, \mathbf{y} \in \mathcal{K} \,.$$

Definition 2.3. *Projection Operation.* Let $\Pi_{\mathcal{K}}(\cdot)$ denote the projection operation onto set $\mathcal{K}$. Namely,

$$\Pi_{\mathcal{K}}(\mathbf{y}) \triangleq \arg\min_{\mathbf{x}\in\mathcal{K}} \|\mathbf{x} - \mathbf{y}\| \,.$$

Notations. We denote $\alpha_{1:t} = \sum_{i=1}^t \alpha_i$, $\|\cdot\|$ as the Euclidean norm, and $[t] := \{1, \ldots, t\}$. For a series of delays $\tau_1, \cdots, \tau_T$ we denote their average by μ_τ and their variance by σ_τ, i.e.,

$$\mu_\tau := \frac{1}{T}\sum_{t=1}^T \tau_t \quad \& \quad \sigma_\tau^2 := \frac{1}{T}\sum_{t=1}^T \tau_t^2 - \mu_\tau^2 \,.$$

3. Delay Adaptive Scheme for General Convex Case

In this section we suggest two algorithms for the stochastic delayed setting, assuming that the objective $f(\cdot)$ is convex and smooth. In Section 3.1 we analyze a general scheme that enables to take any OCO algorithm and turn it to a delay-adaptive and delay-robust stochastic optimization algorithm. This scheme *does not require any prior knowledge of the delays* or their statistics, and furthermore it *does not require any tuning that depends on the delays*.

In Section 3.2 we describe and analyze a fully adaptive algorithm for the delayed setting. This algorithm obtains the optimal guarantees for this setting, while requiring the knowledge of neither the delays, nor any other problem parameters like noise variance, smoothness and gradient scale.

The difficulty of learning in the delayed setting stems from the difference between the stale gradient that we receive and the true gradient of the current iterate. This difference can be related directly to the degradation in the performance, where larger differences lead to worse performance. In the smooth case one can relate the difference between gradients to the difference between query points. Examining this difference for standard SGD with a fixed learning rate η shows that the distance between $\mathbf{w}_t$ and $\mathbf{w}_{t-\tau_t}$ can be bounded by $O(\tau_t\eta G)$ (using the update rule of SGD). Nevertheless, (optimizing over η) this bound is of the order of $O(G\sqrt{\tau_t/T})$, which leads to suboptimal performance.

Thus, in the context of the delayed setting, we would wish for an algorithm that employs *slowly changing query points*. Ideally, we would like to employ queries $\{\mathbf{x}_t\}_{t\in[T]}$ such that $\|\mathbf{x}_t - \mathbf{x}_{t-\tau_t}\| \le O(\tau_t/t)$, which is significantly smaller than the $O(\sqrt{\tau_t/T})$ that we have seen for standard SGD. Fortunately, it was recently demonstrated in (Cutkosky, 2019) and (Kavis et al., 2019) that it is possible to achieve slowly varying queries, while still maintain same guarantees as of standard SGD. They proposed an alternative approach to online to batch conversion, which queries the gradients at the *iterate averages* rather than the iterates themselves. When the objective is smooth, the use of such a conversion scheme stabilizes the predictions of the algorithm and thus leads to optimal convergence rates for the delayed setting. In other words, by querying the gradient oracle at the running averages, we implicitly adapt to gradient delays. We follow (Cutkosky, 2019; Kavis et al., 2019), and relate to this scheme as *anytime online to batch conversion.*

3.1. General Delay-Adaptive Scheme

In this section we analyze algorithm 1, which utilizes a general OCO algorithm $\mathcal{A}$, together with anytime online to batch conversion. Recall that in the standard online to batch scheme, gradients are queried at the iterates of the OCO algorithm. Conversely, in the anytime online to batch scheme that we utilize in algorithm 1, the OCO algorithm $\mathcal{A}$ produces iterates $\mathbf{w}_t$, while the gradients that $\mathcal{A}$ receives

Algorithm 1 Delay Adaptive Anytime Online to Batch

```
**Input:** # of iterations $T$ , $\mathbf{w}_1 \in \mathcal{K}$, weights $\{\alpha_t\}_{t\in[T]}$
**for** $t = 1$ **to** $T$ **do**
    $\mathbf{x}_t \leftarrow \frac{\sum_{i=1}^t \alpha_i \mathbf{w}_i}{\alpha_{1:t}}$
    get $\mathbf{g}_{t-\tau_t}$ from worker
    define $f_t(\mathbf{x}) = \mathbf{g}_{t-\tau_t}^\top \mathbf{x}$
    send $\alpha_t f_t(\mathbf{x})$ to $\mathcal{A}$
    get $\mathbf{w}_{t+1}$ from $\mathcal{A}$
**end for**
**return** $\mathbf{x}_T$
```

are queried at $\mathbf{x}_t$'s, which are weighted averages of the iterates.

By querying the gradients at the average we ensure slowly varying query points, even when the OCO iterates change rapidly. This allows us to receive general convergence guarantees for the delayed setting, compatible with any number of OCO algorithms, including such which are not tuned for the delayed setting. The result is stated in Theorem 3.1.

Theorem 3.1. *Assume that $f : \mathcal{K} \mapsto \mathbb{R}$ is L-smooth. Let $Reg_T(\mathbf{w}^*)$ be the regret of $\mathcal{A}$ with respect to the following sequence $\{f_t(\mathbf{x}) := \mathbf{g}_{t-\tau_t}^\top \mathbf{x}\}_{t\in[T]}$,*

$$Reg_T(\mathbf{w}^*) := \sum_{t=1}^T \alpha_t \mathbf{g}_{t-\tau_t}^\top (\mathbf{w}_t - \mathbf{w}^*) \ . \tag{2}$$

Then, using Alg. 1 with $\alpha_t = t$ ensures,

$$\mathbb{E}\left[f(\mathbf{x}_T) - f(\mathbf{w}^*)\right] = O\left(\frac{Reg_T(\mathbf{w}^*)}{T^2} + LD^2\frac{\mu_\tau}{T}\right) \ ,$$

where μ_τ is the average delay.

Theorem 3.1 implies that by using anytime online to batch conversion it is possible to convert any OCO algorithm into a delay-robust and delay-adaptive algorithm. In addition, while previous works obtained error bounds which are proportional to $O(\tau_{\max}/T)$, our bounds depend on $O(\mu_\tau/T)$ which might be substantially smaller.

Note that there are several standard algorithms that can be plugged into Alg. 1, such as OGD (Online Gradient Descent), and FTRL (Follow the Regularized Leader) (Zinkevich, 2003), (Shalev-Shwartz et al., 2011). For example, a delay adaptive version of SGD, can be applied by using OGD as $\mathcal{A}$ inside Alg. 1. The update rule in this case boils down to,

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{K}}(\mathbf{w}_t - \tilde{\eta}_t \mathbf{g}_{t-\tau_t}) \ . \tag{3}$$

where $\tilde{\eta}_t \propto \alpha_t/\sqrt{\sum_{i=1}^t \alpha_i^2} \approx 1/\sqrt{t}$, and $\mathbf{g}_{t-\tau_t}$ is a stale gradient of one of the past query points. Moreover, the query points $\{\mathbf{x}_t\}_{t\in[T]}$ are *weighted averages of past iterates; i.e.,* $\mathbf{x}_t = \sum_{i=1}^t \mathbf{w}_i/\alpha_{1:t}$. Note that in this case, $\tilde{\eta}_t$ does not depend on the delays.

The average (weighted) regret of OGD can be shown to be,

$$O\left(\frac{\text{Reg}_T(\mathbf{w}^*)}{\alpha_{1:T}}\right) = O(GD/\sqrt{T}) \ .$$

Combining this with Theorem 3.1 implies that delay adaptive SGD (Eq. (3)) obtains an overall rate of $O(GD/\sqrt{T} + LD^2\mu_\tau/T)$. For completeness, we include in Section C a proof for the OGD average regret bound. The full proof of Theorem 3.1 can be seen in the supplementary material. We present here a rough description of it.

Proof Sketch. First, using the gradient inequality together with $\alpha_t(\mathbf{x}_t - \mathbf{w}_t) = \alpha_{1:t-1}(\mathbf{x}_{t-1} - \mathbf{x}_t)$, which follows from the definition of $\mathbf{x}_t$, we can decompose as follows,

$$\begin{aligned} &\mathbb{E}\left[\sum_{t=1}^T \alpha_t(f(\mathbf{x}_t) - f(\mathbf{w}^*))\right] \\ &\leq \mathbb{E}\left[\sum_{t=1}^T \alpha_{1:t-1}\nabla f(\mathbf{x}_t)^\top(\mathbf{x}_{t-1} - \mathbf{x}_t) + \text{Reg}_T(\mathbf{w}^*)\right] \\ &+ \mathbb{E}\left[\sum_{t=1}^T \alpha_t\|\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-\tau_t})\|\|\mathbf{w}_t - \mathbf{w}^*\|\right] \ , \end{aligned} \tag{4}$$

where $\text{Reg}_T(\mathbf{w}^*)$ is the regret of $\mathcal{A}$ with respect to the sequence $\{f_t(\mathbf{x}) := \alpha_t \mathbf{g}_{t-\tau_t}^\top \mathbf{x}\}_{t\in[T]}$. Due to the smoothness of the objective, bounding the last term is equivalent to bounding $\|\mathbf{x}_t - \mathbf{x}_{t-\tau_t}\|$. Since $\mathbf{x}_t$ is a weighted average of the $\mathbf{w}_t$'s, one can show that,

$$\|\mathbf{x}_t - \mathbf{x}_{t-\tau_t}\| = O\left(\tau_t D/t\right) \ . \tag{5}$$

Note that querying the oracle at the iterate averages rather than the iterates themselves was a key factor for establishing this bound. Using Eq. (5), the last term of Eq. (4) is bounded by $LD^2\mu_\tau T$. From here we follow similar steps as in (Cutkosky, 2019) and show that,

$$\alpha_{1:T}\mathbb{E}\left[(f(\mathbf{x}_T) - f(\mathbf{w}^*))\right] \leq \text{Reg}_T(\mathbf{w}^*) + O\left(\sum_{t=1}^T \alpha_t \frac{\tau_t D^2}{t}\right) \ .$$

Plugging $\alpha_t = t$, and using the bound on the average regret concludes the proof. □

3.2. Fully Adaptive Algorithm

In this subsection we propose an *optimal delay-adaptive, delay-robust algorithm for the delayed setting*, which implicitly adapts both the variance and smoothness. Although

Algorithm 2 Delay Optimistic Adaptive Anytime Online to Batch

```
Input: # of iterations $T$, $\mathbf{w}_1 \in \mathcal{K}$, weights $\{\alpha_t\}_{t\in[T]}$
set $\mathbf{g}_{0-\tau_0} = 0$
for $t = 1$ to $T$ do
    send $\alpha_t \mathbf{M}_{t-\tau_t} = \alpha_t \mathbf{g}_{t-1-\tau_{t-1}}$ to $\mathcal{A}$ as the $t$'th hint
    $\mathbf{x}_t \leftarrow \frac{\sum_{i=1}^{t} \alpha_i \mathbf{w}_i}{\alpha_{1:t}}$
    get $\mathbf{g}_{t-\tau_t}$ from worker
    define $f_t(\mathbf{x}) = \mathbf{g}_{t-\tau_t}^\top \mathbf{x}$
    send $\alpha_t f_t(\mathbf{x})$ to $\mathcal{A}$
    get $\mathbf{w}_{t+1}$ from $\mathcal{A}$
end for
return $\mathbf{x}_T$
```

our scheme in Alg. 1 enables to use a variety of OCO algorithms, it does not necessarily obtains the optimal rates for the delayed setting. Moreover, the OCO algorithm $\mathcal{A}$ that we employ may require the knowledge of the problem parameters (like noise variance and smoothness) in order to ensure such optimal performance.

Our algorithmic scheme is depicted in Alg. 2. It is similar to the one we present in Alg. 1, only now we consider specialized OCO algorithms $\mathcal{A}$. Concretely, we assume that $\mathcal{A}$ is an optimistic and adaptive OCO method. Here we limit ourselves to OCO algorithms that receive a sequence of linear losses $f_t(\mathbf{x}) := \mathbf{g}_t^\top \mathbf{x}$.

Optimism: an optimistic OCO algorithm, (Rakhlin & Sridharan, 2013), is an algorithm that receives a "hint" $\mathbf{M}_t$ prior to choosing $\mathbf{w}_t$. When the hints are good estimates of the loss gradient at round t, i.e., $\mathbf{M}_t \approx \mathbf{g}_t$, such algorithms are able to use these hints in order to obtain better regret guarantees.

Adaptive Optimistic Algorithm: An adaptive optimistic OCO method is a method that upon receiving a sequence of linear losses $\{f_t(x) := \mathbf{g}_t^\top \mathbf{x}\}_{t\in[T]}$, and a sequence of hints $\{\mathbf{M}_t\}_{t\in[T]}$, enables to ensure a (weighted) regret bound of the following form,

$$\text{Reg}_T(\mathbf{w}^*) := \sum_{t=1}^{T} \alpha_t \mathbf{g}_t^\top (\mathbf{w}_t - \mathbf{w}^*) \leq O\left(D\sqrt{\sum_{T=1}^{T} \alpha_t^2 \|\mathbf{M}_t - \mathbf{g}_t\|^2} \right), \quad (6)$$

where α_t's are predefined weight vectors. Note that the bound indeed improves when $\mathbf{M}_t \approx \mathbf{g}_t$. In the context of Alg. 2, the hint vector prior to round t is $\mathbf{M}_{t-\tau_t} := \mathbf{g}_{t-1-\tau_{t-1}}$, and $\mathcal{A}$ receives the following loss sequence $\{f_t(\mathbf{x}) := \mathbf{g}_{t-\tau_t}^\top \mathbf{x}\}$. Adaptive optimistic methods are developed e.g. in (Rakhlin & Sridharan, 2013), as well as in Mohri & Yang (2016).

Remark 3.2. *The most natural example of an adaptive optimistic OCO method is the following Optimistic OGD algorithm (Rakhlin & Sridharan, 2013), $\forall t \geq 2$,*

$$\mathbf{w}_t = \Pi_{\mathcal{K}}(\mathbf{y}_{t-1} - \eta_t \alpha_t \mathbf{M}_t) \ \& \ \mathbf{y}_t = \Pi_{\mathcal{K}}(\mathbf{y}_{t-1} - \eta_t \alpha_t \mathbf{g}_t)$$

where, $\eta_t := D/\sqrt{1 + \sum_{i=1}^{t-1} \alpha_i^2 \|\mathbf{g}_i - \mathbf{M}_i\|^2}$, and $\mathbf{y}_1 = \mathbf{w}_1$ is an initial arbitrary point in $\mathcal{K}$.
Observe that optimistic OGD utilizes an additional sequence $\{\mathbf{y}_t\}_{t\in[T]}$: whenever a hint $\mathbf{M}_t$ is received, we use it to take a step from $\mathbf{y}_{t-1}$ to compute the next decision point $\mathbf{w}_t$. Then, once we observe the true feedback $\mathbf{g}_t$, we use it to compute $\mathbf{y}_t$ by taking a gradient step from $\mathbf{y}_{t-1}$.

The scheme in Alg. 2 is a combination of optimistic adaptive OCO method together with the anytime online to batch conversion scheme, that we apply to the delayed stochastic setting. It was previously shown in (Kavis et al., 2019) and (Cutkosky, 2019) that employing optimistic and adaptive OCO methods in the context of stochastic optimization enables to adapt to problem parameters like noise variance and smoothness. Our next statement shows that combining them within the anytime scheme enables to obtain a fully adaptive algorithm for the delayed setting.

Theorem 3.3. *Consider the delayed stochastic setting, and assume that $f : \mathcal{K} \mapsto \mathbb{R}$ is convex and L-smooth. Then, using Alg. 2 with $\alpha_t = t$ and an optimistic adaptive OCO method $\mathcal{A}$ that satisfies Eq. (6), ensures an* optimal convergence rate of,

$$\mathbb{E}\left[f(\mathbf{x}_T) - f(\mathbf{w}^*)\right] \leq O\left(\frac{LD^2(1 + \sqrt{\sigma_\tau^2 + \mu_\tau^2})}{T^{3/2}} \right) + O\left(\frac{LD^2 \mu_\tau}{T} + \frac{\sigma D}{\sqrt{T}} \right). \quad (7)$$

Note that Alg. 2 requires neither knowledge of L, σ nor information regarding the delays.

The rate in Theorem 3.3 matches the optimal rate for this setting. We actually obtain an improvement over (Arjevani et al., 2020; Stich & Karimireddy, 2020) since our bound implies that we do not degrade compared to the non-delayed setting as long as $\mu_\tau \leq O(\sqrt{T})$, while their bound necessitates $\tau_{\max} \leq O(\sqrt{T})$.

Proof sketch of Theorem 3.3. Using the bound on $\|\mathbf{x}_t - \mathbf{x}_{t-\tau_t}\|$ that we have previously shown, together with the bounds on the variance and smoothness, enables us to prove the following,

$$\mathbb{E}[\|\mathbf{M}_{t-\tau_t} - \mathbf{g}_{t-\tau_t}\|^2] \leq O\left(\frac{LD^2 \tau_t^2}{t^2} + \sigma^2 \right).$$

Then, by using some known algebraic inequalities, we can bound $\mathbb{E}[R_T(\mathbf{w}^*)]$ with $O\left(LD^2\sqrt{T(\sigma_\tau^2+\mu_\tau^2)}+D\sigma T^{3/2}\right)$. By combining this with Theorem 3.1, we receive the desired bound. $\square$

4. Delay Adaptive Method for the Strongly Convex Objectives

In this section we present a delay-adaptive algorithm for the delayed setting, assuming that the objective is not only smooth, but also strongly-convex. Previous works on the delayed strongly-convex setting (Stich & Karimireddy, 2020; Arjevani et al., 2020) require the knowledge of all relevant problem parameters including smoothness, strong-convexity, noise variance and maximal delay. Conversely, we only require the knowledge of the strong-convexity parameter. Moreover, our algorithm applies to constraint problems, which resolves an open problem of (Stich & Karimireddy, 2020).

Prior to handling the delayed setting, we first develop an adaptive SGD variant (Alg. 3) that applies to the standard strongly-convex, smooth, and constrained setting (Section 4.1). Particularly, Alg. 3 obtains a rate of $O(1/T^2+\sigma^2/T)$ without any prior knowledge except for the strong-convexity parameter. This is the first adaptive algorithm for this setting which does not require knowledge of the objective smoothness and noise variance, which might be of independent interest.

Then in Section 4.2 we introduce Alg. 4, an adaptation of Alg. 3 for the delayed setting, and prove convergence rate of $O((\sigma_\tau^2+\mu_\tau^2)/T^2+\sigma^2/T)$. Concretely, Alg. 4 implicitly accommodates changes in the delays and does not assume prior knowledge of σ or L. Conversely to the general convex case, our method here is not based on an anytime online-to-batch conversion.

4.1. Adaptive Optimistic Algorithm without Delays

Here we discuss the standard strongly-convex and smooth constrained setting, and develop an adaptive algorithm for this case, which only requires the knowledge of the strong-convexity parameter. Note that this setting was already analyzed in (Joulani et al., 2020) and (Zhang & Zhou, 2019), nevertheless, their methods require the knowledge of L and σ [1], similarly to the methods developed in (Stich & Karimireddy, 2020). Thus, it seems like adapting these methods to the delayed setting necessitates to equip them with the knowledge of strong-convexity and maximal delay, H and $\tau_{\max}$, as is apparent from (Stich & Karimireddy, 2020).

[1] Actually (Zhang & Zhou, 2019) do not require σ explicitly, nevertheless, they do not achieve a bound that captures the refined dependence on σ. It seems that in order to capture this dependence they should encode σ inside their learning rate.

Consider the following update rule,

Algorithm 3 Optimistic Strongly-Convex OGD

```
Input: # of iterations $T$, $\mathbf{x}_0=\mathbf{y}_0\in\mathcal{K}$, weights $\{\alpha_t\}_{t\in[T]}$, Strong-convexity $H$
for $t=1$ to $T$ do
  Define $\eta_t=\frac{8}{H\alpha_{1:t}}$
    $\mathbf{x}_t=\arg\min_{\mathbf{x}\in\mathcal{K}}\alpha_t\mathbf{M}_t^\top\mathbf{x}+\frac{1}{2\eta_t}\|\mathbf{x}-\mathbf{y}_{t-1}\|^2$
    $\mathbf{y}_t=\arg\min_{\mathbf{y}\in\mathcal{K}}\alpha_t\mathbf{g}_t^\top\mathbf{y}+\frac{1}{2\eta_t}\|\mathbf{y}-\mathbf{y}_{t-1}\|^2$
end for
return $\bar{\mathbf{x}}_T\propto\sum_{t=1}^T\alpha_t\mathbf{x}_t$
```

where $\mathbf{M}_t=\nabla f(\mathbf{x}_{t-1})+\boldsymbol{\xi}_{t-1}$ and $\mathbf{g}_t=\nabla f(\mathbf{x}_t)+\boldsymbol{\xi}_t$.

Alg. 3 depicts a gradient weighting scheme of optimistic OGD algorithm (Rakhlin & Sridharan, 2013), that we adapt to the strongly convex case. Indeed, as proven in Appendix E.2, Alg. 3 is equivalent to the optimistic OGD scheme that we described in Remark 3.2, yet with a learning rate which is adapted to the strongly convex case,

Remark 4.1. *The update rule in Alg. 3 is equivalent to,*

$$\mathbf{x}_t=\Pi_{\mathcal{K}}(\mathbf{y}_{t-1}-\tilde{\eta}_t\mathbf{M}_t)\quad\&\quad\mathbf{y}_t=\Pi_{\mathcal{K}}(\mathbf{y}_{t-1}-\tilde{\eta}_t\mathbf{g}_t)$$

where $\tilde{\eta}_t:=\frac{8\alpha_t}{H\alpha_{1:t}}$.

As we mention in Section 3.2, optimistic methods can implicitly utilize smoothness to ensure better performance. When the objective is also strongly-convex, we can further improve our convergence guarantees as stated in Theorem 4.2 below.

Theorem 4.2. *Assume that $f:\mathcal{K}\mapsto\mathbb{R}$ is H-strongly convex and L-smooth. Then, using Alg. 3 with $\alpha_t=t^2$ ensures,*

$$\mathbb{E}[f(\bar{\mathbf{x}}_T)-f(\mathbf{w}^*)]\le O\left(\frac{(G^2+\sigma^2)/H}{T^2}+\frac{\sigma^2}{TH}\right).$$

The proof of Theorem 4.2 is described fully in Appendix E.

4.2. Delayed Constrained Setting

Consider the following update rule when delays are present,

Algorithm 4 Optimistic Strongly-Convex OGD with Delays

```
Input: # of iterations $T$, $\mathbf{x}_0 = \mathbf{y}_0 \in \mathcal{K}$, weights $\{\alpha_t\}_{t\in[T]}$, Strong-convexity $H$
for $t = 1$ to $T$ do
    Define $\eta_t = \frac{8}{H\alpha_{1:t}}$
        $\mathbf{x}_t = \arg\min_{\mathbf{x}\in\mathcal{K}} \alpha_t \mathbf{x}^\top \mathbf{M}_{t-\tau_t} + \frac{1}{2\eta_t}\|\mathbf{x} - \mathbf{y}_{t-1}\|^2$
        $\mathbf{y}_t = \arg\min_{\mathbf{y}\in\mathcal{K}} \alpha_t \mathbf{y}^\top \mathbf{g}_{t-\tau_t} + \frac{1}{2\eta_t}\|\mathbf{y} - \mathbf{y}_{t-1}\|^2$
end for
return $\bar{\mathbf{x}}_T \propto \sum_{t=1}^T \alpha_t \mathbf{x}_t$
```

where $\mathbf{M}_{t-\tau_t} = \nabla f(\mathbf{x}_{t-1-\tau_{t-1}}) + \boldsymbol{\xi}_{t-1-\tau_{t-1}}$ and $\mathbf{g}_{t-\tau_t} = \nabla f(\mathbf{x}_{t-\tau_t}) + \boldsymbol{\xi}_{t-\tau_t}$.

The main statement for the strongly convex case under delayed feedback is stated below.

Theorem 4.3. *Assume that $f : \mathcal{K} \mapsto \mathbb{R}$ is H-strongly convex and L-smooth. Then using Alg. 4 with $\alpha_t = t^2$ ensures,*

$$\mathbb{E}[f(\bar{\mathbf{x}}_T) - f(\mathbf{w}^*)] \leq O\left(\frac{(G^2+\sigma^2)}{HT^2} + \frac{\sigma^2}{HT}\right) + O\left(\frac{L^2(G^2+\sigma^2)(\sigma_\tau^2+\mu_\tau^2)}{H^3T^2}\right) .$$

Theorem 4.3 offers a fully adaptive algorithm with convergence guarantees. Concretely, the slower converging term with variance dependency is similar to the one in (Stich & Karimireddy, 2020), and does not depend upon the delays. The above bound implies that we do not degrade compared to non-delayed SGD as long as $\mu_\tau \leq O(\sqrt{T})$.

Proof Sketch (Theorem 4.3). Here we highlight the part in the proof that incorporates the delays, and demonstrate how we handle it. The additional term in the analysis that arises due to delays is the following,

$$(\mathrm{D}) := \sum_{t=1}^T \alpha_t(\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-\tau_t}))^\top(\mathbf{x}_t - \mathbf{w}^*) .$$

Using $ab \leq \inf_{\rho>0}\left(\rho a^2/2 + b^2/(2\rho)\right)$, together with the strong-convexity of $f(\cdot)$ which implies $H\|\mathbf{x}_t - \mathbf{w}^*\|^2 \leq 2(f(\mathbf{x}_t) - f(\mathbf{w}^*))$ enables to bound (D) as follows,

$$\leq \sum_{t=1}^T \frac{\alpha_t H}{8}\|\mathbf{x}_t - \mathbf{w}^*\|^2 + \frac{2\alpha_t}{H}\left\|\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-\tau_t})\right\|^2$$
$$\leq \sum_{t=1}^T \frac{\alpha_t}{4}(f(\mathbf{x}_t) - f(\mathbf{w}^*)) + \frac{2\alpha_t}{H}\left\|\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-\tau_t})\right\|^2$$

The left term is directly related to the error, and the right term is then bounded using $\|\mathbf{x}_t - \mathbf{x}_{t-\tau_t}\| \leq O(\tau_t/t)$ combined with the smoothness of the objective. □

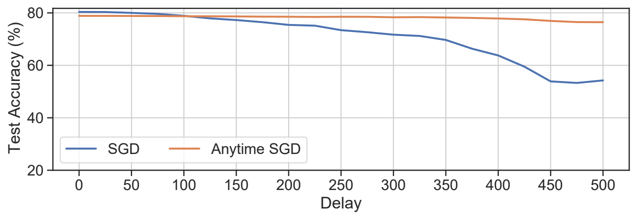

Figure 1. Accuracy as a function of update delays, with learning rate optimized for each of the algorithms for zero delay case.

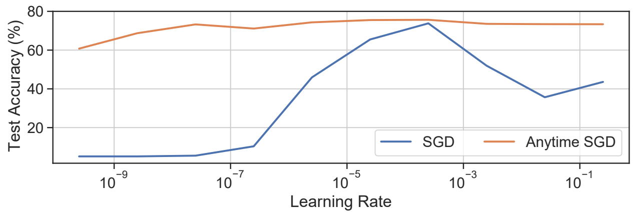

Figure 2. Accuracy as a function of learning rate when $\tau_t = 500$.

5. Experiments

In this section, we study the performance of anytime SGD: Algorithm 1 using SGD as the OCO algorithm, as described in Eq. (3). We trained anytime SGD on Fashion-MNIST (Xiao et al., 2017) dataset with logistic regression model and evaluated it using multi-class log loss. We compare our algorithm to SGD with validated constant step size, as was suggested in (Stich & Karimireddy, 2020). Fashion-MNIST consists of $6 \cdot 10^4$ training examples and $1 \cdot 10^4$ test examples, when each example is a 28x28 grayscale clothing image, associated with a label from 10 categories. The distributed system is simulated via an update queue, similarly to what was done in (McMahan & Streeter, 2014), which allows us to simulate delays of our choice.

We show both scalability and robustness advantages of anytime SGD. As was mentioned before, on scalable systems the delay regime may change during the operation, making both these qualities essential for adequate performance. In addition, hyper-parameter tuning is expensive (Akiba et al., 2019; Bergstra et al., 2011; Falkner et al., 2018), especially when required to tune with respect to different delay regimes.

Our scalability and delay adaptivity experiment is demonstrated in Figure 1, where anytime SGD maintains high final accuracy even with significant amount of delay in the training. In this experiment we tuned the learning rate for each algorithm separately only in the zero delay setting, and then evaluate them on different delay regimes. Figure 1 shows that as the delay increases, the gap between the final accuracy of anytime SGD and SGD grows. Thus, while our algorithm maintains its high performance for different delay regimes with no further tuning, SGD greatly deteriorates. Note that similar results were received when tuning the algorithms with larger delay, and then changing the delay regime.

The robustness of anytime SGD is presented in Figure 2, where anytime SGD achieves high accuracy on a wide range of learning rates, whereas SGD achieves its top accuracy on a very narrow range. In this experiment we evaluated the algorithms with different learning rates, when $\tau_t = 500$. This shows that anytime SGD is much less sensitive to learning changes than SGD.

For additional experimental results, please refer to Section G in the supplement.

6. Discussion

We leverage anytime online to batch scheme to prove convergence guarantees for delayed feedback setting, when updates are delayed to an unknown extent. We propose a delay adaptive training methods for constrained setting which do not depend on prior knowledge of problem parameters. We demonstrate experimentally (see Figures 1,2) that our algorithm outperforms alternative schemes for delayed feedback setting, exhibiting accuracy and robustness even when substantial delays are present.

One interesting direction for future research is to extend our approach to be used with accelerated first-order methods (Cutkosky, 2019; Kavis et al., 2019). Moreover, our results suggest that incorporating anytime online to batch scheme has great potential for other challenging settings as well.

Acknowledgements

The work on this paper was supported in part by the Israeli Ministry of Science, Technology, and Space and by The Hasso Plattner Institute at the Technion. K.Y. Levy acknowledges support from the Israel Science Foundation (grant No. 447/20).

References

Agarwal, A. and Duchi, J. C. Distributed delayed stochastic optimization. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 5451–5452. IEEE, 2012.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Arjevani, Y., Shamir, O., and Srebro, N. A tight convergence analysis for stochastic gradient descent with delayed updates. In *Algorithmic Learning Theory*, pp. 111–132. PMLR, 2020.

Ben-Nun, T. and Hoefler, T. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, volume 24. Neural Information Processing Systems Foundation, 2011.

Bertsekas, D. P. and Tsitsiklis, J. N. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Cotter, A., Shamir, O., Srebro, N., and Sridharan, K. Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems*, pp. 1647–1655, 2011.

Cutkosky, A. Anytime online-to-batch, optimism and acceleration. In *International Conference on Machine Learning*, pp. 1446–1454, 2019.

Dekel, O., Gilad-Bachrach, R., Shamir, O., and Xiao, L. Optimal distributed online prediction using minibatches. *Journal of Machine Learning Research*, 13 (Jan):165–202, 2012.

Dutta, S., Joshi, G., Ghosh, S., Dube, P., and Nagpurkar, P. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd. In *International Conference on Artificial Intelligence and Statistics*, pp. 803–812. PMLR, 2018.

Falkner, S., Klein, A., and Hutter, F. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1437–1446. PMLR, 2018.

Feyzmahdavian, H. R., Aytekin, A., and Johansson, M. An asynchronous mini-batch algorithm for regularized stochastic optimization. *IEEE Transactions on Automatic Control*, 61(12):3740–3754, 2016.

Jain, P., Kakade, S. M., Kidambi, R., Netrapalli, P., and Sidford, A. Parallelizing stochastic approximation through mini-batching and tail-averaging. *arXiv preprint arXiv:1610.03774*, 2016.

Joulani, P., Gyorgy, A., and Szepesvári, C. Online learning under delayed feedback. In *International Conference on Machine Learning*, pp. 1453–1461. PMLR, 2013.

Joulani, P., Raj, A., Gyorgy, A., and Szepesvari, C. A simpler approach to accelerated optimization: iterative averaging meets optimism. In *International Conference on Machine Learning*, pp. 4984–4993. PMLR, 2020.

Kavis, A., Levy, K. Y., Bach, F., and Cevher, V. Unixgrad: A universal, adaptive algorithm with optimal guarantees for constrained optimization. In *Advances in Neural Information Processing Systems*, pp. 6260–6269, 2019.

Leblond, R., Pedregosa, F., and Lacoste-Julien, S. Improved asynchronous parallel optimization analysis for stochastic incremental methods. *arXiv preprint arXiv:1801.03749*, 2018.

Li, M., Zhang, T., Chen, Y., and Smola, A. J. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670. ACM, 2014.

Lian, X., Huang, Y., Li, Y., and Liu, J. Asynchronous parallel stochastic gradient for nonconvex optimization. *arXiv preprint arXiv:1506.08272*, 2015.

McMahan, H. B. and Streeter, M. Delay-tolerant algorithms for asynchronous distributed online learning. 2014.

Mohri, M. and Yang, S. Accelerating online convex optimization via adaptive prediction. In *Artificial Intelligence and Statistics*, pp. 848–856, 2016.

Nedić, A., Bertsekas, D. P., and Borkar, V. S. Distributed asynchronous incremental subgradient methods. *Studies in Computational Mathematics*, 8(C):381–407, 2001.

Needell, D., Srebro, N., and Ward, R. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. *arXiv preprint arXiv:1310.5715*, 2013.

Rakhlin, A. and Sridharan, K. Optimization, learning, and games with predictable sequences. *arXiv preprint arXiv:1311.1869*, 2013.

Recht, B., Re, C., Wright, S., and Niu, F. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24:693–701, 2011.

Ren, Z., Zhou, Z., Qiu, L., Deshpande, A., and Kalagnanam, J. Delay-adaptive distributed stochastic optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5503–5510, 2020.

Shalev-Shwartz, S. and Zhang, T. Accelerated mini-batch stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pp. 378–385, 2013.

Shalev-Shwartz, S. et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

Shamir, O. and Srebro, N. Distributed stochastic optimization and learning. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 850–857. IEEE, 2014.

Sra, S., Yu, A. W., Li, M., and Smola, A. J. Adadelay: Delay adaptive distributed stochastic convex optimization. *arXiv preprint arXiv:1508.05003*, 2015.

Stich, S. U. and Karimireddy, S. P. The error-feedback framework: Better rates for sgd with delayed gradients and compressed updates. *Journal of Machine Learning Research*, 21:1–36, 2020.

Takáč, M., Richtárik, P., and Srebro, N. Distributed mini-batch sdca. *arXiv preprint arXiv:1507.08322*, 2015.

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., and Rellermeyer, J. S. A survey on distributed machine learning. *ACM Computing Surveys (CSUR)*, 53(2):1–33, 2020.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhang, L. and Zhou, Z.-H. Stochastic approximation of smooth and strongly convex functions: Beyond the $o(1/t)$ convergence rate. In *Conference on Learning Theory*, pp. 3160–3179. PMLR, 2019.

Zheng, S., Meng, Q., Wang, T., Chen, W., Yu, N., Ma, Z.-M., and Liu, T.-Y. Asynchronous stochastic gradient descent with delay compensation. In *International Conference on Machine Learning*, pp. 4120–4129. PMLR, 2017.

Zhou, Z., Mertikopoulos, P., Bambos, N., Glynn, P., Ye, Y., Li, L.-J., and Fei-Fei, L. Distributed asynchronous optimization with unbounded delays: How slow can you go? In *International Conference on Machine Learning*, pp. 5970–5979. PMLR, 2018.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.