Online optimization for machine learning: parallelism, adaptivity, and model selection

by

Pooria Joulani

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Statistical Machine Learning

Department of Computing Science

University of Alberta

Abstract

We study three problems in the application, design, and analysis of online optimization algorithms for machine learning. First, we consider speeding-up the common task of k-fold cross-validation of online algorithms, and provide TreeCV, an algorithm that reduces the time penalty of k-fold cross-validation from k to $\log(k)$, and is easily adoptable to parallel and distributed computing environments. Second, we consider algorithms for online delayed-feedback distributed optimization, and provide Solid, a meta-algorithm for deriving delay-tolerant versions of standard online optimization algorithms and their analysis. We further apply Solid to obtain algorithms that adapt to the delay pattern observed during optimization, solving an open problem from the literature. Third, we study asynchronous online and stochastic optimization. We start by providing a unifying analysis of standard serial online optimization algorithms. Then, we build on this analysis to design and analyze two new asynchronous online optimization algorithms. The first algorithm, AsynCADA, features the ability to handle generic convex constraints, proximal updates and adaptive step-sizes. The second algorithm, HedgeHog, is the first asynchronous variant of the Hedge algorithm. Both algorithms enjoy linear speed-up if the data is sufficiently sparse, extending the scope of problem settings and algorithmic techniques adoptable to asynchronous optimization. Underlying the analysis of AsynCADA and HedgeHog is a generic framework for studying online optimization algorithms with perturbed state, which is of independent interest.

Preface

The research included in this thesis is my original work. Chapter 2 has been published as the following conference paper:

- P. Joulani *et al.*, "Fast cross-validation for incremental learning," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*, 2015.

Chapter 3 has been published as the following conference paper:

- P. Joulani *et al.*, "Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms," in *Proceedings of the 30th Conference on Artificial Intelligence (AAAI-16)*, 2016.

Chapter 4 has been accepted for publication in the Theoretical Computer Science (TCS) journal. An earlier version has been published as the following conference paper:

- P. Joulani *et al.*, "A modular analysis of adaptive (non-) convex optimization: Optimism, composite objectives, and variational bounds," in *Proceedings of Machine Learning Research (Algorithmic Learning Theory 2017)*, 2017, pp. 681–720.

Chapter 5 is the basis of a conference submission currently under review. Parts of Chapter 5 have been completed while I was at DeepMind®, U.K.

Acknowledgements

I would like to express my gratitude

- To Csaba and Andras: Thank you for being awesome, dependable supervisors as well as friends. Your patience and support provided an environment in which I could freely learn and explore. I will always remember the many hours I spent with Andras to learn math, writing, and research, and the day I entered Csaba's office at 4pm and came out at 7pm, when he helped me hone the details of a probabilistic argument. I learned so much from you, and can't thank you enough.

- To Dale Schuurmans: Thank you for teaching me how to see the forest of ideas from within the trees of publications. What I learned from you has been essential in shaping my career. Thanks also for your questions, ideas, and essential support as a member of my supervisory committee.

- To Martin Müller: Thank you for being central to several of my oral examinations in the past years, for all your comments and questions, and for the detailed reading of all my manuscripts.

- To Michael Bowling: Thank you for the career advice you gave me when I needed it most, and for improving my work with your comments and questions throughout my candidacy and defense.

- To my students: Thank you for sharing with me the joy of learning. Seeing the shine of understanding in your eyes has been such a fulfilling experience.

- To my friends: Thank you for being there when I needed you, even before I asked. Special thanks to Hamidreza Anvari, Milad Yavari, and Farhad Azam for being brothers rather than friends, to MohammadHossein Bitarafan, Saeed Toghi and the

Izadi brothers for making Edmonton winters warm and lively, to Mohammad Mahdi Ajalloeian for all the great discussions we had, and to all the friends in the RLAI Lab.

- To the administrative staff at the Department of Computing Science: Thank you for making it a great experience to live and learn at this department. Special thanks to Sharon Bell for being always helpful, and to Kerstyn Sorenson for making my defense possible with her excellent administration.
- To my parents and grandparents: Thank you for making it impossible to express my gratitude. Thanks for your care, hard work, and sacrifice, for your unconditional love, and for your patience until I could finally understand it all.
- To my brother Parham: Thank you for being an inseparable part of my life, and a great uncle to Zahra and Alireza.
- To Nooshin: Thank you for being my love, support and inspiration all along the way. This thesis in general, and TreeCV in particular, would not exist without you, your brilliant mind, and your patience. It's been wonderful to share my past eight years with you, Zahra, and Alireza, and it is wonderful to look forward to every other moment to share.

This thesis was supported by AICML (now Amii), NSERC, and a QEII Doctoral Scholarship. The last chapter was in part supported by DeepMind®.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Online learning and online optimization algorithms play a central role in modern machine learning. The goal of this thesis is to provide new theoretical and algorithmic tools for creating, utilizing and analyzing online optimization algorithms. We consider three different problems in the application, analysis and design of online algorithms.

1.1 Estimating performance by cross-validation

First, in Chapter 2, we look at a central, but computationally expensive, task in machine learning: k-fold cross-validation (k-CV) for estimating the generalization performance and parameter-tuning of algorithms.

We show that by exploiting the incremental nature of online algorithms, it is possible to avoid the $(k-1)$-times re-training over the data that the standard k-CV estimation method requires. In particular, we propose TreeCV, an algorithm that speeds up, exponentially in k, the computation of the k-CV score for incremental algorithms. The core idea is arranging the training and evaluation of the online algorithm in a tree structure, so that the different k-CV folds can share what is learned from the data common to the folds.

We further show that TreeCV has another desirable property: it immediately extends to parallel multi-threaded computing environments, as well as distributed computing environments in which the data is spread over a network. In this way, TreeCV further improves over the already parallelizable but rather wasteful standard k-CV algorithm.

We also study the theoretical properties of the cross-validation estimate computed by TreeCV. We show that the TreeCV estimate is accurate, in the sense that it is close to the standard k-CV estimate, as long as the underlying incremental algorithms are stable, in the sense of learning a model with low generalization error irrespectively of the re-ordering of the data points they receive.

Finally, we evaluate TreeCV empirically on supervised learning tasks using two standard classification and regression algorithms, validating our theoretical results: TreeCV considerably speeds up cross-validation, even for small values of k such as $k = 5$ or 10, and even makes the calculation of leave-one-out estimates practical on the data-sets we consider.

1.2 Adapting to delays in distributed online optimization

Next, in Chapter 3, we focus on enabling sequential online learning algorithms to take better advantage of the modern parallel and distributed computing environments. Specifically, running online optimization algorithms in distributed environments introduces delay in observing their feedback, breaking the sequential nature of these algorithms. We study the problem of controlling the effect of such delays, and adapting to them, by properly tuning the existing adaptive online optimization algorithms. This problem is especially important in heterogeneous distributed environments (such as cloud-based systems) where the delays can vary greatly over time, and adapting to delays can result in considerable performance gains [66], [101]. Previously, McMahan and Streeter [66] had proposed an algorithm called `AdaptiveRevision`, which used the special structure of an unconstrained online update to adapt to the delays, and left open the problem of analyzing a wider range of algorithms without relying on their rather restrictive assumptions.

To study this effect, we first take a step back, and analyze the effect of delays independently of the specific mechanics of the algorithm. To that end, we present a meta-algorithm, called Solid, that in a black-box manner enables sequential online optimization algorithms to tolerate delay in their feedback. We show that this reduction preserves the sequential performance of the online optimization algorithm, up to a penalty that depends on the magnitude of delays and the algorithm's stability (i.e., how fast the algorithm changes its predictions).

Finally, we apply this meta-algorithm to two generic classes of adaptive online optimization algorithms: Adaptive Mirror Descent (MD) and Adaptive Proximal Follow-the-

Regularized-Leader (FTRL-Prox). We show that these two algorithm classes are stable, and hence Solid results in concrete bounds on their performance. Using these bounds, we can then tune the algorithms' parameters to adapt to the observed delay pattern, solving the aforementioned open problem.

1.3 Improving asynchronous online and stochastic optimization

The final question we study in this thesis is extending online optimization algorithms and their convergence guarantees to asynchronous parallel computing. Given that much of the existing computing power comes in the form of multi-core and multi-processor systems, this question has received considerable attention in the past decade. Especially, there has been much progress in designing and analyzing asynchronous algorithms for stochastic optimization (a subset of the online optimization setting we consider).

Nevertheless, several problems in the design and analysis of asynchronous parallel online and stochastic optimization remain open. In particular, there has been little progress in a) asynchronous algorithms for generically-constrained composite online optimization, especially when involving more complex constraint sets; b) analysis of asynchronous optimization algorithms which don't have a simple gradient descent (GD) type of update; and c) asynchronous online optimization under more relaxed assumptions on the objective, beyond the usual strongly-convex assumption or even beyond the stochastic optimization settings. The first question is important since several successful machine learning methods for sparse learning and classification are formulated as non-smooth composite optimization problems, and scaling them as the data sets get larger is important. The second problem is interesting since optimization algorithms such as the Exponentiated Gradient (EG) algorithm are not instances of GD, yet they are the basis of widely applied algorithms in online learning. In addition, more complex GD-style algorithms such as proximal-update algorithms are practically important in serial optimization. The third problem is interesting since the stochastic optimization problem is mainly concerned with the optimization of a fixed function, whereas machine learning applications such as online advertising typically fall into the more general framework of online learning.

To move toward a solution to the problems above, we provide the basis of a framework for analyzing asynchronous online optimization algorithms, that is flexible enough to capture a wide range of algorithms and problem settings, as well as common asynchrony patterns of interest, such as those arising from parallel shared-memory optimization. We develop this framework in two steps.

1.3.1 Step 1: Improving the analysis of serial online optimization algorithms

To enable a simplified analysis of a larger class of asynchronous online optimization algorithms, we first provide, in Chapter 4, a unifying analysis of the performance of a large class of *serial* algorithms for composite online optimization, in particular, of generalized adaptive MD and adaptive FTRL. Our goal is to provide a *modular* analysis which decomposes the effects of different assumptions and algorithmic techniques, so that this decomposition paves the way for studying the effect of asynchrony under each of these assumptions.

To that end, we build on the existing analysis ideas in serial online learning, but emphasize unifying the algorithms, analyzing the resulting generic algorithms under minimal assumptions, and isolating the effect of each assumption or algorithmic technique. This emphasis allows us to capture, and easily combine, different algorithmic ideas and problem settings.

As a side-product of this refined analysis, we also develop new algorithms and convergence results. In particular, we develop a new, more efficient optimistic MD update, which unlike previous work, requires only one projection per step. Furthermore, we combine the best features of several previous works to analyze a new scale-free algorithm for adaptive optimistic composite-objective online optimization that learns faster in the face of a slowly-changing environment. Finally, we show that these bounds can be effortlessly extended to a family of practically-relevant non-convex optimization problems.

1.3.2 Step 2: Analyzing asynchronous online optimization

In Chapter 5, we build on the framework of Step 1 to analyze asynchronous online optimization algorithms. Extending ideas from the asynchronous stochastic optimization literature, we cast the effect of asynchrony as perturbation to the "state" (i.e., accumulated information) of the online algorithm. Then, we extend the analysis of Step 1, and provide a flexible data-dependent convergence guarantee for generic adaptive FTRL algorithms with perturbed state. This data-dependent guarantee enables us to design and analyze two specific, new algorithm: AsynCADA with adaptive proximal updates for asynchronous composite-objective optimization over generic convex constraint sets, and HedgeHog, an asynchronous variant of the EG algorithm. The two algorithms enjoy linear speed-ups when the data is sparse, and apply not only to stochastic optimization, but to the more general setting of noisy online optimization, addressing the questions mentioned above.

We conclude by discussing future applications of the aforementioned results.

Chapter 2

TreeCV: Fast Cross-Validation for Incremental Algorithms

Abstract

Cross-validation (CV) is one of the main tools for performance estimation and parameter tuning in machine learning. The general recipe for computing a CV estimate is to run a learning algorithm separately for each CV fold, a computationally expensive process. In this paper, we propose a new approach to reduce the computational burden of CV-based performance estimation. As opposed to all previous attempts, which are specific to a particular learning model or problem domain, we propose a general method applicable to a large class of incremental learning algorithms, which are uniquely fitted to big data problems. In particular, our method applies to a wide range of supervised and unsupervised learning tasks with different performance criteria, as long as the base learning algorithm is incremental. We show that the running time of the algorithm scales logarithmically, rather than linearly, in the number of CV folds. Furthermore, the algorithm has favorable properties for parallel and distributed implementation. Experiments with state-of-the-art incremental learning algorithms confirm the practicality of the proposed method.[1]

[1] This chapter has been published as:

- P. Joulani *et al.*, "Fast cross-validation for incremental learning," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*, 2015.

2.1 Introduction

Estimating generalization performance is a core task in machine learning. Often, such an estimate is computed using k-fold cross-validation (k-CV): the dataset is partitioned into k subsets of approximately equal size, and each subset is used to evaluate a model trained on the $k-1$ other subsets to produce a numerical score; the k-CV performance estimate is then obtained as the average of the obtained scores.

A significant drawback of k-CV is its heavy computational cost. The standard method for computing a k-CV estimate is to train k separate models independently, one for each fold, requiring (roughly) k-times the work of training a single model. The extra computational cost imposed by k-CV is especially high for leave-one-out CV (LOOCV), a popular variant, where the number of folds equals the number of samples in the dataset. The increased computational requirements may become a major problem, especially when CV is used for tuning hyper-parameters of learning algorithms in a grid search, in which case one k-CV session needs to be run for every combination of hyper-parameters, dramatically increasing the computational cost even when the number of hyper parameters is small.[2]

To avoid the added cost, much previous research went into studying the efficient calculation of the CV estimate (exact or approximate). However, previous work has been concerned with special models and problems: With the exception of Izbicki [42], these methods are typically limited to linear prediction with the squared loss and to kernel methods with various loss functions, including twice-differentiable losses and the hinge loss (see Section 2.1.1 for details). In these works, the training time of the underlying learning algorithm is $\Theta(n^3)$, where n is the size of the dataset, and the main result states that the CV-estimate (including LOOCV estimates) is yet computable in $O(n^3)$ time. Finally, Izbicki [42] gives a very efficient solution (with $O(n+k)$ computational complexity) for the restrictive case when two models trained on any two datasets can be combined, in constant time, into a single model that is trained on the union of the datasets.

Although these results are appealing, they are limited to methods and problems with specific features. In particular, they are unsuitable for big data problems where the only practical methods are incremental and run in linear, or even sub-linear time [20], [99]. In

[2] For example, the semi-supervised anomaly detection method of Görnitz *et al.* [36] has four hyper-parameters to tune. Thus, testing all possible combinations for, e.g., 10 possible values of each hyper-parameter requires running CV 10000 times.

this paper, we show that CV calculation can be done efficiently for incremental learning algorithms. In Section 2.3, we present a method that, under mild, natural conditions, speeds up the calculation of the k-CV estimate for incremental learning algorithms, in the general learning setting explained in Section 2.2 (covering a wide range of supervised and unsupervised learning problems), and for arbitrary performance measures. The proposed method, TreeCV, exploits the fact that incremental learning algorithms do not need to be fed the whole dataset at once, but instead learn from whatever data they are provided with and later update their models when more data arrives, without the need to be trained on the whole dataset from scratch. As we will show in Section 2.3.1, TreeCV computes a guaranteed-precision approximation of the CV estimate when the algorithms produce *stable* models. We present several implementation details and analyze the time and space complexity of TreeCV in Section 2.4. In particular, we show that its computation time is only $O(\log k)$-times bigger than the time required to train a single model, which is a major improvement compared to the k-times increase required for a naive computation of the CV estimate. Finally, Section 2.5 presents experimental results, which confirm the efficiency of the proposed algorithm.

2.1.1 Related Work

Various methods, often specialized to specific learning settings, have been proposed to speed up the computation of the k-CV estimate. Most frequently, efficient k-CV computation methods are specialized to the regularized least-squares (RLS) learning settings (with squared-RKHS-norm regularization). In particular, the generalized cross-validation method [34], [104] computes the LOOCV estimate in $O(n^2)$ time for a dataset of size n from the solution of the RLS problem over the whole dataset; this is generalized to k-CV calculation in $O(n^3/k)$ time by Pahikkala *et al.* [83]. In the special case of least-squares support vector machines (LSSVMs), Cawley [14] shows that LOOCV can be computed in $O(n)$ time using a Cholesky factorization (again, after obtaining the solution of the RLS problem). It should be noted that all of the aforementioned methods use the inverse (or some factorization) of a special matrix (called the *influence matrix*) in their calculation; the aforementioned running times are therefore based on the assumption that this inverse is available (usually as a by-product of solving the RLS problem, computed in $\Omega(n^3)$ time).[3]

A related idea for approximating the LOOCV estimate is using the notion of *influence functions*, which measure the effect of adding an infinitesimal single point of probability mass to a distribution. Using this notion, Debruyne *et al.* [22] propose to approximate the LOOCV estimate for kernel-based regression algorithms that use any twice-differentiable loss function. Liu *et al.* [63] use *Bouligand influence functions* [19], a generalized notion of influence functions for arbitrary distributions, in order to calculate the k-CV estimate for kernel methods and twice-differentiable loss functions. Again, these methods need an existing model trained on the whole dataset, and require $\Omega(n^3)$ running time.

A notable exception to the square-loss/differentiable loss requirement is the work of Cauwenberghs and Poggio [13]. They propose an incremental training method for support-vector classification (with the hinge loss), and show how to revert the incremental algorithm to “unlearn” data points and obtain the LOOCV estimate. The LOOCV estimate is obtained in time similar to that of a single training by the same incremental algorithm, which is $\Omega(n^3)$ in the worst case.

Closest to our approach is the recent work of Izbicki [42]: assuming that two models trained on any two separate datasets can be combined, in constant time, to a single model that is exactly the same as if the model was trained on the union of the datasets, Izbicki [42] can compute the k-CV estimate in $O(n + k)$ time. However his assumption is very restrictive and applies only to simple methods, such as Bayesian classification.[4] In contrast, roughly, we only assume that a model can be updated efficiently with new data (as opposed to combining the existing model and a model trained on the new data in constant time), and we only require that models trained with permutations of the data be sufficiently similar, not exactly the same.

Note that the CV estimate depends on the specific partitioning of the data on which it is calculated. To reduce the variance due to different partitionings, the k-CV score can be averaged over multiple random partitionings. For LSSVMs, An *et al.* [5] propose a method to efficiently compute the CV score for multiple partitionings, resulting in a total running time of $O(L(n - b)^3)$, where L is the number of different partitionings and b is the number

[3]In the absence of this assumption, stochastic trace estimators [32] or numerical approximation techniques [33], [81] are used to avoid the costly inversion of the matrix.

[4] The other methods considered by Izbicki [42] do not satisfy the theoretical assumptions of that paper.

Setting	$\mathcal{X}$	$\mathcal{Y}$	$\mathcal{P}$	$\ell(f(x), x, y)$
Classification	$\mathbb{R}^d$	$\{+1, -1\}$	$\{+1, -1\}$	$\mathbb{I}\{f(x) \neq y\}$
Regression	$\mathbb{R}^d$	$\mathbb{R}$	$\mathbb{R}$	$(f(x) - y)^2$
K-means clustering	$\mathbb{R}^d$	$\{\text{NoLabel}\}$	$\{c_1, c_2, \ldots, c_K\} \subset \mathbb{R}^d$	$\|x - f(x)\|^2$
Density estimation	$\mathbb{R}^d$	$\{\text{NoLabel}\}$	$\{f : f \text{ is a density}\}$	$-\log(f(x))$

Table 2.1: Instances of the general learning problem considered in the paper. In K-means clustering, c_j denotes the center of the jth cluster.

of data points in each test set. In the case when all possible partitionings of the dataset are used, the complete CV (CCV) score is obtained. Mullin and Sukthankar [73] study efficient computation of CCV for nearest-neighbor-based methods; their method runs in time $O(n^2k + n^2 \log(n))$.

2.2 Problem Definition

We consider a general setting that encompasses a wide range of supervised and unsupervised learning scenarios (see Table 2.1 for a few examples). In this setting, we are given a dataset $\{z_1, z_2, \ldots, z_n\}$,[5] where each *data point* $z_i = (x_i, y_i)$ consists of an *input* $x_i \in \mathcal{X}$ and an *outcome* $y_i \in \mathcal{Y}$, for some given sets $\mathcal{X}$ and $\mathcal{Y}$. For example, we might have $\mathcal{X} \subset \mathbb{R}^d, d \geq 1$, with $\mathcal{Y} = \{+1, -1\}$ in binary classification and $\mathcal{Y} \subset \mathbb{R}$ in regression; for unsupervised learning, $\mathcal{Y}$ is a singleton: $\mathcal{Y} = \{\text{NoLabel}\}$. We define a *model* as a function[6] $f : \mathcal{X} \to \mathcal{P}$ that, given an input $x \in \mathcal{X}$, makes a *prediction*, $f(x) \in \mathcal{P}$, where $\mathcal{P}$ is a given set (for example, $\mathcal{P} = \{+1, -1\}$ in binary classification: the model predicts which class the given input belongs to). Note that the prediction set need not be the same as the outcome set, particularly for unsupervised learning tasks. The quality of a prediction is assessed by a *performance measure* (or *loss function*) $\ell : \mathcal{P} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that assigns a scalar value $\ell(p, x, y)$ to the prediction p for the pair (x, y); for example, $\ell(p, x, y) = \mathbb{I}\{p \neq y\}$ for the prediction error (misclassification rate) in binary classification (where $\mathbb{I}\{E\}$ denotes the indicator function of an event E).

[5] Formally, we assume that this is a multi-set, so there might be multiple copies of the same data point.

[6] Without loss of generality, we only consider deterministic models: we may embed any randomness required to make a prediction into the value of x, so that $f(x)$ is a deterministic mapping from $\mathcal{X}$ to $\mathcal{P}$.

Next, we define the notion of an *incremental learning algorithm.* Informally, an incremental learning algorithm is a procedure that, given a model learned from previous data points and a new dataset, updates the model to accommodate the new dataset at the fraction of the cost of training the model on the whole data from scratch. Formally, let $\mathcal{M} \subseteq \{f : \mathcal{X} \to \mathcal{P}\}$ be a set of models, and define $\mathcal{Z}^*$ to be the set of all possible datasets of all possible sizes. Disregarding computation for now, an incremental learning algorithm is a mapping $\mathcal{L} : (\mathcal{M} \cup \{\emptyset\}) \times \mathcal{Z}^* \to \mathcal{M}$ that, given a model f from $\mathcal{M}$ (or $\emptyset$ when a model does not exist yet) and a dataset $Z' = (z'_1, z'_2, \ldots, z'_m)$, returns an "updated" model $f' = \mathcal{L}(f, Z')$. To capture often needed internal states (e.g., to store learning rates), we allow the "padding" of the models in $\mathcal{M}$ with extra information as necessary, while still viewing the models as $\mathcal{X} \to \mathcal{P}$ maps when convenient. Above, f is usually the result of a previous invocation of $\mathcal{L}$ on another dataset $Z \in \mathcal{Z}^*$. In particular, $\mathcal{L}(\emptyset, Z)$ learns a model from scratch using the dataset Z. An important class of incremental algorithms are *online* algorithms, which update the model one data point at a time: to update f with Z', these algorithms make m consecutive calls to $\mathcal{L}$, where each call updates the latest model with the next remaining data point according to a random ordering of the points in Z'.

In the rest of this paper, we consider an incremental learning algorithm $\mathcal{L}$, and a fixed, given partitioning of the dataset $\{z_1, z_2, \ldots, z_n\}$ into k subsets ("chunks") $Z_1, Z_2, \ldots, Z_k$. We use $f_i = \mathcal{L}(\emptyset, Z \setminus Z_i)$ to denote the model learned from all the chunks except Z_i. Thus, the k-CV estimate of the generalization performance of $\mathcal{L}$, denoted $R_{k\text{-CV}}$, is given by

$$R_{k\text{-CV}} = \frac{1}{k} \sum_{i=1}^{k} R_i,$$

where $R_i = \frac{1}{|Z_i|} \sum_{(x,y) \in Z_i} \ell(f_i(x), x, y), i = 1, 2, \ldots, k$, is the performance of the model f_i evaluated on Z_i. The LOOCV estimate $R_{n\text{-CV}}$ is obtained when $k = n$.

2.3 Recursive Cross-Validation

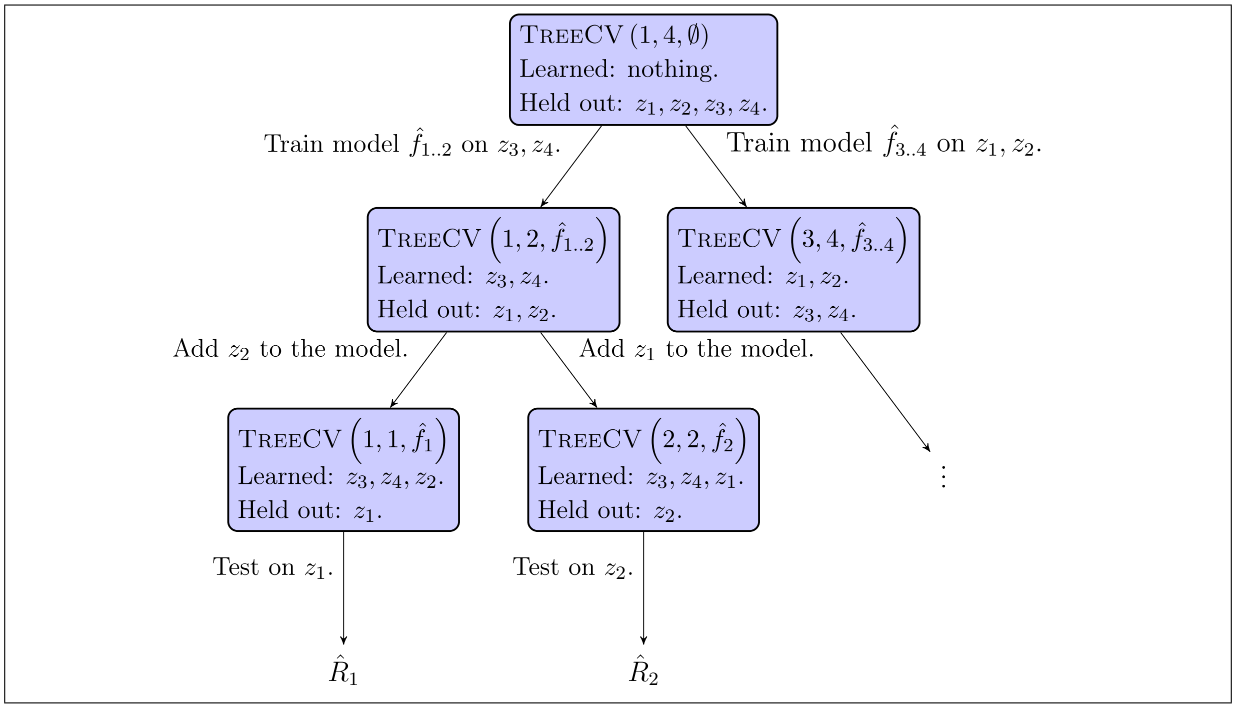
Our algorithm builds on the observation that for every i and j, $1 \le i < j \le k$, the training sets $Z \setminus Z_i$ and $Z \setminus Z_j$ are almost identical, except for the two chunks Z_i and Z_j that are held out for testing from one set but not the other. The naive k-CV calculation method ignores this fact, potentially wasting computational resources. When using an incremental learning algorithm, we may be able to exploit this redundancy: we can first learn a model

Algorithm 1: The Tree-CV recursive procedure

```
1  Function TreeCV ($s, e, \hat{f}_{s..e}$):
       Input: indices $s$ and $e$, and the model $\hat{f}_{s..e}$ trained so far.
2      if $e = s$ then
3          $\hat{R}_s \leftarrow \frac{1}{|Z_s|} \sum_{(x,y) \in Z_s} \ell\left(\hat{f}_{s..e}(x), x, y\right)$.
4          return $\frac{1}{k}\hat{R}_s$.
5      end if
6      else
7          Let $m \leftarrow \left\lfloor \frac{s+e}{2} \right\rfloor$.
8          Update the model with the chunks $Z_{m+1}, \ldots, Z_e$ to get
            $\hat{f}_{s..m} = \mathcal{L}(\hat{f}_{s..e}, Z_{m+1}, \ldots, Z_e)$.
9          Let $r \leftarrow$ TreeCV $\left(s, m, \hat{f}_{s..m}\right)$.
10         Update the model with the chunks $Z_s, \ldots, Z_m$ to get
            $\hat{f}_{m+1..e} = \mathcal{L}(\hat{f}_{s..e}, Z_s, \ldots, Z_m)$.
11         Let $r \leftarrow r +$ TreeCV $\left(m+1, e, \hat{f}_{m+1..e}\right)$.
12         return $r$.
13     end if
14 end
```

only from the examples shared between the two training sets, and then "increment" the differences into two different copies of the model learned. When the extra cost of saving and restoring a model required by this approach is comparable to learning a model from scratch, then this approach may result in a considerable speedup.

To exploit the aforementioned redundancy in training all k models at the same time, we organize the k-CV computation process in a tree structure. The resulting recursive procedure, TreeCV$(s, e, \hat{f}_{s..e})$, shown in Algorithm 1, receives two indices s and e, $1 \leq s \leq e \leq k$, and a model $\hat{f}_{s..e}$ that is trained on all chunks except $Z_s, Z_{s+1}, \ldots, Z_e$, and returns $(1/k)\sum_{i=s}^{e} \hat{R}_i$, the normalized sum of the performance scores $\hat{R}_i, i = s, \ldots, e$, corresponding to testing $\hat{f}_{i..i}$, the model trained on $Z \setminus Z_i$, on the chunk Z_i, for $i = s, \ldots, e$. TreeCV divides the hold-out chunks into two groups $Z_s, Z_{s+1}, \ldots, Z_m$ and $Z_{m+1}, \ldots Z_e$, where $m = \left\lfloor \frac{s+e}{2} \right\rfloor$ is the mid-point, and obtains the test performance scores for the two groups separately by recursively calling itself. More precisely, TreeCV first updates the model by training it on the second group of chunks, $Z_{m+1}, \ldots, Z_e$, resulting in the model $\hat{f}_{s..m}$, and makes a recursive call TreeCV$(s, m, \hat{f}_{s..m})$ to get $(1/k)\sum_{i=s}^{m} \hat{R}_i$. Then, it repeats the same procedure for the other group of chunks: starting from the original model


TreeCV $(1, 4, \emptyset)$
Learned: nothing.
Held out: z_1, z_2, z_3, z_4.
Train model $\hat{f}_{1..2}$ on z_3, z_4.
Train model $\hat{f}_{3..4}$ on z_1, z_2.
TreeCV $\left(1, 2, \hat{f}_{1..2}\right)$
Learned: z_3, z_4.
Held out: z_1, z_2.
TreeCV $\left(3, 4, \hat{f}_{3..4}\right)$
Learned: z_1, z_2.
Held out: z_3, z_4.
Add z_2 to the model.
Add z_1 to the model.
TreeCV $\left(1, 1, \hat{f}_1\right)$
Learned: z_3, z_4, z_2.
Held out: z_1.
TreeCV $\left(2, 2, \hat{f}_2\right)$
Learned: z_3, z_4, z_1.
Held out: z_2.
Test on z_1.
Test on z_2.
$\hat{R}_1$
$\hat{R}_2$
Figure 2.1: An example run of TreeCV on a dataset of size four, calculating the LOOCV estimate.

$\hat{f}_{s..e}$ it had received, it updates the model, this time using the first group of the remaining chunks, $Z_s, \ldots, Z_m$, that were previously held out, and calls TreeCV$(m+1, e, \hat{f}_{m+1..e})$ to get $(1/k)\sum_{i=m+1}^{e} \hat{R}_i$ (for the second group of chunks). The recursion stops when there is only one hold-out chunk ($s = e$), in which case the performance score $\hat{R}_s$ of the model $\hat{f}_{s..s}$ (which is now trained on all the chunks except for Z_s) is directly calculated and returned. Calling TreeCV$(1, n, \emptyset)$ calculates $\hat{R}_{k\text{-CV}} = \frac{1}{k}\sum_{i=1}^{k} \hat{R}_i$. Figure 2.1 shows an example of the recursive call tree underlying a run of the algorithm calculating the LOOCV estimate on a dataset of four data points. Note that the tree structure imposes a new order of feeding the chunks to the learning algorithm, e.g., z_3 and z_4 are learned before z_2 in the first branch of the tree.

2.3.1 Accuracy of TreeCV

To simplify the analysis, in this section and the next, we assume that each chunk is of the same size, that is $n = kb$ for some integer $b \geq 1$.

Note that the models $\hat{f}_{s..s}$ used in computing $\hat{R}_s$ are learned incrementally. If the learning algorithm learns the same model no matter whether it is given the chunks all at once or gradually, then $\hat{f}_{s..s}$ is the same as the model f_s used in the definition of $R_{k\text{-CV}}$, and $R_{k\text{-CV}} = \hat{R}_{k\text{-CV}}$. If this assumption does not hold, then $\hat{R}_{k\text{-CV}}$ is still close to $R_{k\text{-CV}}$ as long as the models $\hat{f}_{s..s}$ are sufficiently similar to their corresponding models f_s. In the rest of this section, we formalize this assertion.

First, we define the notion of stability for an incremental learning algorithm. Intuitively, an incremental learning algorithm is stable if the performance of the models are nearly the same no matter whether they are learned incrementally or in batch. Formally, suppose that a dataset $\{z_1, \ldots, z_n\}$ is partitioned into $l+1$ nonempty chunks Z^{test} and $Z_1^{\text{train}}, \ldots, Z_l^{\text{train}}$, and we are using Z^{test} as the test data and the chunks $Z_1^{\text{train}}, \ldots, Z_l^{\text{train}}$ as the training data. Let $f^{\text{batch}} = \mathcal{L}(\emptyset, Z_1^{\text{train}} \cup \ldots \cup Z_l^{\text{train}})$ denote the model learned from the training data when provided all at the same time, and let

$$f^{\text{inc}} = \mathcal{L}\Bigg(\mathcal{L}\Bigg(\ldots\Big(\mathcal{L}(\emptyset, Z_1^{\text{train}}), Z_2^{\text{train}}\Big), \ldots, Z_{l-1}^{\text{train}}\Bigg), Z_l^{\text{train}}\Bigg)$$

denote the model learned from the same chunks when they are provided incrementally to $\mathcal{L}$. Let $R^{\text{test}}(f) = \frac{1}{|Z^{\text{test}}|}\sum_{(x,y)\in Z^{\text{test}}} \ell\left(f(x), x, y\right)$ denote the performance of a model f on the test data Z^{test}.

Definition 2.1 (Incremental stability)**.** *The algorithm $\mathcal{L}$ is g-incrementally stable for a function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ if, for any dataset $\{z_1, z_2, \ldots, z_n\}$, $b < n$, and partition $Z^{\text{test}}, Z_1^{\text{train}}, \ldots, Z_l^{\text{train}}$ with nonempty cells $Z_i^{\text{train}}, 1 \le i \le l$ and $|Z^{\text{test}}| = b$, the test performance of the models f^{batch} and f^{inc} defined above satisfy*

$$\left|R^{\text{test}}(f^{\text{inc}}) - R^{\text{test}}(f^{\text{batch}})\right| \le g\left(n-b, b\right).$$

If the data $\{z_1, \ldots, z_n\}$ is drawn independently from the same distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and/or the learning algorithm $\mathcal{L}$ is randomized, we say that $\mathcal{L}$ is g-incrementally stable in expectation if

$$\left|\mathbb{E}\{R^{\text{test}}(f^{\text{inc}})\} - \mathbb{E}\{R^{\text{test}}(f^{\text{batch}})\}\right| \le g\left(n-b, b\right)$$

for all partitions selected independently of the data and the randomization of $\mathcal{L}$.

The following statement is an immediate consequence of the above definition:

Theorem 2.1. *Suppose $n = bk$ for some integer $b \ge 1$ and that algorithm $\mathcal{L}$ is g-incrementally stable. Then,*

$$\left|\hat{R}_{k\text{-CV}} - R_{k\text{-CV}}\right| \le g\left(n-b, b\right).$$

If $\mathcal{L}$ is g-incrementally stable in expectation then

$$\left|\mathbb{E}\left\{\hat{R}_{k\text{-CV}}\right\} - \mathbb{E}\{R_{k\text{-CV}}\}\right| \leq g\,(n-b, b)\,.$$

Proof. We prove the first statement only, the proof of the second part is essentially identical. Recall that $Z_j, j = 1, 2, \ldots, k$ denote the chunks used for cross-validation. Fix i and let $l = \lceil \log k \rceil$. Let $Z^{\text{test}} = Z_i$ and $Z_j^{\text{train}}, j = 1 \ldots l$, denote the union of the chunks used for training at depth j of the recursion branch ending with the computation of $\hat{R}_i$. Then, by definition, $\hat{R}_i = R^{\text{test}}(f^{\text{inc}})$ and $R_i = R^{\text{test}}(f^{\text{batch}})$. Therefore, $|\hat{R}_i - R_i| \leq g\,(n-b, b)$, and the statement follows since $\hat{R}_{k\text{-CV}}$ and $R_{k\text{-CV}}$ are defined as the averages of the $\hat{R}_i$ and R_i, respectively. □

It is then easy to see that incremental learning methods with a bound on their excess risk are incrementally stable in expectation.

Theorem 2.2. *Suppose the data $\{z_1, \ldots, z_n\}$ is drawn independently from the same distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. Let $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ be drawn from $\mathcal{D}$ independently of the data and let $f^* \in \arg\min_{f \in \mathcal{M}} \mathbb{E}\{\ell(f(X), X, Y)\}$ denote a model in $\mathcal{M}$ with minimum expected loss. Assume there exist upper bounds $m^{\text{batch}}(n-b)$ and $m^{\text{inc}}(n-b)$ on the excess risks of f^{batch} and f^{inc}, trained on $n' = n - b$ data points, such that*

$$\mathbb{E}\big\{\ell(f^{\text{batch}}(X), X, Y) - \ell(f^*(X), X, Y)\big\} \leq m^{\text{batch}}(n')$$

and

$$\mathbb{E}\big\{\ell(f^{\text{inc}}(X), X, Y) - \ell(f^*(X), X, Y)\big\} \leq m^{\text{inc}}(n')$$

for all n and for every partitioning of the dataset that is independent of the data, (X, Y), and the randomization of $\mathcal{L}$. Then $\mathcal{L}$ is incrementally stable in expectation w.r.t. the loss function ℓ, with $g(n', b) = \max\{m^{\text{batch}}(n'), m^{\text{inc}}(n')\}$.

Proof. Since the data points in the sets $Z_1^{\text{train}}, \ldots, Z_l^{\text{train}}$ and Z^{test} are independent, f^{batch} and f^{inc} are both independent of Z^{test}. Hence, $\mathbb{E}\big\{R^{\text{test}}(f^{\text{batch}})\big\} = \mathbb{E}\big\{\ell\left(f^{\text{batch}}(X), X, Y\right)\big\}$ and $\mathbb{E}\big\{R^{\text{test}}(f^{\text{inc}})\big\} = \mathbb{E}\big\{\ell\left(f_n^{\text{inc}}(X), X, Y\right)\big\}$. Therefore,

$$\begin{aligned}&\mathbb{E}\big\{R^{\text{test}}(f^{\text{inc}})\big\} - \mathbb{E}\big\{R^{\text{test}}(f^{\text{batch}})\big\}\\ &= \mathbb{E}\big\{R^{\text{test}}(f^{\text{inc}})\big\} - \mathbb{E}\{\ell(f^*(X), X, Y)\}\end{aligned}$$

$$+\mathbb{E}\{\ell(f^*(X),X,Y)\}-\mathbb{E}\big\{R^{\text{test}}(f^{\text{batch}})\big\}$$
$$\le \mathbb{E}\big\{R^{\text{test}}(f^{\text{inc}})\big\}-\mathbb{E}\{\ell(f^*(X),X,Y)\}\le m^{\text{inc}}(n')$$

where we used the fact that f^* is optimal. Similarly, $\mathbb{E}\big\{R^{\text{test}}(f^{\text{batch}})\big\}-\mathbb{E}\big\{R^{\text{test}}(f^{\text{inc}})\big\}\le m^{\text{batch}}(n')$, finishing the proof. □

In particular, for online learning algorithms satisfying some regret bound, standard online-to-batch conversion results [15], [49] yield excess-risk bounds for independent and identically distributed data. Similarly, excess-risk bounds are often available for stochastic gradient descent (SGD) algorithms which scan the data once (see, e.g., [74]). For online learning algorithms (including single-pass SGD), the batch version is usually defined by running the algorithm using a random ordering of the data points or sampling from the data points with replacement. Typically, this version also satisfies the same excess-risk bounds. Thus, the previous theorem shows that these algorithms are incrementally stable with $g(n,b)$ being the excess-risk bound for n samples.

Note that this incremental stability is w.r.t. the loss function whose excess-risk is bounded. For example, after visiting n data points, the regret of PEGASOS [99] with bounded features is bounded by $O(\log(n))$. Using the online-to-batch conversion of Kakade and Tewari [49], this gives an excess risk bound $m(n)=O(\log(n)/n)$, and hence PEGASOS is stable w.r.t. the regularized hinge loss with $g(n,b)=m(n)=O(\log(n)/n)$. Similarly, SGD over a compact set with bounded features and a bounded convex loss is stable w.r.t. that convex loss with $g(n,b)=O(1/\sqrt{n})$ [74]. Experiments with these algorithms are shown in Section 2.5. Finally, we note that algorithms like PEGASOS or SGD could also be used to scan the data multiple times. In such cases, these algorithms would not be useful incremental algorithms, as it is not clear how one should add a new data point without a major retraining over the previous points. Currently, our method does not apply to such cases in a straightforward way.

2.4 Complexity Analysis

In this section, we analyze the running time and storage requirements of TreeCV, and discuss some practical issues concerning its implementation, including parallelization.

2.4.1 Memory Requirements

Efficient storage of and updates to the model are crucial for the efficiency of Algorithm 1: Indeed, in any call of TreeCV$(s, e, \hat{f}_{s..e})$ that does not correspond to simply evaluating a model on a chunk of data (i.e., $s \neq e$), TreeCV has to update the original model $\hat{f}_{s..e}$ twice, once with $Z_s, \dots, Z_m$, and once with $Z_{m+1}, \dots, Z_e$. To do this, TreeCV can either store $\hat{f}_{s..e}$, or revert to $\hat{f}_{s..e}$ from $\hat{f}_{s..m}$. In general, for any type of model, if the model for $\hat{f}_{s..e}$ is modified in-place, then we need to create a copy of it before it is updated to the model for $\hat{f}_{s..m}$, or, alternatively, keep track of the changes made to the model during the update. Whether to use the copying or the save/revert strategy depends on the application and the learning algorithm. For example, if the model state is compact, copying is a useful strategy, whereas when the model undergoes few changes during an update, save/revert might be preferred.

Compared to a single run of the learning algorithm $\mathcal{L}$, TreeCV requires some extra storage for saving and restoring the models it trains along the way. When no parallelization is used in implementing TreeCV, we are in exactly one branch at every point during the execution of the algorithm. Since the largest height of a recursion branch is of $O(\log k)$, and one model (or the changes made to it) is saved in each level of the branch, the total storage required by TreeCV is $O(\log(k))$-times the storage needed for a single model.

TreeCV can be easily parallelized by dedicating one thread of computation to each of the data groups used in updating $\hat{f}_{s..e}$ in one call of TreeCV. In this case one typically needs to copy the model since the two threads need to be able to run independently of each other; thus, the total number of models TreeCV needs to store is $O(k)$, since there are $2k - 1$ total nodes in the recursive call tree, with exactly one model stored per node. Note that a standard parallelized CV calculation also needs to store $O(k)$ models.

Finally, note that TreeCV is potentially useful in a distributed environment, where each chunk of the data is stored on a different node in the network. Updating the model on a given chunk can then be relegated to that computing node (the model is sent to the processing node, trained and sent back, i.e., this is not using all the nodes at once), and it is only the model (or the updates made to the model), not the data, that needs to be communicated to the other nodes. Since at every level of the tree, each chunk is added to exactly one model, the total communication cost of doing this is $O(k \log(k))$.

Running Time

Next, we analyze the time complexity of TreeCV when calculating the k-CV score for a dataset of size n under our previous simplifying assumption that $n = bk$ for some integer $b \geq 1$.

The running time of TreeCV is analyzed in terms of the running time of the learning algorithm $\mathcal{L}$ and the time it takes to copy the models (or to save and then revert the changes made to it while it is being updated by $\mathcal{L}$). Throughout this subsection, we use the following definitions and notations: for $m = 0, 1, \ldots, n$, $l = 1, \ldots, n - m$, and $j = 1, \ldots, k$,

- $t_u(m, l) \geq 0$ denotes the time required to update a model, already trained on m data points, with a set of l additional data points;
- $t_s(m, l) \geq 0$ is the time required to copy the model, (or save and revert the changes made to it) when the model is already trained on m data points and is being updated with l more data points;
- $t(j)$ is the time spent in saving, restoring, and updating models in a call to TreeCV $\left(s, e, \hat{f}_{s..e}\right)$ with $j = e - s + 1$ hold-out chunks (and with $\hat{f}_{s..e}$ trained on $k - j$ chunks);
- t_ℓ denotes the time required to test a model on one of the k chunks (where the model is trained on the other $k - 1$ chunks);
- $T(j)$ denotes the *total* running time of TreeCV$(s, e, \hat{f}_{s..e})$ when the number of chunks held out is $j = e - s + 1$, and $\hat{f}_{s..e}$ is already trained with $n - bj$ data points. Note that $T(k)$ is the total running time of TreeCV to calculate the k-CV score for a dataset of size n.

By definition, for all $j = 2 \ldots k$, we have

$$\begin{aligned} t(j) = t_u(n - bj, b\lfloor j/2 \rfloor) + t_s(n - bj, b\lfloor j/2 \rfloor) \\ + t_u(n - bj, b\lceil j/2 \rceil) + t_s(n - bj, b\lceil j/2 \rceil) + t_c, \end{aligned}$$

where $t_c \geq 0$ accounts for the cost of the operations other than the recursive function calls.

We will analyze the running time of TreeCV under the following natural assumptions: First, we assume that $\mathcal{L}$ is not slower if data points are provided in batch rather than one by one. That is,

$$t_u(m,l) \le \sum_{i=m}^{m+l-1} t_u(i,1), \tag{2.1}$$

for all $m = 0,\dots,n$ and $l = 1,\dots,n-m$.[7] Second, we assume that updating a model requires work comparable to saving it or reverting the changes made to it during the update. This is a natural assumption since the update procedure is also writing those changes. Formally, we assume that there is a constant $c \ge 0$ (typically $c < 1$) such that for all $m = 0,\dots,n$ and $l = 1,\dots,n-m$,

$$t_s(m,l) \le c\, t_u(m,l). \tag{2.2}$$

To get a quick estimate of the running time, assume for a moment the idealized case that $k = 2^d$, $t_u(m,l) = l t_u(0,1)$ for all m and l, and $t_c = 0$. Since $n2^{-j}$ data points are added to the models of a node at level j in the recursive call tree, the work required in such a node is $(1+c)n2^{-j}t_u(0,1)$. There are 2^j such nodes, hence the cumulative running time at level j nodes is $(1+c)nt_u(0,1)$, hence the total running time of the algorithm is $(1+c)nt_u(0,1)\log_2 k$, where $\log_2$ denotes base-2 logarithm.

The next theorem establishes a similar logarithmic penalty (compared to the running time of feeding the algorithm with one data point at a time) in the general case.

Theorem 2.3. *Assume* (2.1) *and* (2.2) *are satisfied. Then the total running time of* TreeCV *can be bounded as*

$$T(k) \le n(1+c)t_u^* \log_2(2k) + (k-1)t_c + kt_\ell,$$

where $t_u^* = \max_{0 \le i \le n-1} t_u(i,1)$.

Proof. By (2.1), $t_u(n-bj,l) \le \sum_{i=0}^{l-1} t_u(n-bj+i,1) \le l\, t_u^*$ for all $l = 1,\dots,bj$. Combining with (2.2), for any $2 \le j \le k$ we obtain

$$t(j) \le \ (1+c)t_u(n-bj, b\lfloor j/2 \rfloor)$$

[7]If this is not the case, we would always input the data one by one even if there are more data points available.

$$
\begin{aligned}
&+\ (1+c)t_u(n-bj, b\lceil j/2\rceil) + t_c \\
&\le (1+c)bt_u^*\left(\lfloor j/2\rfloor + \lceil j/2\rceil\right) + t_c \\
&= \frac{(1+c)n}{k}\, j\ t_u^* + t_c := aj + t_c \qquad (2.3)
\end{aligned}
$$

where $a = (1+c)nt_u^*/k$. Next we show by induction that for $j \ge 2$ this implies

$$T(j) \le aj(\log_2(j-1)+1) + (j-1)t_c + jt_\ell. \qquad (2.4)$$

Substituting $j = k$ in (2.4) proves the theorem since $\log_2(j-1)+1 \le \log_2(2j)$. By the definition of TreeCV,

$$T(j) = \begin{cases} T\left(\lfloor \frac{j}{2}\rfloor\right) + T\left(\lceil \frac{j}{2}\rceil\right) + t(j), & j \ge 2; \\ t_\ell, & j = 1. \end{cases}$$

This implies that (2.4) holds for $j = 2, 3$. Assuming (2.4) holds for all $2 \le j' < j$, $4 \le j \le k$, from (2.3) we get

$$
\begin{aligned}
T(j) &= T(\lfloor j/2\rfloor) + T(\lceil j/2\rceil) + t(j) \\
&\le aj\left(\log_2(\lceil j/2\rceil - 1) + 2\right) + t_c(j-1) + jt_\ell \\
&\le aj(\log_2(j-1)+1) + t_c(j-1) + jt_\ell
\end{aligned}
$$

completing the proof of (2.4). □

For fully incremental, linear-time learning algorithms (such as PEGASOS or single-pass SGD), we obtain the following upper bound:

Corollary 2.1. *Suppose that the learning algorithm $\mathcal{L}$ satisfies the property given by* (2.2) *and $t_u(0,m) = mt_u^*$ for some $t_u^* > 0$ and all $1 \le m \le n$. Then*

$$T(k) \le (1+c)T_{\mathcal{L}} \log_2(2k) + t_c(k-1) + kt_\ell,$$

where $T_{\mathcal{L}} = t_u(0,n)$ is the running time of a single run of $\mathcal{L}$.

2.5 Experiments

In this section we evaluate TreeCV and compare it with the standard (k-repetition) CV calculation. We consider two incremental algorithms: linear PEGASOS [99] for

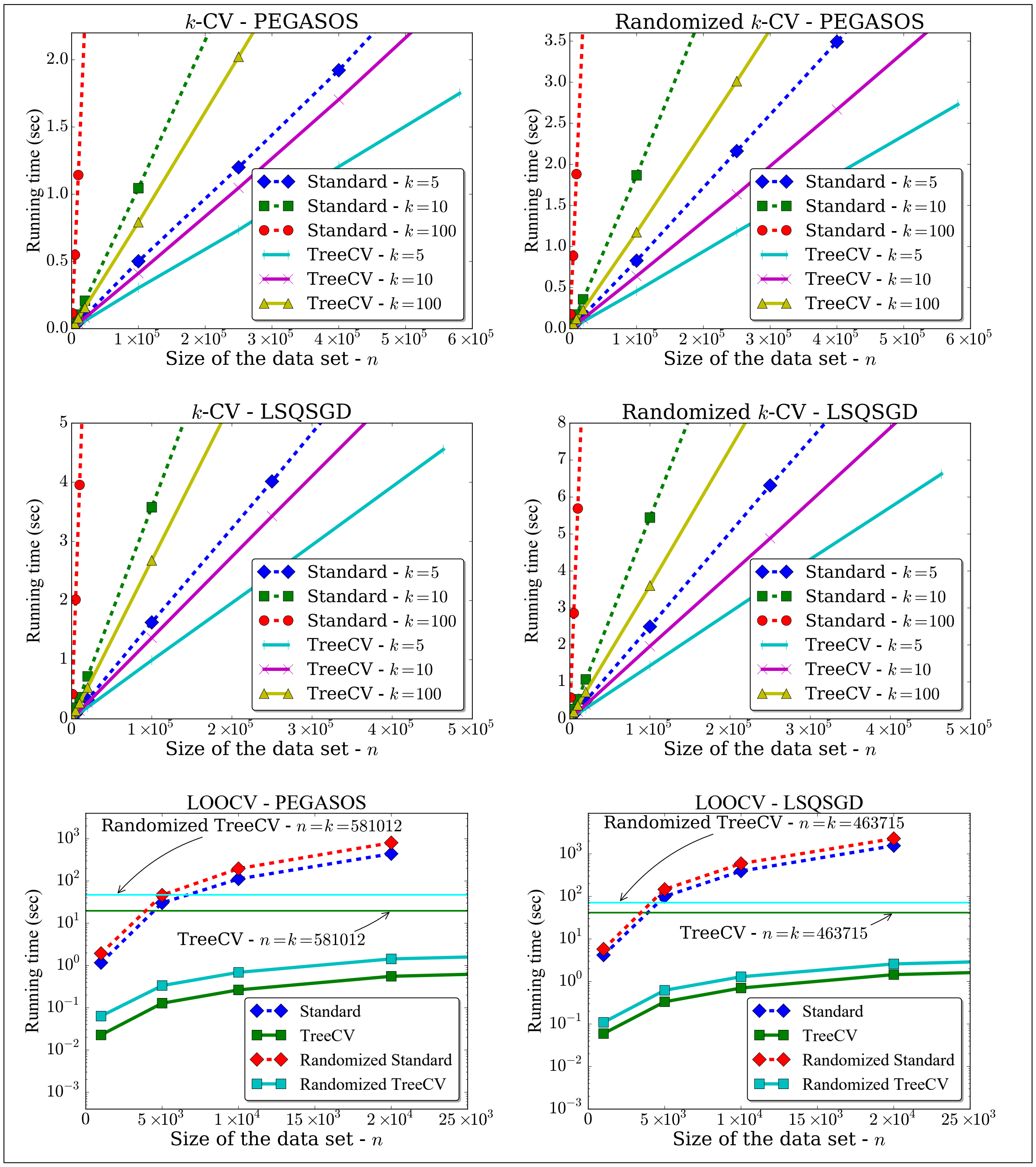


Figure 2.2: Running time of TreeCV and standard k-CV for different values of k as a function of the number of data points n, averaged over 100 independent repetitions, with and without random permutation of data points. Top row: PEGASOS; middle row: least-square SGD; bottom row: Leave-one-out.

SVM classification, and least-square stochastic gradient descent (LSQSGD) for linear least-squares regression (more precisely, LSQSGD is the robust stochastic approximation algorithm of Nemirovski *et al.* [74] for the squared loss and parameter vectors constrained in the unit l_2-ball). Following the suggestions in the original papers, we take the last hypothesis from PEGASOS and the average hypothesis from LSQSGD as our model. We focus on the large-data regime in which the algorithms learn from the data in a single pass.

The algorithms were implemented in Python/Cython and Numpy. The tests were run on a single core of a computer with an Intel Xeon E5430 processor and 20 GB of RAM. We used datasets from the UCI repository [60], downloaded from the LibSVM website [17].

We tested PEGASOS on the UCI Covertype dataset (581,012 data points, 54 features, 7 classes), learning class “1” against the rest of the classes. The features were scaled to have unit variance. The regularization parameter was set to $\lambda = 10^{-6}$ following the suggestion of Shalev-Shwartz *et al.* [99]. For LSQSGD, we used the UCI YearPredictionMSD dataset (463,715 data points, 90 features) and, following the suggestion of Nemirovski *et al.* [74], set the step-size to $\alpha = n^{-1/2}$. The target values where scaled to $[0, 1]$.

Naturally, PEGASOS and LSQSGD are sensitive to the order in which data points are provided (although they are incrementally stable as mentioned after Theorem 2.2). In a vanilla implementation, the order of the data points is fixed in advance for the whole CV computation. That is, there is a fixed ordering of the chunks and of the samples within each chunk, and if we need to train a model with chunks $Z_{i_1}, \dots, Z_{i_j}$, the data points are given to the training algorithm according to this hierarchical ordering. This introduces certain dependence in the CV estimation procedure: for example, the model trained on chunks $Z_1, \dots, Z_{k-1}$ has visited the data in a very similar order to the one trained on $Z_1, \dots, Z_{k-2}, Z_k$ (except for the last n/k steps of the training). To eliminate this dependence, we also implemented a randomized version in which the samples used in a training phase are provided in a random order (that is, we take all the data points for the chunks $Z_{i_1}, \dots, Z_{i_j}$ to be used, and feed them to the training algorithm in a random order).

Table 2.2 shows the values of the CV estimates computed under different scenarios. It can be observed that the standard (k-repetition) CV method is quite sensitive to the order of the points: the variance of the estimate does not really decay as the number of folds k increases, while we see the expected decay for the randomized version. On the

CV estimates for PEGASOS (misclassification rate $\times 100$)				
	TreeCV		Standard	
	fixed	randomized	fixed	randomized
$k = 5$	30.682 ± 1.2127	30.839 ± 0.9899	30.825 ± 1.9248	30.768 ± 1.1243
$k = 10$	30.665 ± 0.8299	30.554 ± 0.7125	30.767 ± 1.7754	30.541 ± 0.7993
$k = 100$	30.677 ± 0.3040	30.634 ± 0.2104	30.636 ± 2.0019	30.624 ± 0.2337
$k = n$	30.640 ± 0.0564	30.637 ± 0.0592	N/A	N/A

CV estimates for LSQSGD (squared error $\times 100$)				
	TreeCV		Standard	
	fixed	randomized	fixed	randomized
$k = 5$	25.299 ± 0.0019	25.298 ± 0.0018	25.299 ± 0.0019	25.299 ± 0.0017
$k = 10$	25.297 ± 0.0016	25.297 ± 0.0015	25.297 ± 0.0016	25.297 ± 0.0016
$k = 100$	25.296 ± 0.0012	25.296 ± 0.0013	25.296 ± 0.0011	25.296 ± 0.0013
$k = n$	25.296 ± 0.0012	25.296 ± 0.0012	N/A	N/A

Table 2.2: k-CV performance estimates averaged over 100 repetitions (and their standard deviations), for the full datasets with and without data repermutation: PEGASOS (top) and LSQSGD (bottom).

other hand, the non-randomized version of TreeCV does not show such a behavior, as the automatic re-permutation that happens during TreeCV might have made the k folds less correlated. However, randomizing the order of the training points typically reduces the variance of the TreeCV-estimate, as well.

Figure 2.2 shows the running times of TreeCV and the standard CV method, as a function of n, for PEGASOS (top row) and LSQSGD (middle row). The columns show the running times for different values of k, with and without randomizing the order of the data points (right and left column, resp.). The bottom row shows the the running time (log-scale) for LOOCV calculations. TreeCV outperforms the standard method in all of the cases. It is notable that TreeCV makes the calculation of LOOCV practical even for $n = 581{,}012$, in a fraction of the time required by the standard method at $n = 10{,}000$: for example, for PEGASOS, TreeCV takes around 20 seconds (46 when randomized) for computing LOOCV at $n = 581{,}012$, while the standard method takes around 124 seconds (175 when randomized) at $n = 10{,}000$. Furthermore, one can see that the variance reduction achieved by randomizing the data points comes at the price of a constant factor bigger running time (the factor is around 1.5 for the standard method, and 2 for TreeCV).

This comes from the fact that both the training time and the time of generating a random perturbation is linear in the number of points (assuming generating a random number uniformly from $\{1, \ldots, n\}$ can be done in constant time).

2.6 Conclusion

We presented a general method, TreeCV, to speed up cross-validation for incremental learning algorithms. The method is applicable to a wide range of supervised and unsupervised learning settings. We showed that, under mild conditions on the incremental learning algorithm being used, TreeCV computes an accurate approximation of the k-CV estimate, and its running time scales logarithmically in k (the number of CV folds), while the running time of the standard method of training k separate models scales linearly with k.

Experiments on classification and regression, using two well-known incremental learning algorithms, PEGASOS and least-square SGD, confirmed the speedup and predicted accuracy. When the model learned by the learning algorithm depends on whether the data is provided incrementally or in batch (or on the order of the data, as in the case of online algorithms), the CV estimate calculated by our method was still close to the CV computed by the standard method, but with a lower variance.

Chapter 3

Delay-Tolerant Online Convex Optimization: Unified Analysis & Adaptive Gradient Algorithms

Abstract

We present a unified, black-box-style method for developing and analyzing online convex optimization (OCO) algorithms for full-information online learning in delayed-feedback environments. Our new, simplified analysis enables us to substantially improve upon previous work and to solve a number of open problems from the literature. Specifically, we develop and analyze asynchronous AdaGrad-style algorithms from the Follow-the-Regularized-Leader (FTRL) and Mirror-Descent family that, unlike previous works, can handle projections and adapt both to the gradients and the delays, without relying on either strong convexity or smoothness of the objective function, or data sparsity. Our unified framework builds on a natural reduction from delayed-feedback to standard (non-delayed) online learning. This reduction, together with recent unification results for OCO algorithms, allows us to analyze the regret of generic FTRL and Mirror-Descent algorithms in the delayed-feedback setting in a unified manner using standard proof techniques. In addition, the reduction is exact and can be used to obtain both upper and lower bounds on the regret in the delayed-feedback setting.[1]

[1] This chapter and the related appendices have been published as the following conference paper:

- P. Joulani *et al.*, "Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms," in *Proceedings of the 30th Conference on Artificial Intelligence (AAAI-16)*, 2016.

3.1 Introduction

Online learning algorithms are at the heart of modern machine learning algorithms. The sequential nature of these algorithms makes them ideal for learning from data that is too large to be processed in a batch mode. However, their very sequential nature also suggests that they may be unfit to be used in parallel and distributed processing environments. To address this potential issue, several papers have studied asynchronous and distributed versions of online learning and stochastic optimization algorithms in recent years (see Section 3.3). These papers have shown that the ability to tolerate delays in receiving feedback is a key for obtaining asynchronous online learning algorithms.

Depending on the specifics of a machine learning task, a user can choose from several online learning algorithms (see, e.g., the book of Cesa-Bianchi and Lugosi [16]). Previous work has typically focused on extending these algorithms to various delayed-feedback scenarios on a case-by-case basis, and usually under the stochastic optimization setting. However, many ideas and core challenges of delay-tolerant online learning are common across different domains and algorithms. In this paper, we take a different approach: we propose a unified theoretical framework for analyzing full-information online learning algorithms under delayed feedback. This unified approach enables us to simultaneously analyze various online convex optimization (OCO) methods (with linear losses or implicit updates) in the delayed-feedback setting, without the need to resort to ad-hoc analysis techniques.

The framework that we present is based on a natural reduction from the delayed-feedback online learning problem to standard, non-delayed online learning, as well as on recent unified analyses of OCO algorithms [26], [67]. In particular, our first main result gives an easy-to-interpret *identity* relating the regret of an algorithm operated in a delayed environment to the regret of the algorithm when operated in a non-delayed environment. All of our subsequent results are then derived from this general identity. We demonstrate the flexibility and power of our framework by solving several open problems from the literature. In particular, we analyze general delay-adaptive AdaGrad-style algorithms, both with and without projection, without relying on either strong convexity or smoothness of the loss function, or data sparsity.

The rest of this paper is organized as follows. We start with the formal definition of the learning setting we consider (Section 3.2), followed by a summary of our results and their connection to the previous work (Section 3.3). We present our general reduction in Section 3.4 and the unified analysis of OCO algorithms in Section 3.5. Section 3.6 demonstrates the application of our framework in solving the aforementioned open problems. We conclude the paper in Section 3.7 and discuss some potential future work.

3.1.1 Notation and definitions

We will work with a closed, convex, non-empty subset $\mathcal{X}$ of a Hilbert space $\mathbb{X}$ over the reals. That is, $\mathbb{X}$ is a real vector space equipped with an inner product $\langle \cdot, \cdot \rangle$ that is complete with respect to (w.r.t.) the norm induced by $\langle \cdot, \cdot \rangle$. For example, we might have $\mathbb{X} = \mathbb{R}^d$ where $\langle \cdot, \cdot \rangle$ is the dot-product, or $\mathbb{X} = \mathbb{R}^{m \times n}$, the set of $m \times n$ real matrices, where $\langle A, B \rangle = \operatorname{tr}\left(A^\top B\right)$. Let $\mathcal{R} : S \to \mathbb{R}$ be a strictly convex, differentiable function over a convex closed domain $S \subset \mathbb{X}$ with a non-empty interior S°. Then, the $\mathcal{R}$-induced Bregman divergence between the points $x \in S$, $y \in S^\circ$ is defined as $\mathcal{B}_\mathcal{R}(x, y) = \mathcal{R}(x) - \mathcal{R}(y) - \langle \nabla \mathcal{R}(y), x - y \rangle$. The function $\mathcal{R}$ is α-strongly convex with respect to a norm $\|\cdot\|$ on S if $\mathcal{B}_\mathcal{R}(x, y) \geq (\alpha/2)\|x - y\|^2$ for all $x \in S, y \in S^\circ$. The indicator of an event $\mathcal{E}$ is denoted by $\mathbb{I}\{\mathcal{E}\}$. For any sequence $c_i, c_{i+1}, \ldots, c_j$, we use $c_{i:j}$ to denote its sum, and we define $c_{i:j} = 0$ when $i > j$. For any function f, we denote the set of all sub-gradients of f at x by $\partial f(x)$, and use $f'(x)$ to denote any member of $\partial f(x)$.

3.2 Problem setting

We consider prediction under delayed feedback in an online convex optimization setting, building on the delayed-feedback online learning framework of Joulani *et al.* [44].[2] Let $\mathcal{F} \subset \{f : \mathcal{X} \to \mathbb{R}\}$ be a set of convex functions. The pair $(\mathcal{X}, \mathcal{F})$ defines the sequential prediction game shown in Figure 3.1. The game consists of a forecaster making predictions against a fixed (but unknown) sequence of loss functions $f_1, f_2, \ldots, f_n \in \mathcal{F}$, possibly chosen in an adversarial manner before the game starts. In every round $t = 1, 2, \ldots, n$ of the game, the forecaster makes a prediction $x_t \in \mathcal{X}$ based on the feedback (specified below) that it has observed in rounds $1, \ldots, t-1$, and suffers the loss $f_t(x_t)$. The goal of the

[2]Note, however, that the reduction in Section 3.4 applies more generally to full-information online learning, not just OCO.

forecaster is to minimize its total loss compared to the loss of the best constant prediction $x^* \in \mathcal{X}$. More precisely, with the *regret* against an arbitrary prediction $x \in \mathcal{X}$ defined as

$$R_n(x) = \sum_{t=1}^{n} f_t(x_t) - f_t(x) \,,$$

the goal of the forecaster is to minimize $R_n(x^*)$, where $x^* = \arg\min_{x \in \mathcal{X}} \sum_{t=1}^{n} f_t(x)$ is the best prediction in hindsight.

The feedback based on which the forecaster can make prediction x_t is a subset of the loss functions from the previous rounds, $f_1, f_2, \ldots, f_{t-1}$. In particular, in a *non-delayed* setting, in each round $s = 1, \ldots, n$, the forecaster always observes f_s before the end of the round; thus, the forecaster can make the prediction x_t based on $f_1, f_2, \ldots, f_{t-1}$ (we will call an algorithm non-delayed if it is designed for this setting). On the other hand, in the *delayed-feedback* setting that we consider in this paper, the forecaster observes f_t only after a delay of (say) τ_t time steps, at the end of round $t + \tau_t$, where we assume that the delays $\tau_1, \tau_2, \ldots, \tau_n$ are fixed (but unknown) non-negative integers. Hence, after predicting x_t in time step t, the forecaster in a delayed-feedback setting observes $H_t = \{f_s : 1 \le s \le t, s + \tau_s = t\}$, the multi-set of loss functions from rounds $1, 2, \ldots, t$ that arrive at the end of time step t. As such, the prediction x_t can be based only on the observed loss functions $\cup_{s=1}^{t-1} H_s$, *i.e.*, based on the subset $\{f_s : 1 \le s \le t-1, s + \tau_s < t\} \subset \{f_1, f_2, \ldots, f_{t-1}\}$ of the loss functions from rounds $1, 2, \ldots, t-1$. Note that the non-delayed setting corresponds to the special case when $\tau_t = 0$ and $H_t = \{f_t\}$ for all $t = 1, 2, \ldots, n$. Finally, note that the delays can reorder the feedbacks, so that the feedback f_t of the interaction at time step $t < t'$ might arrive after feedback $f_{t'}$; this can happen when $t + \tau_t \ge t' + \tau_{t'}$. Note that the feedback does not include the index of the round the loss function corresponds to, *i.e.*, the feedback is not time-stamped.

3.3 Contributions and related work

Our first contribution is providing a unified framework for analyzing the regret of OCO algorithms under delayed feedback. Our proof-technique has two main steps:

1- Black-box reduction: First, we show (Theorem 3.1) that when any deterministic non-delayed full-information online learning algorithm is used (without modification) in a delayed-feedback environment, the additional regret the algorithm suffers compared to

The environment chooses a sequence of convex loss functions $f_1, \ldots, f_n \in \mathcal{F}$.
Repeat: for each time step $t = 1, 2, \ldots, n$:

1. The forecaster makes a prediction $x_t \in \mathcal{X}$.
2. The forecaster incurs loss $f_t(x_t)$ and receives the set of feedbacks $H_t = \{f_s : s + \tau_s = t\}$.

Goal: minimize $\sup_{x \in \mathcal{X}} R_n(x)$.

Figure 3.1: Delayed-feedback Online Convex Optimization

running in a non-delayed environment depends on its "prediction drift", a quantity that roughly captures how fast the algorithm changes its predictions (Definition 3.1).

2- Unified bounds on the prediction drift: Next, we derive upper bounds (Propositions 3.1 and 3.2) on the prediction drift of two generic non-delayed OCO algorithms, which we call FTRL-PROX and ADA-MD, and obtain general delayed-feedback regret bounds (Theorem 3.4) for these algorithms by combining their drift bounds with the reduction result of Theorem 3.1. These two algorithms generalize, respectively, the Follow-The-Regularized-Leader and the Mirror-Descent classes of OCO algorithms and include various ADAGRAD-style algorithms as special cases.

Our second contribution is to develop, using the new framework mentioned above, FTRL- and Mirror-Descent-based ADAGRAD-style OCO methods (Section 3.6) that can adapt both to the observed gradients and the observed delays and can handle projection. These contributions solve a number of open problems in the literature, as follows:

Problem posed by McMahan and Streeter [66]: In a recent paper, McMahan and Streeter [66] provide a delay-adaptive ADAGRAD-style extension of unconstrained single-coordinate Online Gradient Descent (OGD) for linear losses, through an indirect method called `AdaptiveRevision`. Their analysis is specific to OGD, relies crucially on the absence of projection, and assumes that the delays do not change the order the feedback is received (i.e., f_s is received before f_t for all $s < t$; the so-called `InOrder` assumption). The authors pose the question whether there exits a general analysis for algorithms of this type that is less indirect, avoids the `InOrder` assumption, and allows to analyze algorithms with projection and Dual Averaging methods. With the exception of Dual Averaging, the current paper solves this open problem, and obtains simpler algorithms even in the special case considered by McMahan and Streeter [66].

Some problems posed by Mania *et al.* [65]: The recent paper by Mania *et al.* [65] casts the effect of delays as noise on the gradients, and uses this "perturbed iterate" framework to analyze the convergence rate of the sequence of iterates generated by asynchronous unconstrained Stochastic Gradient Descent (SGD). Their analysis relies on strong convexity of the objective function. If the objective function is also smooth and the gradients are sparse and/or the delays satisfy specific bounds, they show that the effect of delays on the rate of convergence of these iterates is asymptotically negligible. The authors pose the question whether it is possible to obtain tight bounds for the function values (rather than the iterates), and whether their framework can be generalized beyond the strong convexity assumption to other learning settings, or to the analysis of AdaGrad-style algorithms. The current paper answers these questions: our framework applies to online convex optimization with linear losses or implicit updates and to function values rather than iterates, and our main reduction result provides an *identity*, not just an upper bound, for the delayed-feedback regret. Furthermore our framework applies to algorithms with projection, does not rely on strong convexity or smoothness of the loss functions,[3] and, as mentioned above, allows us to analyze delay-adaptive AdaGrad-style algorithms.

To our knowledge, Mesterharm [71] was the first to observe, in a special, restricted adversarial classification setting, that the additional regret due to delays depends on how frequently an algorithm changes its predictions. The reduction we present can be considered as a refined and generalized version of his reduction [71, Chapter 8, Algorithm ODB-2]. An advantage of this type of reduction to those in previous works [44], [108] is its resource-efficiency: we use only one instance of a non-delayed online learning algorithm, while previous work created multiple instances, potentially wasting storage and computational resources.[4]

Several recent papers have studied delay-tolerant stochastic optimization [1], [61], [62], [76], [92]; see also the references in the paper of Mania *et al.* [65]. These works typically show that for a specific non-delayed algorithm, for separable objective functions and under data sparsity (and usually assuming smoothness and strong convexity of the loss

[3]Note, however, that our reduction is for full-information online learning, and to be applied with gradient-only information, we have to first linearize the losses. Hence, without full information, our regret bounds apply to the linearized loss, which might not give a regret bound as tight as the original smooth or strongly convex loss.

[4] Joulani *et al.* [44] also provide another reduction using only a single instance under stochastic feedback.

function), the effect of delays on the excess risk is asymptotically negligible, i.e., the rate of convergence is nearly the same as for the corresponding non-delayed algorithms, and hence linear speed ups are possible in parallel processing. We are instead interested in a more basic, unified analysis of the full-information online learning setting to uncover the exact regret penalty due to delays. In addition, assumptions such as data sparsity can be applied to our generic regret bounds to recover some of these results, without the need for smoothness or strong convexity.[5]

An interesting work is the recent paper of Sra *et al.* [101], who consider adapting a 2-norm-regularized Mirror-Descent algorithm to the observed delays in the stochastic optimization setting with specific delay distributions. Compared to their work, we consider a more general version of Mirror-Descent and support FTRL algorithms as well, in the more general online learning setting. However, we would like to emphasize that currently our framework does not contain their work as a special case, since their algorithm does not maintain a non-decreasing regularizer.

Finally, the effect of delayed feedback has also been analyzed beyond the full-information model we consider here. For this, we refer the readers to Joulani *et al.* [44] and the references therein.

3.4 Single-instance black-box reduction

Consider any deterministic non-delayed online learning algorithm (call it Base). Suppose that we use Base, without modification, in a delayed-feedback environment: we feed Base with only the feedback that has arrived, and at each time step, we use the prediction that Base has made after receiving the most recent feedback. This scheme, which we call Solid (for "Single-instance Online Learning In Delayed environments"), is shown in Algorithm 2. In this section, we analyze the regret of Solid in the delayed-feedback setting.

Solid reduces delayed-feedback online learning back to the standard (non-delayed) online learning problem. As we show below, we can express the regret of Solid under delayed feedback in terms of the regret of Base in a non-delayed setting and what we call the *prediction drift* of Base. We start with the definition of the latter.

[5]See, e.g., the comparison with Async-AdaGrad [27] made by McMahan and Streeter [66].

Algorithm 2: Single-instance Online Learning In Delayed environments (SOLID)

```
1 Set $x \leftarrow$ first prediction of BASE.
2 foreach time step $t = 1, 2, \ldots$ do
3     Set $x_t \leftarrow x$ as the prediction for the current time step.
4     Receive the set of feedbacks $H_t$ that arrive at the end of time step $t$.
5     foreach $f_s \in H_t$ do
6         Update BASE with $f_s$.
7         $x \leftarrow$ the next prediction of BASE.
8     end foreach
9 end foreach
```

Definition 3.1 (Prediction drift). *Consider a non-delayed algorithm* BASE *that is run with a sequence of loss functions $f_1, f_2, \ldots, f_n$ in a non-delayed setting, and let $x_s, s = 1, 2, \ldots, n$, denote the s-th prediction of* BASE. *For every $s = 1, 2, \ldots, n$ and $\tau = 1, 2, \ldots, s-1$, the* prediction drift *of* BASE *on f_s from the previous τ time steps is defined as*

$$D_{s,\tau} = f_s(x_{s-\tau}) - f_s(x_s)\,,$$

the difference of the loss f_s of predictions $x_{s-\tau}$ and x_s.

Next, we introduce some further notation that is needed for our regret bound. For $1 \le s \le n$, let $\rho(s)$ denote the time step whose feedback $f_{\rho(s)}$ is the s-th feedback that SOLID gives to BASE, and let $\tilde{f}_s = f_{\rho(s)}$. Let $\tilde{x}_1$ be the first prediction of BASE and $\tilde{x}_{s+1}, s = 1, 2, \ldots, n$, denote the prediction that BASE makes after receiving the s-th feedback $\tilde{f}_s$. Note that not all predictions of BASE become predictions of SOLID. Also note that BASE makes predictions against the losses $\tilde{f}_1, \tilde{f}_2, \ldots, \tilde{f}_n$ sequentially without delays, that is, $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n$ are the predictions of BASE in a non-delayed environment. For $t = 1, 2, \ldots, n$, let $S(t) = \sum_{i=1}^{t-1} \mathbb{I}\{i + \tau_i < t\}$ denote the number of feedbacks that SOLID has observed (and has given to BASE) before making its t-th prediction x_t. Let $\tilde{\tau}_s = s - 1 - S(\rho(s))$ be the number of feedbacks that SOLID gives to BASE while the s-th feedback $\tilde{f}_s$ is outstanding, *i.e.*, the number of feedbacks that BASE receives between the time $\rho(s)$ when SOLID makes the prediction $x_{\rho(s)}$ and the time when the loss function $\tilde{f}_s = f_{\rho(s)}$ is given to BASE. For the analysis below, without loss of generality, we will assume that for any $1 \le t \le n$, $t + \tau_t \le n$, *i.e.*, all feedbacks are received by the end of round n. This does not restrict generality because the feedbacks that arrive in round n are not used to make any predictions and hence cannot influence the regret of SOLID. Note that under this assumption $\sum_{s=1}^n \tilde{\tau}_s = \sum_{t=1}^n \tau_t$ (both count over time the total number of outstanding feedbacks), and $(\rho(s))_{1\le s\le n}$ is a permutation of the integers $\{1,\dots,n\}$.

Theorem 3.1. *Let* BASE *be any deterministic non-delayed forecaster. For every $x \in \mathcal{X}$, the regret of* SOLID *using* BASE *satisfies*

$$R_n(x) = \tilde{R}_n^{\text{BASE}}(x) + \sum_{s=1}^n \tilde{D}_{s,\tilde{\tau}_s}\,, \tag{3.1}$$

where $\tilde{R}_n^{\text{BASE}}(x) = \sum_{s=1}^n \tilde{f}_s(\tilde{x}_s) - \sum_{s=1}^n \tilde{f}_s(x)$ is the (non-delayed) regret of BASE *relative to any $x \in \mathcal{X}$ for the sequence of losses $\tilde{f}_1, \tilde{f}_2, \dots, \tilde{f}_n$, and $\tilde{D}_{s,\tilde{\tau}_s} = \tilde{f}_s(\tilde{x}_{s-\tilde{\tau}_s}) - \tilde{f}_s(\tilde{x}_s)$ is the prediction drift of* BASE *while feedback $\tilde{f}_s$ is outstanding.*

Proof. By construction, for all time steps $t = 1,2,\dots,n$, the prediction x_t of SOLID is the latest prediction of BASE, so we have $x_t = \tilde{x}_{S(t)+1}$, or, equivalently, $x_{\rho(s)} = \tilde{x}_{s-\tilde{\tau}_s}$ for $s = 1,2,\dots,n$. Furthermore, by definition, $\tilde{f}_s = f_{\rho(s)}$, so we have $f_{\rho(s)}(x_{\rho(s)}) - \tilde{f}_s(\tilde{x}_s) = \tilde{f}_s(\tilde{x}_{s-\tilde{\tau}_s}) - \tilde{f}_s(\tilde{x}_s) = \tilde{D}_{s,\tilde{\tau}_s}$. Hence,

$$\begin{aligned} R_n(x) &= \sum_{t=1}^n f_t(x_t) - \sum_{t=1}^n f_t(x) \\ &= \sum_{s=1}^n f_{\rho(s)}(x_{\rho(s)}) - \sum_{s=1}^n f_{\rho(s)}(x) \\ &= \sum_{s=1}^n \tilde{D}_{s,\tilde{\tau}_s} + \sum_{s=1}^n \tilde{f}_s(\tilde{x}_s) - \sum_{s=1}^n \tilde{f}_s(x) \\ &= \sum_{s=1}^n \tilde{D}_{s,\tilde{\tau}_s} + \tilde{R}_n^{\text{BASE}}(x). \end{aligned}$$ □

Note that this result is an *identity*: upper and lower bounds on the (worst-case) regret and prediction drift of any algorithm BASE can be used to obtain upper and lower bounds on the delayed-feedback regret of SOLID.

Theorem 3.1 shows, in particular, that *stable* algorithms, *i.e.*, algorithms with small prediction drifts, are likely to suffer a small additional regret in delayed environments. While in general changing the predictions too slowly might reduce adaptivity and result in a larger regret, Theorem 3.1 shows that in delayed environments the extra regret might be worth the trade-off against the extra penalty from the prediction drift. This formalizes the intuition that in delayed environments one should reduce the learning rate of the algorithms, and helps us characterize the amount by which the learning rate should be decreased.

3.5 Stability of OCO algorithms

In this section, we prove upper-bounds on the prediction drift of a family of online convex optimization algorithms including Follow-The-Regularized-Leader (FTRL), Mirror Descent, and their adaptive variants such as the AdaGrad-style algorithms of McMahan and Streeter [70] and Duchi *et al.* [26], both with and without projection. In particular, we study FTRL-Prox and Ada-MD,[6] which are defined as follows. Both of these algorithms use a sequence of "regularizer functions" $r_0, \dots, r_n : S \to \mathbb{R}$, chosen by the algorithm sequentially (possibly based on previous observations). We assume that $S \subset \mathbb{X}$ is convex and $\mathcal{X} \subset S^\circ$.[7] The first prediction of both algorithms is

$$x_1 = \operatorname*{arg\,min}_{x \in \mathcal{X}} \; r_0(x). \tag{3.2}$$

Then, for $s > 1$, FTRL-Prox predicts

$$x_s = \operatorname*{arg\,min}_{x \in \mathcal{X}} \; f_{1:s-1}(x) + r_{0:s-1}(x), \tag{3.3}$$

while the predictions of Ada-MD are given by

$$x_s = \operatorname*{arg\,min}_{x \in \mathcal{X}} \; f_{s-1}(x) + \mathcal{B}_{r_{0:s-1}}(x, x_{s-1}), \tag{3.4}$$

where $\mathcal{B}_{r_{0:s-1}}(.,.)$ is the Bregman-divergence induced by $r_{0:s-1}$. For FTRL-Prox, we assume that the regularizers r_s are selected such that x_s minimizes r_s on $\mathcal{X}$.

Note that these algorithms have been studied previously in the literature, e.g., by McMahan and Streeter [70], McMahan [67], and Duchi *et al.* [26]. To put our analysis into context, first we state the existing non-delayed regret bounds for these algorithms. In what follows, $f'_s(x_s)$ denotes any sub-gradient of f_s at x_s, and $x_0 := x_1$.

Assumption 3.1. *The loss functions $f_s, s = 1, 2, \dots, n$, are convex, and for all $s = 0, 1, 2, \dots, n$, the regularizer r_s is convex and non-negative. Furthermore, for* FTRL-Prox *we assume that x_s minimizes r_s on $\mathcal{X}$.*

For FTRL-Prox, we have the following regret bound in the non-delayed setting (Theorem 1 of McMahan [67]).

[6]The nomenclature here is somewhat inconsistent. Here we use Ada-MD to mean adaptive mirror descent with implicit update [54] which contains the normal, linear mirror descent as a special case.

[7]By making more assumptions on r_i, *i.e.*, assuming that they are Legendre functions (see, e.g., Cesa-Bianchi and Lugosi [16]), this assumption on the domain of r_i could be relaxed.

Theorem 3.2 (Regret of FTRL-Prox)**.** *Suppose that Assumption 3.1 holds and that* $f_{1:s} + r_{0:s}$ *is* 1*-strongly convex on* $\mathcal{X}$ *w.r.t. some norm* $\|.\|_{(s)}$ *for all* $s = 0, \ldots, n$*. Then the regret of* FTRL-Prox *is upper-bounded as*

$$R_n^{\text{FTRL-Prox}}(x^*) \leq r_{0:n}(x^*) + \frac{1}{2}\sum_{s=1}^{n} \|f'_s(x_s)\|^2_{(s),*}, \tag{3.5}$$

where $\|.\|_{(s),*}$ *is the dual norm of* $\|.\|_{(s)}$*.*

We also have the next regret bound for Ada-MD (following Proposition 3 of Duchi *et al.* [26]):

Theorem 3.3 (Regret of Ada-MD)**.** *Suppose that Assumption 3.1 holds, and for all* $s = 0, \ldots, n$*,* $r_{0:s}$ *is differentiable and* 1*-strongly convex on* $\mathcal{X}$ *w.r.t. some norm* $\|.\|_{(s)}$*. Then, the regret of* Ada-MD *is upper-bounded as*

$$R_n^{\text{Ada-MD}}(x^*) \leq \sum_{s=0}^{n} \mathcal{B}_{r_s}(x^*, x_s) + \frac{1}{2}\sum_{s=1}^{n} \|f'_s(x_s)\|^2_{(s),*}\,. \tag{3.6}$$

The next propositions bound the prediction drifts of FTRL-Prox and Ada-MD. The proofs are short and use standard FTRL and Mirror-Descent techniques. Note that since f_s is convex for $s = 1, 2, \ldots, n$, for any sequence of norms $\|.\|_{(j)}, j = 1, 2, \ldots, n$, and any $1 \leq \tau < s \leq n$,

$$\begin{aligned} D_{s,\tau} &= \sum_{j=s-\tau}^{s-1} f_s(x_j) - f_s(x_{j+1}) \\ &\leq \sum_{j=s-\tau}^{s-1} \langle f'_s(x_j), x_j - x_{j+1}\rangle \\ &\leq \sum_{j=s-\tau}^{s-1} \|f'_s(x_j)\|_{(j),*}\ \|x_j - x_{j+1}\|_{(j)}\,, \end{aligned} \tag{3.7}$$

where the last step follows by Hölder's inequality. We will use this inequality in our proofs below.

Proposition 3.1 (Prediction drift of FTRL-Prox)**.** *Under the conditions of Theorem 3.2, for every* $1 \leq \tau < s \leq n$*,*

$$D_{s,\tau} \leq \sum_{j=s-\tau}^{s-1} \|f'_s(x_j)\|_{(j),*}\|f'_j(x_j)\|_{(j),*}\,. \tag{3.8}$$

Proof. Starting from (3.7), it remains to bound $\|x_j - x_{j+1}\|_{(j)}$. Define $h_0 = r_0$ and $h_s = f_s + r_s$ for $s = 1, 2, \ldots, n$. Then, by our assumptions, x_s minimizes $h_{0:s-1}$ over $\mathcal{X}$, and $h_{0:j}$ is 1-strongly convex w.r.t. $\|.\|_{(j)}$. Note that since x_j minimizes r_j over $\mathcal{X}$, it also minimizes $\phi_1 = h_{0:j-1} + r_j$. Then, since x_{j+1} minimizes $h_{0:j} = h_{0:j-1} + r_j + f_j$, Lemma 3.2 (in Appendix 3.A) with ϕ_1 above and $\delta = f_j$ gives

$$\|x_j - x_{j+1}\|_{(j)} \le \|f'_j(x_j)\|_{(j),*}. \qquad \square$$

Proposition 3.2 (Prediction drift of Ada-MD)**.** *Under the conditions of Theorem 3.3, for every $1 \le \tau < s \le n$,*

$$D_{s,\tau} \le \sum_{j=s-\tau}^{s-1} \|f'_s(x_j)\|_{(j),*} \|f'_j(x_{j+1})\|_{(j),*} \,. \tag{3.9}$$

Proof. As above, we start from (3.7) and bound $\|x_j - x_{j+1}\|_{(j)}$. Recall that $r_{0:j}$ is differentiable by assumption. By the strong convexity of $r_{0:j}$,

$$\begin{aligned}\|x_j - x_{j+1}\|_{(j)}^2 &\le \mathcal{B}_{r_{0:j}}(x_{j+1}, x_j) + \mathcal{B}_{r_{0:j}}(x_j, x_{j+1}) \\ &= \langle r'_{0:j}(x_{j+1}) - r'_{0:j}(x_j), x_{j+1} - x_j \rangle.\end{aligned}$$

Also, by the first-order optimality condition on x_{j+1},

$$\langle f'_j(x_{j+1}) + r'_{0:j}(x_{j+1}) - r'_{0:j}(x_j), x_j - x_{j+1} \rangle \ge 0.$$

Combining the above,

$$\begin{aligned}\|x_j - x_{j+1}\|_{(j)}^2 &\le \langle f'_j(x_{j+1}), x_j - x_{j+1} \rangle \\ &\le \|f'_j(x_{j+1})\|_{(j),*} \|x_j - x_{j+1}\|_{(j)}.\end{aligned}$$

The proposition follows by the non-negativity of norms. $\square$

Note that the proofs use only the standard FTRL-Prox and Ada-MD analysis techniques. Combining the above bounds with Theorem 3.1, it is straightforward to obtain regret guarantees for FTRL-Prox and Ada-MD in delayed-feedback environments. Consider an algorithm Base which is used inside Solid in a delayed-feedback game. Recall that Base receives the sequence of loss functions $\tilde{f}_1, \tilde{f}_2, \ldots, \tilde{f}_n$ and makes predictions $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n$. Also recall that $\tilde{\tau}_n$ denoted the update delay, *i.e.*, the number of updates that Base performs from the time Solid selects x_t in time step t until the time when Base receives the corresponding loss function f_t.

Theorem 3.4. *Suppose Assumption 3.1 holds, and we run* SOLID *in a delayed-feedback environment. Let $\tilde{r}_s, s = 0, 1, \dots, n$, denote the regularizers that* BASE *uses in its simulated non-delayed run inside* SOLID, *and let $\|.\|_{(s)}$ denote the associated strong-convexity norms. Let R_n denote the regret of* SOLID *in its delayed-feedback environment.*

(i) If BASE *is an* FTRL-PROX *algorithm and the conditions of Theorem 3.2 hold, then*

$$R_n \le \tilde{r}_{0:n}(x^*) + \frac{1}{2}\sum_{s=1}^{n} \|\tilde{f}'_s(\tilde{x}_s)\|^2_{(s),*} + \\ \sum_{s=1}^{n}\sum_{j=s-\tilde{\tau}_s}^{s-1} \|\tilde{f}'_s(\tilde{x}_j)\|_{(j),*}\|\tilde{f}'_j(\tilde{x}_j)\|_{(j),*}\,.$$

(ii) If BASE *is an* ADA-MD *algorithm and the conditions in Theorem 3.3 hold, then*

$$R_n \le \sum_{s=0}^{n} \mathcal{B}_{\tilde{r}_s}(x^*, \tilde{x}_s) + \frac{1}{2}\sum_{s=1}^{n} \|\tilde{f}'_s(\tilde{x}_s)\|^2_{(s),*} \\ \sum_{s=1}^{n}\sum_{j=s-\tilde{\tau}_s}^{s-1} \|\tilde{f}'_s(\tilde{x}_j)\|_{(j),*}\|\tilde{f}'_j(\tilde{x}_{j+1})\|_{(j),*}.$$

These bounds are still somewhat unwieldy. To get a more indicative result, suppose that there exists a norm $\|.\|$ such that for all $s = 1, 2, \dots, n$, we have $\|.\|_{(s)} = \frac{1}{\sqrt{\eta_s}}\|.\|$ for some non-negative constant η_s (e.g., consider a single-coordinate ADAGRAD algorithm). Note that by the non-negativity of Bregman divergences, this condition implies that the sequence (η_s) is non-increasing. Further suppose that there exists a constant G such that $\|f'_s(x)\| \le G$ for all $s = 1, 2, \dots, n$ and $x \in \mathcal{X}$, and a constant R such that $\eta_n \tilde{r}_{0:n}(x^*) \le 2R^2$ for FTRL-PROX or $\tilde{\eta}_n \sum_{s=0}^{n} \mathcal{B}_{\tilde{r}_s}(x^*, x_s) \le 2\mathbb{R}^2$ for ADA-MD. Let $\tau^* = \max_{1\le s\le n} \tilde{\tau}_s$ be the maximum delay. Theorems 3.2 and 3.3 give

$$R_n \le \frac{2R^2}{\eta_n} + \frac{G^2}{2}\sum_{s=1}^{n}\eta_s,$$

for FTRL-PROX and ADA-MD in the non-delayed setting, whereas Theorem 3.4 gives

$$R_n \le \frac{2R^2}{\tilde{\eta}_n} + \frac{G^2}{2}\sum_{s=1}^{n}\tilde{\eta}_s(1 + 2\tilde{\tau}_s),$$

for SOLID in the delayed-feedback setting, where $\tilde{\eta}_s, s = 1, 2, \dots, n$, denote the learning rates used by BASE inside SOLID. Using a constant learning rate η_s set as a function of the cumulative delay $\mathcal{T} = \sum_{s=1}^{n} \tilde{\tau}_s = \sum_{s=1}^{n} \tau_s$ (if available in advance), the regret becomes $O(\sqrt{T + 2\mathcal{T}})$, while scaling down the non-delayed learning rates η_s by $\sqrt{1 + 2\tau^*}$, we get a multiplicative regret penalty of $\sqrt{1 + 2\tau^*}$ compared to the non-delayed case.

While the above regret penalty matches (up to a constant factor) the worst-case lower bound for delayed environments [71], [108], McMahan and Streeter [66] show that one can tune the learning rates to adapt to the actual observed delays and past gradients. In the next section, we considerably generalize their results and solve the open problems they have posed.

3.6 Adaptive learning-rate tuning for linear loss functions

In this section we restrict our attention to linear loss functions $f_t = \langle g_t, \cdot \rangle$. Consider the setting of Theorem 3.4, and let $\tilde{g}_s$ denote the gradient of $\tilde{f}_s$. Consider a norm $\|.\|$, and for $s = 1, 2, \ldots, n$, define $\hat{g}_s = \|\tilde{g}_s\|_*$. The following result is a corollary of Theorem 3.4 that generalizes the bound in Lemma 1 of McMahan and Streeter [66] to FTRL-Prox and Ada-MD, either with or without projection. The proof is given in Appendix 3.B.

Corollary 3.1. *Consider the case of linear losses and suppose that the conditions of Theorem 3.4 hold. Suppose that $\tilde{r}_{0:s}, s = 0, 1, \ldots, n$, is $(1/\tilde{\eta}_s)$-strongly convex w.r.t. the norm $\|.\|$. Then, the regret of* Solid *satisfies*

$$R_n \leq \frac{2R^2}{\tilde{\eta}_n} + \frac{1}{2} \sum_{j=1}^{n} \tilde{\eta}_j \hat{G}_j^{\text{fwd}}, \tag{3.10}$$

where for $j = 1, 2, \ldots, n$,

$$\hat{G}_j^{\text{fwd}} = \hat{g}_j^2 + 2\hat{g}_j \sum_{s=j+1}^{n} \hat{g}_s \mathbb{I}\{s - \tilde{\tau}_s \leq j\},$$

and $R > 0$ is a constant upper-bounding the regularizer terms, i.e., $\tilde{\eta}_n \tilde{r}_{0:n}(x^*) \leq 2R^2$ for FTRL-Prox*, or $\tilde{\eta}_n \sum_{s=0}^{n} \mathcal{B}_{\tilde{r}_s}(x^*, \tilde{x}_s) \leq 2R^2$ for* Ada-MD.

Note that in a non-delayed setting, the regret bound of these adaptive algorithms is of the form

$$R_n \leq \frac{2R^2}{\eta_n} + \frac{1}{2} \sum_{s=1}^{n} \eta_s \|g_s\|_*^2.$$

In such a case we would let $\eta_s = O(1/\sqrt{\sum_{j=1}^{s} \hat{g}_j^2})$ and then use Lemma 3.3 (see Appendix 3.A) to get a regret bound of the form

$$R_n \leq 2\sqrt{2}R^2 \sqrt{\sum_{s=1}^{n} \|g_s\|_*^2}\,.$$

Similarly, with $\tilde{\eta}_s = O(1/\sqrt{\hat{G}_{1:s}^{\text{fwd}}})$, in a delayed-feedback setting we would achieve a regret of the form

$$R_n \leq 2\sqrt{2}R^2 \sqrt{\hat{G}_{1:n}^{\text{fwd}}}\,. \tag{3.11}$$

Unfortunately, this is not possible since $\hat{G}_s^{\text{fwd}}$ depends on future, unobserved gradients, and hence the $\tilde{\eta}_s$ given above cannot be computed at time step s.

To work around this problem, McMahan and Streeter [66] define a quantity $\hat{G}_s^{\text{bck}}$ that depends only on the observed gradients. The goal is to bound $\hat{G}_{1:s}^{\text{fwd}}$ from above and below by a function of $\hat{G}_j^{\text{bck}}, j = 1, 2, \ldots, s-1$, plus an additive term independent of s; then, setting the learning rate based on that quantity results in a regret bound that is only an additive term (independent of n) larger than the bound of (3.11). Similarly, in our setting we define

$$\hat{G}_s^{\text{bck}} = \hat{g}_s^2 + 2\hat{g}_s \sum_{j=s-\tilde{\tau}_s}^{s-1} \hat{g}_j.$$

The next lemma bounds $\hat{G}_{1:s}^{\text{fwd}}$ from above and below using $\hat{G}_{1:s}^{\text{bck}}$.

Lemma 3.1. *Let $G_* = \max_{1\leq j\leq n} \hat{g}_j$ and $\tau_* = \max_{1\leq s\leq n} \tilde{\tau}_s$. For all $t = 1, 2, \ldots, n$,*

$$\hat{G}_{1:t}^{\text{bck}} \leq \hat{G}_{1:t}^{\text{fwd}} \leq \hat{G}_{1:t}^{\text{bck}} + (\tau_*^2 + \tau_*)G_*^2. \tag{3.12}$$

In addition,

$$\hat{G}_{1:n}^{\text{fwd}} = \hat{G}_{1:n}^{\text{bck}}. \tag{3.13}$$

Instead of using $\hat{G}_{1:s}^{\text{bck}}$ directly as in our Lemma 3.1, McMahan and Streeter [66] use $\hat{G}_{1:o(s)}^{\text{bck}}$ for their bounds, where $o(s)$ is the index of the largest outstanding gradient at the time of update s. Their bounds need an extra `In-Order` assumption on the delays, i.e., that the delays do not change the order of the updates. In addition, $\hat{G}_{1:o(s)}^{\text{bck}}$ is not efficiently computable in an online fashion (it requires keeping track of the outstanding updates), and they use an indirect algorithm (called `AdaptiveRevision`) on top of this learning rate schedule that "revises" the previous gradients and can be implemented in practice. We do not require this indirect approach, since $\hat{G}_s^{\mathrm{bck}}$ can be efficiently computed in an online fashion (in fact, this is the quantity z that the `AdaptiveRevision` algorithm of McMahan and Streeter [66] also needs and maintains, using the network as storage).

Based on Lemma 3.1, we can show that setting the learning rate, for some $\alpha > 0$, as

$$\tilde{\eta}_j = \alpha \left(\sqrt{\hat{G}_{1:j}^{\mathrm{bck}} + (\tau_*^2 + \tau_*) G_*^2} \right)^{-1}, \tag{3.14}$$

results in only a constant additional regret compared to using the learning rate $\tilde{\eta}_j = O\left(1/\sqrt{\hat{G}_{1:j}^{\mathrm{fwd}}}\right)$. We prove this in the following theorem.

Theorem 3.5. *Consider the conditions of Corollary 3.1. If $\alpha = \sqrt{2}R$ and $\tilde{\eta}_t$ is given by* (3.14)*, then the regret of* Solid *with* FTRL-Prox *or* Ada-MD *can be bounded as*

$$R_n(x^*) \le 2\sqrt{2}R\sqrt{\hat{G}_{1:n}^{\mathrm{fwd}}} + \sqrt{2(\tau_*^2 + \tau_*)}RG_* \,.$$

This generalizes the bound

$$R_n(x^*) \le 2\sqrt{2}R\sqrt{\max_{1\le s\le n} \hat{G}_{1:s}^{\mathrm{fwd}}} + O(\tau^* R G_*) \tag{3.15}$$

obtained in Theorem 3 of McMahan and Streeter [66] for `AdaptiveRevision`. Note, however, that in the case of `AdaptiveRevision`, the algorithm is applied to a single coordinate, and the $\hat{g}$ values are the actual (possibly negative) gradients, not their norms. To refine our bound to the one-dimensional setting, one can define the step-size $\tilde{\eta}_j$ based on the maximum $\hat{G}_{1:i}^{\mathrm{bck}}$ for $1 \le i \le j$ (this is still efficiently computable at time j, and corresponds to the quantity z' in `AdaptiveRevision`). Then, a modified Lemma 3.1 together with Corollary 10 of McMahan and Streeter [66] gives a regret bound similar to (3.15).

3.7 Conclusion and future work

We provided a unified framework for developing and analyzing online convex optimization algorithms under delayed feedback. Based on a general reduction, we extended two generic adaptive online learning algorithms (an adaptive FTRL and an adaptive Mirror-Descent algorithm) to the delayed feedback setting. Our analysis resulted in generalized delay-tolerant AdaGrad-style algorithms that adapt both to the gradients and the delays, solving a number of open problems posed by McMahan and Streeter [66] and Mania *et al.* [65].

An interesting problem for future research is analyzing delay-tolerant adaptive Dual Averaging algorithms using this framework. Deriving lower bounds for asynchronous optimization using Theorem 3.1 is also of natural interest. Finally, it seems to be possible to extend our framework to use gradients or higher-order information only, using a shifting argument; this is also left for future work.

Appendices

3.A Technical lemmas

A more general version of the following lemma has appeared before [67, Lemma 8]. Here we provide a simpler version that is sufficient for our needs. The proof only uses basic techniques. Another slight difference to the presentation of McMahan (2014) is that we make the optimization domain explicit.

Lemma 3.2. *Let $\phi_1, \delta : \mathcal{X} \to \mathbb{R}$ be convex functions, $\phi_2 = \phi_1 + \delta$, $x_1 = \arg\min_{x\in\mathcal{X}} \phi_1(x)$ and $x_2 = \arg\min_{x\in\mathcal{X}} \phi_2(x)$. Assume further that ϕ_2 is 1-strongly convex w.r.t. some norm $\|.\|$, and let $\|.\|_*$ denote its associated dual norm. Then, for any $b \in \partial\delta(x_1)$, we have*

$$\|x_1 - x_2\| \le \|b\|_* \,. \tag{3.16}$$

Proof. Define $\phi_0(x) = \phi_1(x) + \delta(x) - \langle b, x\rangle$, and note that since $x_1 \in \mathcal{X}$, δ is convex, and b is its sub-gradient at x_1, x_1 minimizes $\delta(x) - \langle b, x\rangle$ over $\mathcal{X}$. Hence, x_1 also minimizes $\phi_0(x)$ over $\mathcal{X}$. In addition, ϕ_0 is 1-strongly convex w.r.t. the norm $\|.\|$, since by definition $\phi_0(x) = \phi_2(x) - \langle b, x\rangle$. Then, if b_2 denotes any sub-gradient of ϕ_2 at x_2 and b_0 denotes any sub-gradient of ϕ_0 at x_1, the first order optimality conditions on ϕ_2 and ϕ_0 at x_2 and x_1, respectively, imply that

$$\langle b_0 - b_2, x_1 - x_2\rangle \le 0. \tag{3.17}$$

In addition, strong convexity of ϕ_0 and ϕ_2 implies:

$$\phi_2(x_1) - \phi_2(x_2) - \langle b_2, x_1 - x_2\rangle \ge \frac{1}{2}\|x_1 - x_2\|^2,$$

and

$$\phi_0(x_2) - \phi_o(x_1) - \langle b_0, x_2 - x_1\rangle \ge \frac{1}{2}\|x_1 - x_2\|^2.$$

Adding the two sides together and using (3.17),

$$\begin{aligned}\|x_1 - x_2\|^2 &\leq \langle b_0 - b_2, x_1 - x_2\rangle + \\ &\quad \phi_2(x_1) - \phi_o(x_1) + \\ &\quad \phi_0(x_2) - \phi_2(x_2) \\ &\leq \langle b, x_1\rangle - \langle b, x_2\rangle \\ &\leq \|b\|_* \|x_1 - x_2\|,\end{aligned}$$

using Hölder's inequality in the last step. Non-negativity of the norms completes the proof. □

The next lemma is due to McMahan (2014).

Lemma 3.3 (McMahan (2014, Lemma 9)). *For any sequence of real numbers $x_1, x_2, \ldots, x_n$ such that $x_{1:t} > 0$ for all $t = 1, 2, \ldots, n$, we have*

$$\sum_{t=1}^{n} \frac{x_t}{\sqrt{x_{1:t}}} \leq 2\sqrt{x_{1:n}}\,.$$

3.B Missing proofs

Proof of Corollary 3.1. Note that $\tilde{r}_{0:s}$ is 1-strongly convex w.r.t. the norm $\sqrt{(1/\tilde{\eta}_s)}\|.\|$, the dual of which is given by $\sqrt{\tilde{\eta}_s}\|.\|_*$. Hence, from Theorem 3.4,

$$\begin{aligned}R_n &\leq \frac{2R^2}{\tilde{\eta}_n} + \frac{1}{2}\sum_{s=1}^{n} \tilde{\eta}_s \hat{g}_s^2 + \sum_{s=1}^{n}\sum_{j=s-\tilde{\tau}_s}^{s-1} \tilde{\eta}_j \hat{g}_s \hat{g}_j \\ &= \frac{2R^2}{\tilde{\eta}_n} + \frac{1}{2}\sum_{j=1}^{n} \tilde{\eta}_j \hat{g}_j^2 + \sum_{j=1}^{n}\sum_{s=j+1}^{n} \tilde{\eta}_j \hat{g}_s \hat{g}_j \ \mathbb{I}\{s - \tilde{\tau}_s \leq j\} \\ &= \frac{2R^2}{\tilde{\eta}_n} + \frac{1}{2}\sum_{j=1}^{n} \tilde{\eta}_j \left(\hat{g}_j^2 + 2\hat{g}_j \sum_{s=j+1}^{n} \hat{g}_s \mathbb{I}\{s - \tilde{\tau}_s \leq j\}\right),\end{aligned}$$

finishing the proof. □

Proof of Lemma 3.1. From the definition,

$$\hat{G}_{1:t}^{\text{bck}} = \sum_{s=1}^{t} \hat{G}_s^{\text{bck}} = \sum_{s=1}^{t} \hat{g}_s^2 + 2\sum_{s=1}^{t}\sum_{j=s-\tilde{\tau}_s}^{s-1} \hat{g}_s \hat{g}_j$$

$$
\begin{aligned}
&= \sum_{s=1}^{t} \hat{g}_s^2 + 2\sum_{j=1}^{t}\sum_{s=j+1}^{t} \hat{g}_s\hat{g}_j \mathbb{I}\{s-\tilde{\tau}_s \le j\} \\
&= \sum_{j=1}^{t} \hat{g}_j^2 + 2\sum_{j=1}^{t}\hat{g}_j\sum_{s=j+1}^{n} \hat{g}_s \mathbb{I}\{s-\tilde{\tau}_s \le j\} \\
&\quad - 2\sum_{j=1}^{t}\hat{g}_j\sum_{s=t+1}^{n} \hat{g}_s \mathbb{I}\{s-\tilde{\tau}_s \le j\} \\
&= \sum_{j=1}^{t} \hat{G}_j^{\text{fwd}} - 2\sum_{j=1}^{t}\hat{g}_j\sum_{s=t+1}^{n} \hat{g}_s \mathbb{I}\{s-\tilde{\tau}_s \le j\}.
\end{aligned}
$$

Given that the subtracted term is non-negative, we have $\hat{G}_{1:t}^{\text{bck}} \le \hat{G}_{1:t}^{\text{fwd}}$. Also, for $t=n$, the subtracted term is zero, proving the last part of the lemma. Therefore, it remains to bound the subtracted term by $(\tau_*^2+\tau_*)G_*^2$, or, equivalently, to bound $\sum_{j=1}^{t}\sum_{s=t+1}^{n}\mathbb{I}\{s-\tilde{\tau}_s \le j\}$ by $\frac{1}{2}(\tau_*^2+\tau_*)$. To that end, note that for $j \le t-\tau_*$ and $s>t$, the indicator $\mathbb{I}\{s-\tilde{\tau}_s \le j\}$ is always zero. Also, note that $\mathbb{I}\{s-\tilde{\tau}_s \le j\}=0$ for $s > j+\tau_*$. Hence,

$$
\begin{aligned}
\sum_{j=1}^{t}\sum_{s=t+1}^{n}\mathbb{I}\{s-\tilde{\tau}_s \le j\} &= \sum_{j=t-\tau_*+1}^{t}\sum_{s=t+1}^{j+\tau_*}\mathbb{I}\{s-\tilde{\tau}_s \le j\} \\
&\le \sum_{j=t-\tau_*+1}^{t}(j+\tau_*-t) \\
&= \sum_{i=1}^{\tau_*} i = \frac{1}{2}\tau_*(\tau_*+1)\,,
\end{aligned}
$$

concluding the proof. □

Proof of Theorem 3.5. By Corollary 3.1, it suffices to bound the two terms on the r.h.s. of (3.10). Since $\sqrt{a+b} \le \sqrt{a}+\sqrt{b}$ for any nonnegative numbers a and b,

$$
\begin{aligned}
\frac{2R^2}{\tilde{\eta}_n} &= \sqrt{2}R\sqrt{\hat{G}_{1:n}^{\text{bck}} + (\tau_*^2+\tau_*)G_*^2} \\
&\le \sqrt{2}R\sqrt{\hat{G}_{1:n}^{\text{bck}}} + \sqrt{2(\tau_*^2+\tau_*)}RG_* \\
&= \sqrt{2}R\sqrt{\hat{G}_{1:n}^{\text{fwd}}} + \sqrt{2(\tau_*^2+\tau_*)}RG_*,
\end{aligned}
$$

using (3.13) in the last step. Also, from (3.12),

$$
\tilde{\eta}_j = \frac{\alpha}{\sqrt{\hat{G}_{1:j}^{\text{bck}} + (\tau_*^2+\tau_*)G_*^2}} \le \frac{\alpha}{\sqrt{\hat{G}_{1:j}^{\text{fwd}}}}.
$$

Therefore, by Lemma 3.3,

$$\begin{aligned}\frac{1}{2}\sum_{j=1}^{n}\tilde{\eta}_j\hat{G}_j^{\mathrm{fwd}} &\leq \frac{1}{2}\sum_{j=1}^{n}\frac{\alpha}{\sqrt{\hat{G}_{1:j}^{\mathrm{fwd}}}}\hat{G}_j^{\mathrm{fwd}}\\ &\leq \sqrt{2}R\sqrt{\hat{G}_{1:n}^{\mathrm{fwd}}}.\end{aligned}$$

Combining with (3.10) completes the proof. □

Chapter 4

A Unified Modular Analysis of Online Optimization

Abstract

Recently, much work has been done on extending the scope of online learning and incremental stochastic optimization algorithms. In this paper we contribute to this effort in two ways: First, based on a generalization of Bregman divergences and a generic regret decomposition, we provide a self-contained, modular analysis of the two workhorses of online learning: (general) adaptive versions of Mirror Descent (MD) and the Follow-the-Regularized-Leader (FTRL) algorithms. The analysis is done with extra care so as not to introduce assumptions not needed in the proofs and allows to combine, in a straightforward way, different algorithmic ideas (e.g., adaptivity, optimism, implicit updates, variance reduction) and learning settings (e.g., strongly convex or composite objectives). This way we are able to reprove, extend and refine a large body of the literature, while keeping the proofs concise. The second contribution is a by-product of this careful analysis: We present algorithms with improved variational bounds for smooth, composite objectives, including a new family of optimistic MD algorithms with only one projection step per round. Furthermore, we provide a simple extension of adaptive regret bounds to a class of practically relevant non-convex problem settings (namely, star-convex loss functions and their extensions) with essentially no extra effort.[1]

[1] This chapter and the related appendices are accepted for publication in the Theoretical Computer Science (TCS) journal. An earlier version has been published as the following conference paper:

- P. Joulani *et al.*, "A modular analysis of adaptive (non-) convex optimization: Optimism, composite objectives, and variational bounds," in *Proceedings of Machine Learning Research (Algorithmic Learning Theory 2017)*, 2017, pp. 681–720.

4.1 Introduction

Online and stochastic optimization algorithms form the underlying machinery in much of modern machine learning. Perhaps the most well-known example is Stochastic Gradient Descent (SGD) and its adaptive variants, the so-called AdaGrad algorithms [26], [70]. Other special cases include multi-armed and linear bandit algorithms, as well as algorithms for online control, tracking and prediction with expert advice [16], [38], [97].

There are numerous algorithmic variants in online and stochastic optimization, such as adaptive [26], [70] and optimistic algorithms [18], [50], [72], [88], [90], implicit updates [52], [54], composite objectives [25], [26], [109], or non-monotone regularization [101]. Each of these variants has been analyzed under a specific set of assumptions on the problem, e.g., smooth [24], [48], [55], convex [38], [69], [82], [97], or strongly convex [40], [69], [82], [98] objectives. However, a useful property is typically missing from the analyses: modularity. It is typically not clear from the original analysis whether the algorithmic idea can be mixed with other techniques, or whether the effect of the assumptions extend beyond the specific setting considered. For example, based on the existing analyses it is very much unclear to what extent AdaGrad techniques, or the effects of smoothness, or variational bounds in online learning, extend to new learning settings. Thus, for every new combination of algorithmic ideas, or under every new learning setting, the algorithms are typically analyzed from scratch.

A special new learning setting is non-convex optimization. While the bulk of results in online and stochastic optimization assume the convexity of the loss functions, online and stochastic optimization algorithms have been successfully applied in settings where the objectives are non-convex. In particular, highly popular deep learning techniques [35] are based on the application of stochastic optimization algorithms to non-convex objectives. In the face of this discrepancy between the state of the art in theory and practice, an on-going thread of research attempts to generalize the analyses of stochastic optimization to non-convex settings. In particular, certain non-convex problems have been shown to actually admit efficient optimization methods, usually taking some form of a gradient method (one such problem is matrix completion, see, e.g., [9], [31]).

The goal of this paper is to provide a flexible, modular analysis of online and stochastic optimization algorithms that allows one to easily combine different algorithmic techniques and learning settings under as few assumptions as possible. We demonstrate the benefits of such an analysis by combining and transferring algorithmic ideas to create new algorithms, and by exploring the extent to which the convexity assumption can be relaxed in our analysis.

4.1.1 Contributions

First, building on previous attempts to unify the analyses of online and stochastic optimization [38], [69], [82], [97], we provide a unified analysis of a large family of optimization algorithms in general Hilbert spaces. The analysis is crafted to exhibit modularity by decoupling the contribution of each assumption or algorithmic idea from the analysis, so as to enable us to combine different assumptions and techniques without analyzing the algorithms from scratch. In particular, we rely on a generalized Bregman divergence definition, followed by a careful decomposition of the optimization performance (optimization error or regret) into two parts: the first part captures the generic performance of the algorithm, whereas the second part connects the assumptions about the learning setting to the information given to the algorithm. Lemma 4.1 in Section 4.2.1 provides such a decomposition.[2] Then, in Theorem 4.1, we bound the generic (first) part, using a careful analysis of the linear regret of generalized adaptive Follow-The-Regularized-Leader (FTRL) and Mirror Descent (MD) algorithms.

Second, we use this analysis framework to provide a concise summary of a large body of previous results. Section 4.4 provides the basic results, and Sections 4.5 to 4.7 present the relevant extensions and applications. A combination of our techniques with variance reduction techniques is explored in Section 4.8.

Third, building on the aforementioned modularity, we analyze new learning algorithms. In particular, in Section 4.7.4 we analyze a new adaptive, optimistic, composite-objective FTRL algorithm with variational bounds for smooth convex loss functions, which combines the best properties and avoids the limitations of the previous work. We also present a new class of optimistic MD algorithms with only one MD update per round (Section 4.7.2).

[2] Our approach can be viewed as a refined version of the so-called "follow the leader/be the leader"-style analysis. Previous work (e.g., [69], [97]) may give the impression that "follow-the-leader/be-the-leader" analyses lose constant factors while other methods such as primal-dual analysis do not. This is not the case about our analysis. In fact, we improve constants in optimistic online learning; see Section 4.7 for details.

Finally, we extend the previous results to special classes of non-convex optimization problems in Section 4.9; as a result of relaxing the assumptions in our framework as much as possible, these results come without any extra effort just by examining to what non-convex problems our results generalize immediately. The family of non-convex functions we arrive at generalizes a small, but practically important class of functions (the set of star-convex functions) considered in previous work on non-convex optimization [37], [58], [79].

4.1.2 Notation and definitions

We work with a (possibly infinite-dimensional) Hilbert space $\mathcal{H}$ over the reals. That is, $\mathcal{H}$ is a real vector space equipped with an inner product $\langle\cdot,\cdot\rangle : \mathcal{H}\times\mathcal{H}\to\mathbb{R}$, such that $\mathcal{H}$ is complete with respect to (w.r.t.) the norm induced by $\langle\cdot,\cdot\rangle$. Examples include $\mathcal{H}=\mathbb{R}^d$ (for a positive integer d) where $\langle\cdot,\cdot\rangle$ is the standard dot-product, or $\mathcal{H}=\mathbb{R}^{m\times n}$, the set of $m\times n$ real matrices, where $\langle A,B\rangle=\operatorname{tr}\left(A^\top B\right)$, or $\mathcal{H}=\ell^2(\mathcal{C})$, the set of square-integrable real-valued functions on $\mathcal{C}\subset\mathbb{R}^d$, where $\langle f,g\rangle=\int_{\mathcal{C}} f(x)g(x)dx$ for any $f,g\in\mathcal{H}$.

We denote the extended real line by $\overline{\mathbb{R}}:=\mathbb{R}\cup\{-\infty,+\infty\}$, and work with functions of the form $f:\mathcal{H}\to\overline{\mathbb{R}}$. Given a set $C\subset\mathcal{H}$, the *indicatrix* of C is the function $\mathcal{I}_C:\mathcal{H}\to\overline{\mathbb{R}}$ given by $\mathcal{I}_C(x)=0$ for $x\in C$ and $\mathcal{I}_C(x)=+\infty$ for $x\notin C$. The *effective domain* of a function $f:\mathcal{H}\to\overline{\mathbb{R}}$, denoted by $\operatorname{dom}(f)$, is the set $\{x\in\mathcal{H}\mid f(x)<+\infty\}$ where f is less than infinity; conversely, we identify any function $f:C\to\overline{\mathbb{R}}$ defined only on a set $C\subset\mathcal{H}$ by the function $f+\mathcal{I}_C$. A function f is *proper* if $\operatorname{dom}(f)$ is non-empty and $f(x)>-\infty$ for all $x\in\mathcal{H}$.

Let $f:\mathcal{H}\to\overline{\mathbb{R}}$ be proper. We denote the set of all sub-gradients of f at $x\in\mathcal{H}$ by $\partial f(x)$, that is,

$$\partial f(x):=\{\ u\in\mathcal{H}\mid\forall y\in\mathcal{H},\langle u,y-x\rangle+f(x)\le f(y)\ \}\ .$$

The function f is *sub-differentiable* at x if $\partial f(x)\neq\emptyset$; we use $f'(x)$ to denote any member of $\partial f(x)$. Note $\partial f(x)=\emptyset$ when $x\notin\operatorname{dom}(f)$.

Let $x\in\operatorname{dom}(f)$, assume that $f(x)>-\infty$, and let $z\in\mathcal{H}$. The *directional derivative* of f at x in the direction z is defined as $f'(x;z):=\lim_{\alpha\downarrow 0}\frac{f(x+\alpha z)-f(x)}{\alpha}$, provided that the limit exists in $[-\infty,+\infty]$. The function f is *differentiable* at x if it has a *gradient* at x, that is, a vector $\nabla f(x)\in\mathcal{H}$ such that $f'(x;z)=\langle\nabla f(x),z\rangle$ for all $z\in\mathcal{H}$. The function f is *locally sub-differentiable* at x if it has a *local sub-gradient* at x, that is, a vector $g_x\in\mathcal{H}$

such that $\langle g_x, z\rangle \le f'(x; z)$ for all $z \in \mathcal{H}$. We denote the set of local sub-gradients of f at x by $\delta f(x)$. Note that if $f'(x; z)$ exists for all $z \in \mathcal{H}$, and f is sub-differentiable at x, then it is also locally sub-differentiable with $g_x = u$ for any $u \in \partial f(x)$. Similarly, if f is differentiable at x, then it is also locally sub-differentiable, with $g_x = \nabla f(x)$. The function f is called *directionally differentiable at* $x \in \mathrm{dom}(f)$ if $f(x) > -\infty$ and $f'(x; z)$ exists in $[-\infty, +\infty]$ for all $z \in \mathcal{H}$; f is called *directionally differentiable* if it is directionally differentiable at every $x \in \mathrm{dom}(f)$. Note that a directionally differentiable function is proper.

Next, we define a generalized[3] notion of Bregman divergence:

Definition 4.1 (Bregman divergence)**.** *Let f be directionally differentiable at $x \in \mathrm{dom}(f)$. The f-induced* Bregman divergence *from x is the function from $\mathcal{H} \to \overline{\mathbb{R}}$, given by*

$$\mathcal{B}_f(y, x) := \begin{cases} f(y) - f(x) - f'(x; y - x)\,. & \textit{if } f(y) \textit{ is finite;} \\ +\infty & \textit{otherwise}\,. \end{cases} \tag{4.1}$$

A function $f : \mathcal{H} \to \overline{\mathbb{R}}$ is *convex* if for all $x, y \in \mathrm{dom}(f)$ and all $\alpha \in (0, 1)$, $\alpha f(x) + (1-\alpha) f(y) \ge f(\alpha x + (1-\alpha) y)$. We can show that a proper convex function is always directionally differentiable, and the Bregman divergence it induces is always non-negative (see Appendix 4.E). Let $\|.\|$ denote a norm on $\mathcal{H}$ and let $L, \beta > 0$. A directionally differentiable function $f : \mathcal{H} \to \overline{\mathbb{R}}$ is *β-strongly convex* w.r.t. $\|.\|$ iff $\mathcal{B}_f(x, y) \ge \frac{\beta}{2}\|x - y\|^2$ for all $x, y \in \mathrm{dom}(f)$. The function f is *L-smooth* w.r.t. $\|.\|$ iff for all $x, y \in \mathrm{dom}(f)$, $|\mathcal{B}_f(x, y)| \le \frac{L}{2}\|x - y\|^2$.

We use $\{c_t\}_{t=i}^{j}$ to denote the sequence $c_i, c_{i+1}, \ldots, c_j$, and $c_{i:j}$ to denote the sum $\sum_{t=i}^{j} c_t$, with $c_{i:j} := 0$ for $i > j$.

4.2 Problem setting: online optimization

We study a general first-order iterative optimization setting that encompasses several common optimization scenarios, including online, stochastic, and full-gradient optimization. Consider a convex set $\mathcal{X} \subset \mathcal{H}$, a sequence of directionally differentiable functions

[3]If f is differentiable at x, then (4.1) matches the traditional definition of Bregman divergence. Previous work also considered generalized Bregman divergences, e.g., the works of Kiwiel [53] and Telgarsky and Dasgupta [103] and the references therein. However, our definition is not limited to convex functions, allowing us to study convex and non-convex functions and regularizers under a unified theory; see, e.g., Sections 4.3 and 4.9.

Input: convex set $\mathcal{X} \subset \mathcal{H}$; directionally differentiable functions $f_1, f_2, \ldots, f_T$ from $\mathcal{H}$ to $\overline{\mathbb{R}}$.

- The algorithm selects an initial point $x_1 \in \mathcal{X}$.
- **For each** time step $t = 1, 2, \ldots, T$:
  - The algorithm observes feedback $g_t \in \mathcal{H}$ and selects the next point $x_{t+1} \in \mathcal{X}$.

Goal: Minimize the regret $R_T(x^*)$ (given by (4.2)) against any $x^* \in \mathcal{X}$.

Figure 4.1: Iterative optimization.

$f_1, f_2, \ldots, f_T$ from $\mathcal{H}$ to $\overline{\mathbb{R}}$ with $\mathcal{X} \subset \operatorname{dom}(f_t)$ for all $t = 1, 2, \ldots, T$, and a first-order iterative optimization algorithm. The algorithm starts with an initial point x_1. Then, in each iteration $t = 1, 2, \ldots, T$, the algorithm suffers a loss $f_t(x_t)$ from the latest point x_t, receives some feedback $g_t \in \mathcal{H}$, and selects the next point x_{t+1}. Typically, $\langle g_t, \cdot \rangle$ is supposed to be an estimate or lower bound on the directional derivative of f_t at x_t. This protocol is summarized in Figure 4.1.

Unlike online convex optimization (OCO), at this stage we do not assume that the f_t are convex[4] or differentiable, nor do we assume that g_t are gradients or sub-gradients. Our goal is to minimize the *regret* $R_T(x^*)$ against any $x^* \in \mathcal{X}$, defined as

$$R_T(x^*) = \sum_{t=1}^{T} \left(f_t(x_t) - f_t(x^*) \right) . \tag{4.2}$$

More precisely, we want to guarantee that $\sup_{x^* \in \mathcal{X}} R_T(x^*)$ is small.

In this paper we are also concerned with the problem of stochastic optimization, where the goal is to minimize a function f defined by $f(x) := \mathbb{E}_{\xi \sim D} F(x, \xi)$. That is, the optimization algorithm has to find an estimate $\hat{x}_T$ such that $\mathbb{E}\{f(\hat{x}_T)\} - f(x^*)$ is small, given noisy gradient observations g_t such that $\mathbb{E}\{g_t | x_t\} \in \delta f(x_t)$. It is well-known (e.g., [97, Theorem 5.1]) that for any f, this equals $\mathbb{E}\{R_T(x^*)/T\}$ if $\hat{x}_T$ is selected uniformly at random from $x_1, \ldots, x_T$. Also, if f is convex, $\mathbb{E}\{f(\hat{x}_T)\} - f(x^*) \le \mathbb{E}\{R_T(x^*)/T\}$ if $\hat{x}_T$ is the average of $x_1, \ldots, x_T$. Thus, the analysis of online algorithms that provide guarantees on the regret $R_T(x^*)$ can be used to provide performance bounds for iterative stochastic optimization, and we will analyze stochastic optimization algorithms using this

[4]There is a long tradition of non-convex assumptions in the Stochastic Approximation (SA) literature, see, e.g., the book of Bertsekas and Shreve [8]. Our results differ in that they apply to more recent advances in online learning (e.g., AdaGrad algorithms), and we derive any-time regret bounds, rather than asymptotic convergence results, for specific non-convex function classes.

approach. Note that this sometimes comes with a sacrifice. For example, in strongly-convex stochastic optimization, using suffix-averaging (averaging just the last αT iterates with $\alpha < 1$ instead of all T of them) [89], [100] or successive tuning and restarting [39] leads to a slightly improved, optimal performance that is not available through the direct reduction mentioned above [39]. The regret framework is also not directly applicable to explain some refined notions, such as acceleration (see, e.g., [55]). Furthermore, in optimization one often cares about the performance of the last iterate x_T. Even in convex problems, the analysis of the last iterate follows only indirectly from a regret analysis [89], [100]. Some of these problems can be addressed, for example, by considering a weighted regret where the terms in $R_T(x^*)$ have different weights (see, e.g., [105]) or by additional tricks applied on top of the regret analysis (such as done by [39], [89], [100]), which is beyond the scope of our paper. Finally, our analysis concerns global convergence, and hence applies only to convex and a small class of non-convex problems, while there are several other results in non-convex stochastic optimization that we do not recover (mostly about proving convergence rates to local optima, e.g., [3], [110]).

4.2.1 Regret decomposition

Below, we provide a decomposition of $R_T(x^*)$, which holds for any sequence of points $x_1, x_2, \ldots, x_{T+1}$ and any x^*. The decomposition is in terms of the *forward linear regret* $R_T^+(x^*)$, defined as

$$R_T^+(x^*) := \sum_{t=1}^{T} \langle g_t, x_{t+1} - x^* \rangle \,.$$

Intuitively, R_T^+ is the regret (in linear losses) of the "cheating" algorithm that uses action x_{t+1} at time t, and depends only on the choices of the algorithm and the feedback it receives.

Lemma 4.1 (Regret decomposition)**.** *Let $x^*, x_1, x_2, \ldots, x_{T+1}$ be any sequence of points in $\mathcal{X}$. For $t = 1, 2, \ldots, T$, let $f_t : \mathcal{H} \to \overline{\mathbb{R}}$ be directionally differentiable with $\mathcal{X} \subset \operatorname{dom}(f_t)$, and let $g_t \in \mathcal{H}$. Then,*

$$R_T(x^*) = R_T^+(x^*) + \sum_{t=1}^{T} \langle g_t, x_t - x_{t+1} \rangle - \sum_{t=1}^{T} \mathcal{B}_{f_t}(x^*, x_t) + \sum_{t=1}^{T} \delta_t \,, \tag{4.3}$$

where $\delta_t = -f'(x_t; x^* - x_t) + \langle g_t, x^* - x_t \rangle$.

Proof. By definition,

$$\begin{aligned} f_t(x_t) - f_t(x^*) &= -\mathcal{B}_{f_t}(x^*, x_t) - f'(x_t; x^* - x_t) \\ &= \langle g_t, x_t - x^* \rangle - \mathcal{B}_{f_t}(x^*, x_t) + \delta_t \\ &= \langle g_t, x_{t+1} - x^* \rangle + \langle g_t, x_t - x_{t+1} \rangle - \mathcal{B}_{f_t}(x^*, x_t) + \delta_t \,. \end{aligned}$$

Summing over t completes the proof. □

Intuitively, the second term captures the regret due to the algorithm's inability to look ahead into the future.[5] The last two terms capture, respectively, the gain in regret that is possible due to the curvature of f_t , and the accuracy of the first-order (gradient) information g_t. In light of this lemma, controlling the regret reduces to controlling the individual terms in (4.3). A similar use of the forward regret appears in [1], though their analysis is limited to smooth loss functions.

Next, we provide upper bounds on $R_T^+(x^*)$ for a large class of online algorithms.

4.3 The algorithms: Ada-FTRL and Ada-MD

In this section, we analyze Ada-FTRL and Ada-MD. These two algorithms generalize the well-known core algorithms of online optimization: FTRL [38], [97] and MD [7], [25], [75], [107]. In particular, Ada-FTRL and Ada-MD capture variants of FTRL and MD such as Dual-Averaging [77], [109], AdaGrad [26], [70], composite-objective algorithms [25], [26], [109], implicit-update MD [52], [54], strongly-convex and non-linearized FTRL [40], [69], [82], [98], optimistic FTRL and MD [18], [50], [72], [88], [90], and even algorithms like AdaDelay [101] that violate the common non-decreasing regularization assumption existing in much of the previous work.[6]

4.3.1 Ada-FTRL: Generalized adaptive Follow-the-Regularized-Leader

The Ada-FTRL algorithm works with two sequences of *regularizers*, $p_1, p_2, \ldots, p_T$ and $q_0, q_1, q_2, \ldots, q_T$, where each p_t and q_t is a function from $\mathcal{H}$ to $\overline{\mathbb{R}}$. At time $t = 0, 1, 2, \ldots, T$,

[5] This is also related to the concept of "prediction drift", which appears in learning with delayed feedback [46], and to the role of stability in online algorithms [93].

[6] AdaDelay [101] is an algorithm for delayed-feedback distributed stochastic optimization. While this setting is beyond the scope of the current paper and is left for future work, we discuss the potential implications of our framework for AdaDelay in Section 4.10.

having received $(g_s)_{s=1}^t$, Ada-FTRL uses $g_{1:t}, p_{1:t}$ and $q_{0:t}$ to compute the next point x_{t+1}. The regularizers p_t and q_t can be built by Ada-FTRL in an online adaptive manner using the information generated up to the end of time step t (including g_t and x_t). In particular, we use p_t to distinguish the "proximal" part of this adaptive regularization: for all $t = 1, 2, \ldots, T$, we require that p_t (but not necessarily q_t) be minimized over $\mathcal{X}$ at x_t, that is[7],

$$p_t(x_t) = \inf_{x \in \mathcal{X}} p_t(x) < +\infty \,. \tag{4.4}$$

With the definitions above, for $t = 0, 1, 2, \ldots, T$, Ada-FTRL selects x_{t+1} such that

$$x_{t+1} \in \operatorname*{arg\,min}_{x \in \mathcal{X}} \; \langle g_{1:t}, x \rangle + p_{1:t}(x) + q_{0:t}(x) \,. \tag{4.5}$$

In particular, this means that the initial point x_1 satisfies[8]

$$x_1 \in \operatorname*{arg\,min}_{x \in \mathcal{X}} \; q_0(x) \,.$$

In addition, for notational convenience, we define $r_t := p_t + q_{t-1}, t = 1, 2, \ldots, T$, so that

$$x_{t+1} \in \operatorname*{arg\,min}_{x \in \mathcal{X}} \; \langle g_{1:t}, x \rangle + q_t(x) + r_{1:t}(x) \,. \tag{4.6}$$

Finally, we need to make a minimal assumption to ensure that Ada-FTRL is well-defined.

Assumption 4.1 (Well-posed Ada-FTRL)**.** *The functions q_0 and $p_t, q_t, t = 1, 2, \ldots, T$, are proper. In addition, for all $t = 0, 1, \ldots, T$, the* arg min *sets that define x_{t+1} in* (4.5) *are non-empty, and their optimal values are finite. Finally, for all $t = 1, 2, \ldots, T$, $r_{1:t}$ is directionally differentiable, and p_t satisfies* (4.4)*.*

Table 4.1 provides examples of several special cases of Ada-FTRL. In particular, Ada-FTRL combines, unifies and considerably extends the two major types of FTRL algorithms previously considered in the literature, that is, the so-called FTRL-Centered and FTRL-Prox algorithms [69] and their variants, as discussed in the subsequent sections.

[7] Note that x_t does not depend on p_t, but is rather computed using only $p_{1:t-1}$. Once x_t is calculated, p_t can be chosen so that (4.4) holds (and then used in computing x_{t+1}).

[8] The case of an arbitrary x_1 is equivalent to using, e.g., $q_0 \equiv 0$ (and changing q_1 correspondingly).

Algorithm	Regularization	Notes, Conditions and Assumptions
Online Gradient Descent (OGD)	$q_0 = \frac{1}{2\eta}\|.\|_2^2$ $q_t = p_t = 0$	$\mathcal{X} = \mathbb{R}^d, \eta > 0$ Update: $x_{t+1} = x_t - \eta g_t$
Dual Averaging (DA)	$q_t = \frac{\alpha_t}{2}\|.\|_2^2$ $p_t = 0$	$\alpha_{0:t} > 0, \alpha_t \ge 0 \ (t \ge 1)$
AdaGrad – Dual Averaging	$q_{0:t} = \frac{1}{2}\|x\|_{(t)}^2$ $p_t = 0$	$\|x\|_{(t)}^2 := \frac{1}{\eta} x^\top (Q_{0:t}^{1/2}) x$ $Q_0 = \gamma I$ $Q_{1:t} = \sum_{s=1}^t g_s g_s^\top$ (full-matrix update) $Q_{1:t}^{(j,j)} = \sum_{s=1}^t g_{s,j}^2$ (diagonal-matrix update)
FTRL-Prox	$q_t = 0$ $p_t = \frac{1}{2}\|x - x_t\|_{(t)}^2 - \frac{1}{2}\|x - x_t\|_{(t-1)}^2$	$Q_0 = 0$ Q_t and $\|\cdot\|_{(t)}$ as in AdaGrad-DA
Composite-Objective Online Learning	$q_0 = \tilde{q}_0$ $q_t = \psi_t + \tilde{q}_t$ $p_t = \tilde{p}_t$	For adding composite-objective learning to any instance of Ada-FTRL (see also Section 4.5) $x_{t+1} = \arg\min_{\mathcal{X}} \langle g_{1:t}, x\rangle + \psi_{1:t}(x) + \tilde{p}_{1:t}(x) + \tilde{q}_{0:t}(x)$

Table 4.1: Some special instances of Ada-FTRL; for more examples, see also the survey of McMahan [69].

4.3.2 Ada-MD: Generalized adaptive Mirror-Descent

As in Ada-FTRL, the Ada-MD algorithm uses two sequences of regularizer functions from $\mathcal{H}$ to $\overline{\mathbb{R}}$: $r_1, r_2, \ldots, r_T$ and $q_0, q_1, \ldots, q_T$. Further, we assume that the domains of (r_t) are non-increasing, that is, $\text{dom}(r_t) \subset \text{dom}(r_{t-1})$ for $t = 2, 3, \ldots, T$. Again, q_t, r_t can be created using the information generated by the end of time step t. The initial point x_1 of Ada-MD satisfies[9]

$$x_1 \in \arg\min_{x \in \mathcal{X}} \ q_0(x) .$$

Furthermore, at time $t = 1, 2, \ldots, T$, having observed $(g_s)_{s=1}^t$, Ada-MD uses g_t, q_t and $r_{1:t}$ to select the point x_{t+1} such that

$$x_{t+1} \in \arg\min_{x \in \mathcal{X}} \langle g_t, x\rangle + q_t(x) + \mathcal{B}_{r_{1:t}}(x, x_t) . \tag{4.7}$$

In addition, similarly to Ada-FTRL, we define $p_t := r_t - q_{t-1}, t = 1, 2, \ldots, T$, though we do not require p_t to be minimized at x_t in Ada-MD.[10]

[9] The case of an arbitrary x_1 is equivalent to using, e.g., $q_0 \equiv 0$ (and changing r_1 correspondingly).
[10] We use the convention $(+\infty) - (+\infty) = +\infty$ in defining p_t.

Finally, we present our assumption on the regularizers of Ada-MD. Compared to Ada-FTRL, we require a stronger assumption to ensure that Ada-MD is well-defined, and that the Bregman divergences in (4.7) have a controlled behavior.

Assumption 4.2 (Well-posed Ada-MD)**.** *The regularizers $q_t, r_t, t = 1, 2, \dots, T$, are proper, and q_0 is directionally differentiable. In addition, for all $t = 0, 1, \dots, T$, the* arg min *sets that define x_{t+1} in* (4.7) *are non-empty, and their optimal values are finite. Finally, for all $t = 1, 2, \dots, T$, $q_t, r_{1:t}$, and $r_{1:t} + q_t$ are directionally differentiable, $x_t \in \operatorname{dom}(r_{1:t})$, and $r'_{1:t}(x_t; \cdot)$ is linear in the directions inside* $\operatorname{dom}(r_{1:t})$*, that is, there is a vector in $\mathcal{H}$, denoted by $\nabla r_{1:t}(x_t)$, such that $r'_{1:t}(x_t, x - x_t) = \langle \nabla r_{1:t}(x_t), x - x_t \rangle$ for all $x \in \operatorname{dom}(r_{1:t})$.*

Remark 4.1. *Our results also hold under the weaker condition that $r'_{1:t}(x_t; \cdot - x_t)$ is* concave[11] *(rather than linear) on* $\operatorname{dom}(r_{1:t})$*. However, in case of a convex $r_{1:t}$, this weaker condition would again translate into having a linear $r'_{1:t}$, because a convex $r_{1:t}$ implies a convex $r'_{1:t}$ [6, Proposition 17.2]. While we do not* require *that $r_{1:t}$ be convex, all of our subsequent examples in the paper use convex $r_{1:t}$. Thus, in the interest of readability, we have made the stronger assumption of linear directional derivatives here.*

Remark 4.2. *Note that $r'_{1:t}$ needs to be linear only in the directions inside the domain of $r_{1:t}$. As such, we avoid the extra technical conditions required in previous work, e.g., that $r_{1:t}$ be a Legendre function to ensure x_t remains in the interior of* $\operatorname{dom}(r_{1:t})$ *and $\nabla r_{1:t}(x_t)$ is well-defined.*

4.3.3 Analysis of Ada-FTRL and Ada-MD

Next we present bounds on the forward regret of Ada-FTRL and Ada-MD, and discuss their implications; the proof is provided in Appendix 4.F.

Theorem 4.1 (Forward regret of Ada-FTRL and Ada-MD)**.** *For any $x^* \in \mathcal{X}$ and any sequence of linear losses $\langle g_t, \cdot \rangle, t = 1, 2, \dots, T$, the forward regret of* Ada-FTRL *under Assumption 4.1 satisfies*

$$R_T^+(x^*) \le \sum_{t=0}^{T} (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) - \sum_{t=1}^{T} \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) , \qquad (4.8)$$

[11] Without such assumptions, a Bregman divergence term in $r'_{1:t}$ appears in the regret bound of Ada-MD. Concavity ensures that this term is not positive and can be dropped, greatly simplifying the bounds.

whereas the forward regret of Ada-MD *under Assumption 4.2 satisfies*

$$R_T^+(x^*) \le \sum_{t=0}^{T} (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^{T} \mathcal{B}_{p_t}(x^*, x_t) - \sum_{t=1}^{T} \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t)\,. \tag{4.9}$$

Remark 4.3. *Theorem 4.1 does not require the regularizers to be non-negative or (even non-strongly) convex.*[12] *This opens up the possibility to accommodate a wide range of non-standard algorithmic solutions, such as the application of non-monotonic learning rates; see also Section 4.10 for a discussion.*

Remark 4.4. *In practice,* Ada-FTRL *and* Ada-MD *need to pick a specific x_{t+1} from the possibly multiple optima of* (4.5) *and* (4.7)*. The bounds of Theorem 4.1 apply irrespective of the tie-breaking scheme.*

In subsequent sections, we show that the generality of Ada-FTRL and Ada-MD, together with the flexibility of Assumptions 4.1 and 4.2, considerably facilitates the handling of various algorithmic ideas and problem settings, and allows us to combine them without requiring a new analysis for each new combination.

4.4 Recoveries and extensions

Lemma 4.1 and Theorem 4.1 together immediately result in generic upper bounds on the regret, given in (4.25) and (4.26) in Appendix 4.A. Under different assumptions on the losses and regularizers, these generic bounds directly translate into concrete bounds for specific learning settings. We explore these concrete bounds in the rest of this section.

First, we provide a list of the assumptions on the losses and the regularizers for different learning settings.[13] We consider two special cases of the setting of Section 4.2: Online optimization and stochastic optimization. In online optimization, we make the following assumption:

Assumption 4.3 (Online optimization setting)**.** *For $t = 1, 2, \ldots, T$, f_t is locally sub-differentiable, and g_t is a local sub-gradient of f_t at x_t.*

[12]Nevertheless, such assumptions are useful when combining the theorem with Lemma 4.1.

[13]In fact, compared to previous work (e.g., the references listed in Section 4.1 and Section 4.3), these are typically relaxed versions of the usual assumptions.

Note that f_t may be non-convex, and g_t does not need to define a global lower-bound (i.e., be a sub-gradient) of f_t; see Section 4.1.2 for the formal definition of local sub-gradients.

Recall that the stochastic optimization setting is concerned with minimizing a function f, defined by $f(x) := \mathbb{E}_{\xi\sim D}F(x,\xi)$. In this case our performance metric is redefined to be the expected stochastic regret, $\mathbb{E}\{R_T(x^*)\} = \mathbb{E}\Big\{\sum_{t=1}^{T}(f(x_t) - f(x^*))\Big\}$ (see Section 4.2). Typically, if F is differentiable in x, then $g_t = \nabla F(x_t, \xi_t)$, where ξ_t is a random variable, e.g., sampled independently from D. In parallel to Assumption 4.3, we summarize our assumptions for this setting is as follows:

Assumption 4.4 (Stochastic optimization setting). *The function f (defined above) is locally sub-differentiable, $f_t = f$ for all $t = 1, 2, \ldots, T$, and g_t is, in expectation, a local sub-gradient of f at x_t: $\mathbb{E}\{g_t|x_t\} \in \delta f(x_t)$.*

Again, it is enough for g_t to be an unbiased estimate of a local sub-gradient (Section 4.1.2).

In both settings we will rely on the non-negativity of the loss divergences at x^*:

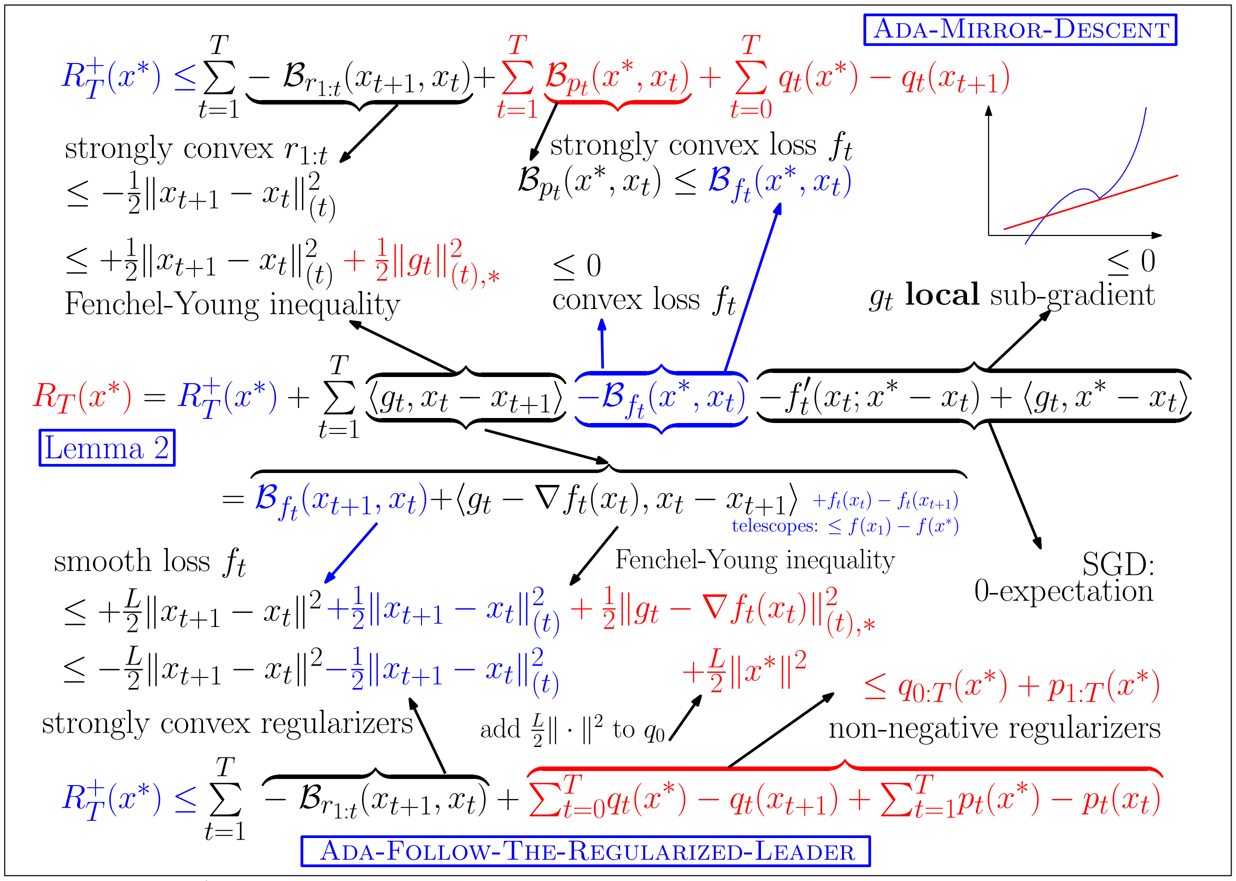
Assumption 4.5 (Non-negative loss-divergence). *For the competitor point of interest x^* (with respect to which we aim to bound $R_T(x^*)$), and for all $t = 1, 2, \ldots, T$, $\mathcal{B}_{f_t}(x^*, x_t) \geq 0$.*

It is well known that this assumption is satisfied when each f_t is convex. However, as we shall see in Section 4.9, this condition also holds for certain classes of non-convex functions (e.g., star-convex functions and more). In the stochastic optimization setting, since $f_t = f$, this condition boils down to $\mathcal{B}_f(x^*, x_t) \geq 0$, $t = 1, 2, \ldots, T$.

In both settings, the regret can be reduced when the losses are strongly convex. Furthermore, in the stochastic optimization setting, the smoothness of the loss is also helpful in decreasing the regret. The next two assumptions capture these conditions.

Assumption 4.6 (Loss smoothness). *The function f is differentiable and 1-smooth w.r.t. some norm $\|\cdot\|$.*

Assumption 4.7 (Loss strong convexity). *The losses are 1-strongly convex w.r.t. the regularizers, that is, $\mathcal{B}_{f_t}(x^*, x_t) \geq \mathcal{B}_{p_t}(x^*, x_t)$ for $t = 1, 2, \ldots, T$.*



Ada-Mirror-Descent
$R_T^+(x^*) \le \sum_{t=1}^T \underbrace{-\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t)} + \sum_{t=1}^T \underbrace{\mathcal{B}_{p_t}(x^*, x_t)} + \sum_{t=0}^T q_t(x^*) - q_t(x_{t+1})$
strongly convex $r_{1:t}$
$\le -\frac{1}{2}\|x_{t+1} - x_t\|_{(t)}^2$
strongly convex loss f_t
$\mathcal{B}_{p_t}(x^*, x_t) \le \mathcal{B}_{f_t}(x^*, x_t)$
$\le +\frac{1}{2}\|x_{t+1} - x_t\|_{(t)}^2 + \frac{1}{2}\|g_t\|_{(t),*}^2$
Fenchel-Young inequality
≤ 0
convex loss f_t
≤ 0
g_t **local** sub-gradient
$R_T(x^*) = R_T^+(x^*) + \sum_{t=1}^T \underbrace{\langle g_t, x_t - x_{t+1}\rangle} \underbrace{-\mathcal{B}_{f_t}(x^*, x_t)} \underbrace{-f_t'(x_t; x^* - x_t) + \langle g_t, x^* - x_t\rangle}$
Lemma 2
$= \mathcal{B}_{f_t}(x_{t+1}, x_t) + \langle g_t - \nabla f_t(x_t), x_t - x_{t+1}\rangle$ $+f_t(x_t) - f_t(x_{t+1})$
telescopes: $\le f(x_1) - f(x^*)$
smooth loss f_t
Fenchel-Young inequality
SGD: 0-expectation
$\le +\frac{L}{2}\|x_{t+1} - x_t\|^2 + \frac{1}{2}\|x_{t+1} - x_t\|_{(t)}^2 + \frac{1}{2}\|g_t - \nabla f_t(x_t)\|_{(t),*}^2$
$\le -\frac{L}{2}\|x_{t+1} - x_t\|^2 - \frac{1}{2}\|x_{t+1} - x_t\|_{(t)}^2$
$+\frac{L}{2}\|x^*\|^2$
$\le q_{0:T}(x^*) + p_{1:T}(x^*)$
strongly convex regularizers
add $\frac{L}{2}\|\cdot\|^2$ to q_0
non-negative regularizers
$R_T^+(x^*) \le \sum_{t=1}^T \underbrace{-\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t)} + \underbrace{\sum_{t=0}^T q_t(x^*) - q_t(x_{t+1}) + \sum_{t=1}^T p_t(x^*) - p_t(x_t)}$
Ada-Follow-The-Regularized-Leader

Figure 4.2: A summary of the proof techniques that incorporate each of the assumptions into regret bounds; see Corollaries 4.5–4.7 and Table 4.2. The identity in the middle is from Lemma 4.1, whereas the top and bottom bounds on R_T^+ are due to Theorem 4.1. Each arrow shows the transformation of one of the terms, using the stated assumption or technique. The matching terms cancel, and the terms shown in red appear in the final bounds.

Note that if p_t, q_{t-1} are convex, then it suffices to have $\mathcal{B}_{r_t}$ in the condition (rather than $\mathcal{B}_{p_t}$). Typically, if f_t is strongly convex w.r.t. a norm $\|\cdot\|_{(t)}$, then p_t (or r_t) is set to $\eta\|\cdot\|_{(t)}^2$ for some $\eta > 0$. Again, in stochastic optimization, Assumption 4.7 simplifies to $\mathcal{B}_f(x^*, x_t) \ge \mathcal{B}_{p_t}(x^*, x_t)$, $t = 1, 2, \ldots, T$. Furthermore, if p_t is convex, then Assumption 4.7 implies that f_t is convex.

Finally, the results that we recover depend on the assumption that the total regularization, in both Ada-FTRL and Ada-MD, is strongly convex:

Assumption 4.8 (Strong convexity of regularizers). *For all $t = 1, 2, \ldots, T$, $r_{1:t}$ is 1-strongly convex w.r.t. some norm $\|\cdot\|_{(t)}$.*

Table 4.2 provides a summary of the standard results, under different sub-sets of

Setting / Algorithms	Assumptions	Regret / Expected Stochastic Regret Bound
OO/SO Ada-FTRL	4.1, 4.3/4.4, 4.5, 4.8	$\sum_{t=0}^{T}(q_t(x^*) - q_t(x_{t+1}))$ $+\sum_{t=1}^{T}(p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T}\frac{1}{2}\|g_t\|_{(t),*}^2$
OO/SO Ada-MD	4.2, 4.3/4.4, 4.5, 4.8	$\sum_{t=0}^{T}(q_t(x^*) - q_t(x_{t+1}))$ $+\sum_{t=1}^{T}\mathcal{B}_{p_t}(x^*, x_t) + \frac{1}{2}\|g_t\|_{(t),*}^2$
Strongly-convex OO/SO Ada-MD	4.2, 4.3/4.4, (4.5), 4.8, 4.7	$\sum_{t=0}^{T}(q_t(x^*) - q_t(x_{t+1}))$ $+\sum_{t=1}^{T}\frac{1}{2}\|g_t\|_{(t),*}^2$
Smooth SO Ada-FTRL	4.1, 4.4, 4.6, 4.5, 4.8’	$\sum_{t=0}^{T}(q_t(x^*) - q_t(x_{t+1})) + D$ $+\sum_{t=1}^{T}(p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T}\frac{1}{2}\|\sigma_t\|_{(t),*}^2$
Smooth SO Ada-MD	4.2, 4.4, 4.6, 4.5, 4.8’	$\sum_{t=0}^{T}(q_t(x^*) - q_t(x_{t+1})) + D$ $+\sum_{t=1}^{T}\mathcal{B}_{p_t}(x^*, x_t) + \frac{1}{2}\|\sigma_t\|_{(t),*}^2$
Smooth & strongly-convex SO Ada-MD	4.2, 4.4, 4.6, (4.5), 4.8’, 4.7	$\frac{1}{2}\|x^* - x_1\|^2 + \sum_{t=0}^{T}(q_t(x^*) - q_t(x_{t+1}))$ $+D + \sum_{t=1}^{T}\frac{1}{2}\|\sigma_t\|_{(t),*}^2$

Table 4.2: Recovered and generalized standard results for online optimization (OO) and stochastic optimization (SO); see Corollaries 4.5–4.7. A number in parentheses indicates that the assumption is not directly required, but is implied by the other assumptions. In the bounds above, $\sigma_t := g_t - \nabla f(x_t)$, $D := f(x_1) - \inf_{\mathcal{X}} f(x)$, and 4.8’ refers to a slightly modified version of Assumption 4.8, as described in Corollary 4.7. Note that setting $q_T = 0$ recovers the *off-by-one* property [69] in FTRL-Centered vs. FTRL-Prox; Ada-MD exhibits a similar property. For composite-objective Ada-FTRL and Ada-MD, these same bounds apply with $\tilde{q}_t$ in place of q_t in the summation; see Table 4.1 and Section 4.5.

the assumptions above, that are recovered and generalized using our framework. The derivations of these results are provided in the form of three corollaries in Appendix 4.A. Note that the analysis is absolutely modular: each assumption is simply plugged into (4.25) or (4.26) to obtain the final bounds, without the need for a separate analysis of Ada-FTRL and Ada-MD for each individual setting. A schematic view of the (standard) proof ideas is given in Figure 4.2. Finally, in Table 4.3, we provide examples of concrete bounds for common special cases of Ada-FTRL and Ada-MD recovered by the generic bounds in Table 4.2; see also Table 4.1.

Algorithm / Regularizer	Setting & Assumptions	Regret / Expected Stochastic Bound
Online Gradient Descent Ada-FTRL $q_0 = \frac{1}{2\eta}\|.\|_2^2$, $q_t = p_t = 0$	Convex OO / SO $\|x^*\|_2 \leq R$ $\mathbb{E}\{\|g_t\|_2^2 \| x_t\} \leq G^2$	$\frac{1}{2\eta}R^2 + \frac{\eta}{2}TG^2$ OO: $R_T \leq RG\sqrt{T}$ for $\eta = \frac{R}{G\sqrt{T}}$ SO: $\mathcal{R}_T \leq RG/\sqrt{T}$
Original bounds: Nemirovsky and Yudin [75], Warmuth and Jagota [107], and Zinkevich [111] Optimality: Agarwal *et al.* [2]		
Online Gradient Descent Ada-MD $q_t = 0$, $p_t = \frac{\beta}{2}\|\cdot\|_2^2$	Strongly convex OO / SO f_t β-strongly-convex $\mathbb{E}\{\|g_t\|_2^2 \| x_t\} \leq G^2$	OO: $R_T \leq \sum_{t=1}^T \frac{G^2}{\beta t} \leq \frac{G^2(1+\log(T))}{\beta}$ SO: $\mathcal{R}_T \leq \frac{G^2}{\beta T}(1 + \log(T))$
Original bounds: Hazan and Kale [39] and Hazan *et al.* [40] Optimality: Agarwal *et al.* [2]		
Dual Averaging (DA) Ada-FTRL $q_{0:t} = \frac{L+\alpha_t}{2}\|.\|_2^2$ $p_t = 0$	Convex smooth SO f convex and L-smooth $\mathbb{E}\{\|\sigma_t\|_2^2 \| x_t\} \leq \sigma^2$ $\|x^*\|_2 \leq R$	$R_T \leq \frac{(L+\alpha_{T-1})R^2}{2} + D + \sum_{t=1}^T \frac{\sigma^2}{2\alpha_{t-1}}$ Using $\alpha_{t-1} = \frac{\sigma}{R}\sqrt{2t}$: $\mathcal{R}_T \leq \frac{LR^2+2D}{2T} + \frac{\sqrt{2}\sigma R}{\sqrt{T}}$
Original bound: Dekel *et al.* [24] Suboptimality: Nemirovsky and Yudin [75]		
AdaGrad - MD (diagonal) $p_{1:t} = \frac{1}{2\eta}x^\top(Q_{1:t}^{1/2})x$ $Q_{1:t}^{(j,j)} = \sum_{s=1}^t g_{s,j}^2$, $q_t = 0$	Convex OO / SO $\|x - y\|_\infty \leq R, \forall x, y \in \mathcal{X}$	$R_T \leq (\frac{1}{2\eta}R^2 + \eta)\sum_{i=1}^d \sqrt{\sum_{t=1}^T g_{t,i}^2}$ Using $\eta = R/\sqrt{2}$: $\mathcal{R}_T \leq \frac{\sqrt{2}R}{T}\mathbb{E}\Big\{\sum_{i=1}^d \|g_{1:T,i}\|_2\Big\}$
Original bounds: Duchi *et al.* [26] and McMahan and Streeter [70] Optimality: McMahan and Streeter [70]		
AdaGrad - DA (full) $q_{0:t} = \frac{1}{2\eta}x^\top(Q_{0:t}^{1/2})x$ $Q_{1:t} = \sum_{s=1}^t g_s g_s^\top$ $Q_0 = GI$ and $p_t = 0$	Convex OO / SO $\|x^*\|_2 \leq R$ $\|g_t\|_2 \leq G$	$R_T \leq \frac{G}{\eta}R^2 + (\frac{1}{\eta}R^2 + \eta)\mathrm{tr}(Q_{1:T}^{1/2})$ Using $\eta = R$: $\mathcal{R}_T \leq \frac{GR}{T} + \frac{2R}{T}\mathbb{E}\Big\{\mathrm{tr}(Q_{1:T}^{1/2})\Big\}$
Original bounds: Duchi *et al.* [26] and McMahan and Streeter [70] Optimality: McMahan and Streeter [70]		

Table 4.3: Concrete bounds for online optimization (OO) and stochastic optimization (SO) derived from the generic bounds of Table 4.2; references are given to the original bounds we recover, as well as to results showing their optimality or suboptimality. In all bounds we assume $\mathcal{H} = \mathbb{R}^d$. For SO, $\hat{x}_T$ is the average iterate $x_{1:T}$ or picked uniformly at random; see the discussion before Section 4.2.1. For smooth optimization, D and σ are defined as in Table 4.2, and $\mathcal{R}_T := \mathbb{E}\{f(\hat{x}_T) - f(x^*)\}$ denotes the stochastic optimization risk. Derivations are straightforward, involving plugging in the parameters and/or applying specific results. In particular, the second line uses the basic inequality $\sum_{t=1}^T(1/t) \leq (1 + \log(T))$, the third line uses $\sum_{t=1}^T(1/\sqrt{t}) \leq 2\sqrt{T}$, and the fourth and fifth lines use, respectively, Lemmas 4 and 10 of Duchi *et al.* [26].

4.5 Composite-objective learning and optimization

Next, we consider the composite-objective online learning setting. In this setting, the functions f_t, from which the (local sub-)gradients g_t are generated and fed to the algorithm, comprise only part of the loss. Instead of $R_T(x^*)$, we are interested in minimizing the regret

$$R_T^{(\ell)}(x^*) := \sum_{t=1}^{T} f_t(x_t) + \psi_t(x_t) - f_t(x^*) - \psi_t(x^*) = R_T(x^*) + \sum_{t=1}^{T} \psi_t(x_t) - \psi_t(x^*)\,,$$

using the feedback $g_t \in \delta f_t(x_t)$, where $\psi_t : \mathcal{H} \to \overline{\mathbb{R}}$ are proper functions. The functions ψ_t are not linearized, but are passed directly to the algorithm.

Naturally, one can use the q_t regularizers to pass the ψ_t functions to Ada-FTRL and Ada-MD. Then, we can obtain the exact same bounds as in Table 4.2 on the composite regret $R_T^{(\ell)}(x^*)$; this recovers and extends the corresponding bounds by Duchi *et al.* [25], Duchi *et al.* [26], McMahan [69], and Xiao [109]. In particular, consider the following two scenarios:

Setting 1: ψ_t is known before predicting x_t. In this case, we run Ada-FTRL or Ada-MD with $q_t = \psi_{t+1} + \tilde{q}_t, t = 0, 1, 2, \ldots, T$ (where $\psi_{T+1} := 0$). Thus, we have the update

$$x_{t+1} \in \operatorname*{arg\,min}_{x\in\mathcal{X}} \langle g_{1:t}, x\rangle + \psi_{1:t+1}(x) + \tilde{q}_{0:t}(x) + p_{1:t}(x)\,, \tag{4.10}$$

for Ada-FTRL, and

$$x_{t+1} \in \operatorname*{arg\,min}_{x\in\mathcal{X}} \langle g_t, x\rangle + \psi_{t+1}(x) + \tilde{q}_t(x) + \mathcal{B}_{r_{1:t}}(x, x_t)\,, \tag{4.11}$$

for Ada-MD. Then, we have the following result.

Corollary 4.1. *Suppose that the iterates $x_1, x_2, \ldots, x_{T+1}$ are given by the* Ada-FTRL *update* (4.10) *or the* Ada-MD *update* (4.11)*, and q_t, p_t, and r_t satisfy Assumption 4.1 for* Ada-FTRL*, or Assumption 4.2 for* Ada-MD*. Then, under the conditions of each section of Corollaries 4.5 to 4.7, the composite regret $R_T^{(\ell)}(x^*)$ enjoys the same bound as $R_T(x^*)$, but with $\sum_{t=0}^T \tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})$ in place of $\sum_{t=0}^T q_t(x^*) - q_t(x_{t+1})$.*

Proof. By definition, $\sum_{t=0}^T \tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1}) - q_t(x^*) + q_t(x_{t+1}) = \sum_{t=1}^T \psi_t(x_t) - \psi_t(x^*)$. Thus, $R_T^{(\ell)}(x^*) = R_T(x^*) + \sum_{t=0}^T \tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1}) - q_t(x^*) + q_t(x_{t+1})$. Upper-bounding $R_T(x^*)$ by the aforementioned corollaries completes the proof. □

Setting 2: ψ_t is revealed after predicting x_t, together with g_t. In this case, we run Ada-FTRL and Ada-MD with functions $q_0 = \tilde{q}_0$, $q_t = \psi_t + \tilde{q}_t, t = 1, 2, \dots, T$, so that

$$x_{t+1} \in \operatorname*{arg\,min}_{x \in \mathcal{X}} \langle g_{1:t}, x \rangle + \psi_{1:t}(x) + \tilde{q}_{0:t}(x) + p_{1:t}(x) , \tag{4.12}$$

for Ada-FTRL, and

$$x_{t+1} \in \operatorname*{arg\,min}_{x \in \mathcal{X}} \langle g_t, x \rangle + \psi_t(x) + \tilde{q}_t(x) + \mathcal{B}_{r_{1:t}}(x, x_t) , \tag{4.13}$$

for Ada-MD. Then, we have the following result, proved in Appendix 4.B.

Corollary 4.2. *Suppose that the iterates $x_1, x_2, \dots, x_{T+1}$ are given by the* Ada-FTRL *update* (4.12) *or the* Ada-MD *update* (4.13)*, and q_t, p_t, and r_t satisfy Assumption 4.1 for* Ada-FTRL*, or Assumption 4.2 for* Ada-MD*. Also, assume that $\psi_1(x_1) = 0$ and the ψ_t are non-negative. Finally, suppose that the ψ_t form a non-increasing sequence of functions, that is, $\psi_1 \geq \psi_2 \geq \cdots \geq \psi_{T+1} := 0$.*[14] *Then, under the conditions of each section of Corollaries 4.5 to 4.7, the composite regret $R_T^{(\ell)}(x^*)$ enjoys the same bound as $R_T(x^*)$, but with $\sum_{t=0}^T \tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})$ in place of $\sum_{t=0}^T q_t(x^*) - q_t(x_{t+1})$.*

Remark 4.5. *In both settings, the functions ψ_t are passed as part of the regularizers q_t. Thus, if the ψ_t are strongly convex, less additional regularization is needed in* Ada-FTRL *to ensure the strong convexity of $r_{1:t}$ because $q_{0:t-1}$ will already have some strongly convex components. In addition, in* Ada-MD*, when the ψ_t are convex, the $\mathcal{B}_{p_t}$ terms in* (4.9) *will be smaller than the $\mathcal{B}_{r_t}$ terms found in previous analyses of MD. This is especially useful for implicit updates, as shown in the next section. This also demonstrates another benefit of the generalized Bregman divergence: the ψ_t, and hence the p_t, may be non-smooth in general.*

4.6 Implicit-update Ada-MD and non-linearized Ada-FTRL

Other learning settings can be captured using the idea of passing information to the algorithm using the q_t functions. This information could include, for example, the curvature of the loss. In particular, consider the composite-objective Ada-FTRL and Ada-MD, and

[14] This relaxes the assumption in the literature, e.g., by McMahan [69], that $\psi_t = \alpha_t \psi$ for some fixed, non-negative ψ minimized at x_1, and a non-increasing sequence $\alpha_t > 0$ (e.g., $\alpha_t = 1$); see also Setting 1.

for $t = 1, 2, \ldots, T$, let ℓ_t be a differentiable loss, $f_t = \langle \nabla \ell_t(x_t), x - x_t \rangle$, and $\psi_t = \mathcal{B}_{\ell_t}(\cdot, x_t)$.[15] Then, $\ell_t = f_t + \psi_t$, $g_t = \nabla f_t(x_t) = \nabla \ell_t(x_t)$, and the composite-objective Ada-FTRL update (4.12) is equivalent to

$$x_{t+1} \in \arg\min_{x \in \mathcal{X}} \; \ell_{1:t}(x) + \tilde{q}_{0:t}(x) + p_{1:t}(x)\,. \tag{4.14}$$

Thus, non-linearized FTRL, studied by McMahan [69], is a special case of Ada-FTRL. With the same f_t, ψ_t, the composite-objective Ada-MD update (4.13) is equivalent to

$$x_{t+1} \in \arg\min_{x \in \mathcal{X}} \; \ell_t(x) + \tilde{q}_t(x) + \mathcal{B}_{r_{1:t}}(x, x_t)\,, \tag{4.15}$$

so the implicit-update MD is also a special case of Ada-MD.

Again, combining Lemma 4.1 and Theorem 5.6 results in a compact analysis of these algorithmic ideas. In particular, for both updates (4.14) and (4.15), the bounds of (4.25) and (4.26) apply on the regret in f_t. Then, moving the terms $\psi_t(x^*)$ to the left turns each bound to a bound on the regret in ℓ_t. Furthermore, the terms $-\psi_t(x_{t+1}) = -\mathcal{B}_{\ell_t}(x_{t+1}, x_t)$ that remained on the right-hand side can be merged with the $-\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t)$ terms. Thus, instead of $r_{0:t}$, it is enough for $r_{1:t} + \ell_t$ to be strongly convex w.r.t. the norm $\|\cdot\|_{(t)}$ (see the proofs of Corollaries 4.5 to 4.7). This means that if ℓ_t are strongly convex, then no further regularization is required: $\ell_{1:t}$ is strongly convex, and we get back the well-known logarithmic bounds for strongly-convex FTRL (Follow-The-Leader) and implicit-update MD [40], [52], [54], [69], [82], [98]. In addition, as mentioned before, convexity of ℓ_t further reduces the term $\mathcal{B}_{p_t}$ in implicit-update MD.

Finally, note that multiple pieces of information can be passed to the algorithm through q_t. In particular, none of the above interfere with further use of another composite term ϕ_t and obtaining regret bounds on $\ell_t + \phi_t$, as discussed in Section 4.5.

4.7 Adaptive optimistic learning & variational bounds

The goal of optimistic online learning algorithms [88], [90] is to obtain improved regret bounds when playing against "easy" (i.e., predictable) sequences of losses. This includes algorithms with regret rates that grow with the total "variation", that is, the sum of the

[15] For non-differentiable ℓ_t, let $f_t = \langle g_t, \cdot \rangle$ and $\psi_t = \mathcal{B}_{\ell_t}(\cdot, x_t) + \ell'(x_t; \cdot - x_t) - \langle g_t, \cdot \rangle$ to get the same effect.

norms of the differences between pairs of consecutive losses f_t and f_{t+1} observed in the loss sequence: the regret will be small if the loss sequence changes slowly [18].

Recently, Mohri and Yang [72] proposed an interesting comprehensive framework for analyzing adaptive FTRL algorithms for predictable sequences. The framework has also been extended to MD by Kamalaruban [50]. However, despite their generality, the regret analyses of Mohri and Yang [72] and Kamalaruban [50] can be strengthened. Specifically, the two analyses do not recover the variation-based results of Chiang *et al.* [18] for smooth losses. In addition, their treatment of composite objectives introduces complications, e.g., only applies to Setting 1 of Section 4.5 where ψ_t is known before selecting x_t.

The flexibility of the framework introduced in this paper allows us to alleviate these and other limitations. In particular, we cast the Adaptive Optimistic FTRL (AO-FTRL) algorithm of Mohri and Yang [72] as a special case of Ada-FTRL, and obtain a much simpler form of Adaptive Optimistic MD (AO-MD) as a special case of Ada-MD. Then, we strengthen and simplify the corresponding analyses, and recover and extend the results of Chiang *et al.* [18]. Finally, building on the modularity of our framework, we obtain an adaptive composite-objective algorithm with variational bounds that improves upon the results of Chiang *et al.* [18], Kamalaruban [50], Mohri and Yang [72], Rakhlin and Sridharan [88], and Rakhlin and Sridharan [90].

4.7.1 Adaptive optimistic FTRL

Consider the online optimization setting of Section 4.4 (Assumption 4.3). Suppose that the losses $f_1, f_2, \ldots, f_T$ satisfy Assumption 4.5 (e.g., they are convex), and the sequence of points $x_{t+1}, t = 0, 1, 2, \ldots, T$ is given by

$$x_{t+1} \in \operatorname*{arg\,min}_{x \in \mathcal{X}} \langle g_{1:t} + \tilde{g}_{t+1}, x \rangle + p_{1:t}(x) + \tilde{q}_{0:t}(x) \,,$$

where $\tilde{g}_t, t = 1, 2, \ldots, T+1$, is any sequence of vectors in $\mathcal{H}$. That is, we run Ada-FTRL, but we also incorporate $\tilde{g}_{t+1}$ as a "guess" of the future loss g_{t+1} that the algorithm will suffer. Mohri and Yang [72] refer to this algorithm as AO-FTRL.

It is easy to see that AO-FTRL is a special case of Ada-FTRL: Define $\tilde{g}_0 := 0$,[16] and

[16] This is different from the restriction in Mohri and Yang [72] that $\tilde{g}_1$ be 0; we do not require that restriction. In particular, we allow x_1 to depend on $\tilde{g}_1$, which can be arbitrary.

for $t = 0, 1, \ldots, T$, let $q_t = \tilde{q}_t + \langle \tilde{g}_{t+1} - \tilde{g}_t, \cdot \rangle$. Then, $q_{0:t} = \tilde{q}_{0:t} + \langle \tilde{g}_{t+1}, \cdot \rangle$, so we have

$$x_{t+1} \in \operatorname*{arg\,min}_{x \in \mathcal{X}} \langle g_{1:t}, x \rangle + p_{1:t}(x) + q_{0:t}(x)\,,$$

which is the Ada-FTRL update with this specific choice of q_t. Thus, the exact same manipulations as in Corollary 4.5 give the following theorem, proved in Appendix 4.C:

Theorem 4.2. *If the losses satisfy Assumption 4.5, and the regularizers q_0 and $p_t, q_t, t = 1, 2, \ldots, T$, satisfy Assumptions 4.1 and 4.8, then the regret of AO-FTRL is bounded as*

$$R_T(x^*) \le \sum_{t=0}^{T-1} (\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T} \frac{1}{2} \| g_t - \tilde{g}_t \|_{(t),*}^2\,. \tag{4.16}$$

This bound recovers Theorems 1 and 2 of Mohri and Yang [72]. Similarly, one could prove parallels of Corollaries 4.6 and 4.7 for AO-FTRL. Then, the modularity property allows us (as we do in Section 4.7.4) to apply the composite-objective technique of Section 4.5 and recover Theorems 3-7 of Mohri and Yang [72] (and hence their corollaries). Indeed, the resulting analysis simplifies and improves on the analysis of Mohri and Yang [72] in several aspects: we do not need to separate the cases for FTRL-Prox and FTRL-General, we naturally handle the composite objective case for Settings 1 and 2 while avoiding any complications with proximal regularizers, and do not lose the constant 1/2 factor. Finally, Theorem 4.1 allows us to improve on the results of Chiang *et al.* [18], as we show in Section 4.7.3.

4.7.2 Adaptive optimistic MD

Interestingly, we can use the exact same assignment $q_t = \tilde{q}_t + \langle \tilde{g}_{t+1} - \tilde{g}_t, \cdot \rangle$ in Ada-MD. This results in the update

$$x_{t+1} \in \operatorname*{arg\,min}_{x \in \mathcal{X}} \langle g_t + \tilde{g}_{t+1} - \tilde{g}_t, x \rangle + \tilde{q}_t(x) + \mathcal{B}_{r_{1:t}}(x, x_t)\,.$$

Applying the same argument as in Theorem 4.2, one can show that this optimistic MD algorithm enjoys the regret bound of (4.16) with the $p_t(x^*) - p_t(x_t)$ terms replaced by $\mathcal{B}_{p_t}(x^*, x_t)$. This gives an optimistic MD algorithm with only one projection in each round; all other formulations [18], [50], [88], [90] require two MD steps in each round. This new formulation has the potential to greatly simplify the previous analyses of variants of optimistic MD. In particular, handling implicit updates or composite terms is a matter of including them in $\tilde{q}_t$. Especially, unlike Kamalaruban [50], we can handle Setting 2 in the exact same way as we do in the AO-FTRL case (see Section 4.7.4). Further exploration of the properties of this new class of algorithms is left for future work.

4.7.3 Variation-based bounds for online learning

Suppose that the losses f_t are differentiable and convex, and define $f_0 := 0$. For any norm $\|\cdot\|$, we define the total variation of the losses in $\|\cdot\|_*^2$ as

$$D_{\|\cdot\|} := \sum_{t=1}^{T} \sup_{x \in \mathcal{X}} \|\nabla f_t(x) - \nabla f_{t-1}(x)\|_*^2 . \tag{4.17}$$

Chiang *et al.* [18] use an optimistic MD algorithm to obtain regret bounds of order $O(\sqrt{D_2})$, where $D_2 = D_{\|\cdot\|_2}$, for linear as well as smooth losses.

If the losses are linear, that is, $f_t = \langle g_t, \cdot \rangle$, then Theorem 4.2 immediately recovers the result of Chiang *et al.* [18, Theorem 8]. In particular, let $\tilde{q}_0 = (1/2\eta)\|.\|_2^2$, and for $t = 1, 2, \ldots, T$, let $p_t = \tilde{q}_t = 0$, $\|.\|_{(t)}^2 = \eta\|.\|_2^2$, and $\tilde{g}_t = g_{t-1}$. Then (4.16) gives the regret bound $(\eta/2)\|x^*\|_2^2 + (1/(2\eta))D_2$. If $\|x\|_2 \leq 1$ and we set η based on D_2 (as Chiang et al. assume), we obtain their $O(\sqrt{D_2})$ bound.

If the losses are not linear but are L-smooth, then by the combination of Lemma 4.1 and Theorem 4.1, we still obtain $\sqrt{D_{\|\cdot\|}}$-bounds, as Chiang *et al.* [18, Theorem 10] also obtain for D_2. This is because, unlike the analysis of Mohri and Yang [72], we retain the negative terms $-B_{r_{1:t}}(x_{t+1}, x_t)$ (essentially having the same role as the B_t terms of [18]) in the regret bound. Combined with ideas from Lemma 13 of [18], this gives the desired bounds in terms of $D_{\|\cdot\|}$, proved in Appendix 4.C:

Theorem 4.3. *Consider the conditions of Theorem 4.2, and further suppose that the losses f_t are convex and L-smooth w.r.t. a norm $\|\cdot\|$. For $t = 1, 2, \ldots, T+1$, let $\eta_t > 0$, and suppose that Assumption 4.8 holds with $\|\cdot\|_{(t)}^2 = \eta_t\|\cdot\|^2$. Further assume that $q_0 \geq 0$, $p_t, q_t \geq 0, t \geq 1$, and $\eta_t\eta_{t+1} \geq 8L^2, t = 1, 2, \ldots, T$. Then, AO-FTRL with $\tilde{g}_t = g_{t-1}$ satisfies*

$$R_T(x^*) \leq \tilde{q}_{0:T}(x^*) + p_{1:T}(x^*) + 2\sum_{t=1}^{T} \frac{1}{\eta_t} \max_{x \in \mathcal{X}} \|\nabla f_t(x) - \nabla f_{t-1}(x)\|_*^2 . \tag{4.18}$$

Letting $\eta_t = \eta = \sqrt{D_{\|\cdot\|}}$, and $\tilde{q}_0 = \eta\|\cdot\|^2, \tilde{q}_t, p_t = 0, t \geq 1$, generalizes the $O(\sqrt{D_2})$ bound of Chiang *et al.* [18] to any norm (under the same assumption they make, that

$D_{\|\cdot\|} \geq 8L^2$). In the next section, we provide an algorithm that does not need prior knowledge of $D_{\|\cdot\|}$.

4.7.4 Adaptive optimistic composite-objective learning with variational bounds

Next, we provide a simple analysis of the composite-objective version of AO-FTRL, and obtain variational bounds in terms of $D_{\|\cdot\|}$ for composite objectives with smooth f_t. We focus on Setting 2; similar results are immediate for Setting 1. Consider the update

$$x_{t+1} \in \operatorname*{arg\,min}_{x\in\mathcal{X}} \langle g_{1:t} + \tilde{g}_{t+1}, x\rangle + \psi_{1:t}(x) + p_{1:t}(x) + \tilde{q}_{0:t}(x)\,, \tag{4.19}$$

that is, the composite-objective AO-FTRL algorithm. Then we have the following corollary of Theorem 4.2.

Corollary 4.3. *Suppose that $\psi_t, t = 1, 2, \ldots, T$, satisfy the conditions of Corollary 4.2, and $\tilde{q}_0$ and $p_t, \tilde{q}_t, t \geq 1$, are non-negative. Let $q_0 = \tilde{q}_0 + \langle \tilde{g}_1, \cdot\rangle$ and $q_t = \tilde{q}_t + \psi_t + \langle \tilde{g}_{t+1} - \tilde{g}_t, \cdot\rangle$. Suppose that q_0, $p_t, q_t, t \geq 1$ satisfy Assumptions 4.1 and 4.8, and $f_t, t \geq 1$ satisfy Assumption 4.5. Then, composite-objective AO-FTRL (update* (4.19)*) satisfies*

$$R_T^{(\ell)}(x^*) \leq \tilde{q}_{0:T-1}(x^*) + p_{1:T}(x^*) + \sum_{t=1}^{T} \frac{1}{2}\|g_t - \tilde{g}_t\|_{(t),*}^2\,.$$

Proof. Starting as in Corollary 4.2, defining $\tilde{g}_0 = 0$, and noting that $0 = q_0 - \tilde{q}_0 - \langle \tilde{g}_1, \cdot\rangle$,

$$R_T^{(\ell)}(x^*) \leq R_T(x^*) + \sum_{t=0}^{T} q_t(x_{t+1}) - \tilde{q}_t(x_{t+1}) + \tilde{q}_t(x^*) - q_t(x^*) - \langle \tilde{g}_{t+1} - \tilde{g}_t, x_{t+1} - x^*\rangle\,.$$

Proceeding as in Theorem 4.2 completes the proof. □

The bounds of Mohri and Yang [72] for Setting 2 correspond to the non-proximal FTRL case. As such, one has to set the step-size sequences according to the Dual-Averaging AdaGrad recipe (c.f. Table 4.1), which requires an additional regularization of $\tilde{q}_0 = \sqrt{\delta}\|\cdot\|_2^2$. In contrast, in FTRL-Prox, $\tilde{q}_0 = 0$. This δ value makes Dual-Averaging AdaGrad non-scale-free, while FTRL-Prox is scale-free (i.e., the x_t are independent of the scaling of f_t). Our analysis avoids this problem by the early separation of the proximal (p_t) and non-proximal regularizers (q_t) in Ada-FTRL. In particular, $p_t, \tilde{q}_t$ in Corollary 4.3 can be set as $\tilde{q}_t = 0$ and $p_t = \frac{\eta_t - \eta_{t-1}}{2}\|x - x_t\|^2$ with $\eta_t = \eta\sqrt{\sum_{s=1}^{t}\|g_s - \tilde{g}_s\|_*^2}, \eta > 0$ for

$t = 1, 2, \ldots, T$. This gives composite-objective AO-FTRL-Prox, a scale-free adaptive optimistic algorithm for Setting 2.

In addition, using Theorem 4.3, we can obtain a variational bound for composite-objective optimistic FTRL-Prox (proved in Appendix 4.C), which was not available through the analysis of Mohri and Yang [72] even under Setting 1:

Corollary 4.4. *Let $\psi_t, t = 1, 2, \ldots, T$, be convex and satisfy the conditions of Corollary 4.2. Further assume that f_t are convex and L-smooth w.r.t. some norm $\|\cdot\|$. Suppose that $\mathcal{X}$ is closed, and let $R^2 = \sup_{x,y\in\mathcal{X}} \|x - y\|^2 < +\infty$ be the diameter of $\mathcal{X}$ measured in $\|\cdot\|$. Define $\eta = 2/R$. Suppose we run composite-objective AO-FTRL (update (4.19)) with the following parameters: $\tilde{q}_0 = 0$, and for $t = 1, 2, \ldots, T$, $\tilde{g}_t = g_{t-1}$, $\tilde{q}_t = 0$, and $p_t = \frac{\eta_t - \eta_{t-1}}{2}\|x - x_t\|^2$, where $\eta_0 = 0$ and $\eta_t = 4RL^2 + \eta\sqrt{\sum_{s=1}^t \|g_s - \tilde{g}_s\|_*^2}$ for $t \geq 1$. Then,*

$$R_T^{(\ell)}(x^*) \leq 2R^3L^2 + R + 2R\sqrt{2D_{\|\cdot\|}} = O\left(R\sqrt{D_{\|\cdot\|}}\right) . \tag{4.20}$$

Note that the learning rate η_t is bounded from below (by $4RL^2$), which is essential in the algorithm to achieve a combination of the best properties of Mohri and Yang [72], Chiang *et al.* [18], and Rakhlin and Sridharan [88] and Rakhlin and Sridharan [90]: First, like Mohri and Yang [72], we allow the use of composite-objectives. Second, similarly to Chiang *et al.* [18] (but unlike [72], [88], [90]) our bound applies to the variation of general convex smooth functions, and is still optimal when $L = 0$ (e.g., Corollary 2 of [90]). Third, we do not need the knowledge of $D_{\|\cdot\|}$ (required by [18]) to set the step-sizes, and avoid the regret penalty of using a doubling trick (as done by Rakhlin and Sridharan [88]). Fourth, in the practically interesting case of a composite L1 penalty ($\psi_t = \alpha_t\|\cdot\|_1$), FTRL-Prox, which is the basis of our algorithm, gives sparser solutions [69] than MD, which is the basis of the algorithms of Chiang *et al.* [18] and Rakhlin and Sridharan [90]. Fifth, when $L = 0$, the algorithm is scale-free (unlike [72] and [88]). Finally, the results apply to the variation measured in any norm. Table 4.4 provides a summary of this comparison.

4.8 Variance Reduction with Composite Objectives

In this section, we show that our methods are also directly applicable to stochastic variance reduction methods with composite objectives. We provide new variance-reduced algorithms for stochastic optimization, based on Ada-FTRL and Ada-MD. In contrast,

	Algorithm	$\psi_t \neq 0$	Non-linear variations	Needs doubling trick or knowledge of $D_{\|\cdot\|}$	Scale-free
Chiang *et al.* [18]	MD	No	Yes	Uses a bound on $D_{\|\cdot\|}$	Yes
Rakhlin and Sridharan [88]	FTRL	No	No	Uses doubling trick	No
Rakhlin and Sridharan [90]	MD	No	No	Not needed	No
Mohri and Yang [72]	FTRL	Yes	No	Not needed	No
Kamalaruban [50]	MD	Yes	No	Not needed	Yes
Corollary 4.4	**FTRL**	**Yes**	**Yes**	**Not needed**	**Yes**

Table 4.4: Comparison of the adaptive, optimistic and variational bounds obtained in this paper vs previous work. "Non-linear variations" refers to whether the bounds are given in terms of gradient variations only (linear) or in terms of $D_{\|\cdot\|}$.

to our knowledge, the algorithms in the literature have been limited to different variants of (typically unconstrained) MD. In Section 4.2.1, however, we separated the effect of the algorithm (Ada-FTRL vs. Ada-MD) from the effect of the gradient estimation technique (variance-reduced estimates vs. the standard stochastic gradient). Hence, our results directly provide Ada-MD and Ada-FTRL versions of SVRG and its variants, also allowing constrained optimization (with projections). This is important in practice because, as mentioned in Section 4.7.4, FTRL with a composite L1 penalty results in sparser models than MD. Finally we present an error bound for our methods that covers stochastic variance-reduced gradient methods such as SVRG [43] and SVRG++ [4].

Setting: We focus on the following composite-objective optimization problem:

$$\text{find } x^* = \arg\min_{x \in \mathcal{X}} \ell(x) := f(x) + \phi(x)\,, \tag{4.21}$$

where $\phi : \mathcal{X} \to \mathbb{R}$ is a convex function, $f = \mathbb{E}_{\xi \sim \mathbb{P}_\Xi}[F(\cdot, \xi)]$, and for all $\xi \in \Xi$, $F(\cdot, \xi) : \mathcal{H} \to \mathbb{R}$ is convex, differentiable and L-smooth w.r.t. the Hilbert-space norm, which we denote by $\|\cdot\|$. In addition, we assume that we have access to two types of gradient oracles: an exact oracle which returns $f'(x)$ at any point $x \in \mathcal{X}$ and a stochastic oracle which, given a random element $\xi \sim \mathbb{P}_\Xi$ independent of x, returns $F'(x, \xi)$ (where the derivative is taken w.r.t. the first variable, x). An important special case, typically the focus of previous work, is when $f = \frac{1}{n}\sum_{i=1}^n f_i$, where each $f_i(n)$ is differentiable and the exact and

```
Algorithm 3: Generic SVRG
   Input: BASE: An instance of ADA-FTRL or ADA-MD; $m_1, m_2, \dots$: Epoch
          lengths
 1 $x_1 \leftarrow$ the first iterate of BASE
 2 $\tilde{x} \leftarrow x_1$.
 3 for $s \leftarrow 1, 2, \dots$ do
 4     for $t \leftarrow m_{1:s-1} + 1 \dots m_{1:s}$ do
 5         Sample $\xi_t \sim \mathbb{P}_\Xi$ independently of $x_t, \tilde{x}$
 6         $g_t \leftarrow F'(x_t, \xi_t) - F'(\tilde{x}, \xi_t) + f'(\tilde{x})$
 7         Feed BASE with the gradient estimate $g_t$
 8         $x_{t+1} \leftarrow$ the next iterate of BASE
 9     end for
10     Option I: $\tilde{x} \leftarrow \sum_{j=1}^{m_s} x_{m_{1:s-1}+j}$
11     Option II: Select $\tilde{x}$ from $x_{m_{1:s-1}+j}, j = 1, 2, \dots, m_s$, with equal probability
12 end for
```

stochastic oracles work, respectively, by calculating the gradient at x on all f_i's or on a randomly selected f_i.

In Algorithm 3, we provide an SVRG meta-algorithm, which receives an instance of ADA-FTRL and ADA-MD as the base algorithm, and feeds it with variance-reduced gradient estimates. That is, we run ADA-FTRL and ADA-MD, but feed them with the g_t given by Algorithm 3. Then, we can use the results of Section 4.4 to obtain a risk bound for Algorithm 3.

In particular, suppose that the regularizers used by BASE satisfy Assumption 4.8' with the norms $\|\cdot\|_{(t)} = \sqrt{c}\|\cdot\|$ for some $c > 0$ (recall that $\|\cdot\|$ is the Hilbert-space norm). Given the setting of this section, Assumption 4.6 also holds with the norm $\sqrt{L}\|\cdot\|$. Finally, by construction and the tower rule for expectation, $\mathbb{E}\{g_t|x_t\} = f'(x_t)$. Thus, we are in the "Stochastic Optimization" setting with smooth, convex objectives, and Corollary 4.7 applies. However, thanks to the special construction of g_t, we can further bound the $\|\sigma_t\|_*^2$ terms in the corollary, where $\sigma_t = g_t - f'(x_t)$ is the error of the gradient estimate. This is shown by the next lemma, which is standard in the variance-reduction literature (see, e.g, Lemma A.2 of [4]).

Lemma 4.2. *Consider Algorithm 3 in the optimization setting above. Let x^* be defined as in* (4.21)*, and let $\tilde{x}_s$ denote the value of $\tilde{x}$ used in computing g_t. Then,*

$$\mathbb{E}\big\{\|\sigma_t\|^2\big\} \leq 4L\ \mathbb{E}\{\ell(x_t) - \ell(x^*) + \ell(\tilde{x}_s) - \ell(x^*)\}\,.$$

Using this lemma, we can prove the following performance guarantee for Algorithm 3 (the proof is relegated to Appendix 4.D).

Theorem 4.4 (Composite-objective variance reduction)**.** *Let Algorithm 3 be used with epoch lengths $m_1, \dots, m_S$, and let $b > 0$ be such that $m_{s+1} \le b m_s$ for all $s = 1, 2, \dots, S$. Let $c > 2b + 2$ be arbitrary, and suppose that* Base *is either* Ada-FTRL *with regularizers $q_0 = (1+c)\frac{L}{2}\|\cdot\|^2$ and $q_t = \phi$, $p_t = 0$ for $t \ge 1$, or* Ada-MD *with regularizers $q_0 = 0$, $p_1 = \frac{(1+c)L}{2}\|\cdot\|^2$, and $q_t = \phi, p_{t+1} = 0$ for $t \ge 1$. Furthermore, let R be an upper bound on $\frac{1}{2}\|x_1 - x^*\|^2$ for* Ada-MD *and on $\frac{1}{2}\|x^*\|^2$ for* Ada-FTRL*. Then, after $T = m_{1:S}$ iterations, the estimate $\bar{x}_T = \frac{1}{T}\sum_{t=2}^{T+1} x_t$ satisfies*

$$\mathbb{E}\{\ell(\bar{x}_T) - \ell(x^*)\} \le \frac{2(1 + b + m_1)\left(\ell(x_1) - \ell(x^*)\right) + c(c+1)LR}{T(c - 2b - 2)} .$$

Furthermore, letting $c = 2b + 2$, after S epochs, the estimate $\tilde{x}'_S = \frac{1}{m_S}\sum_{j=2}^{m_S+1} x_{m_{1:s}+j}$ satisfies

$$\mathbb{E}\{\ell(\tilde{x}'_S) - \ell(x^*)\} \le \frac{(1 + b + m_1)\left(\ell(x_1) - \ell(x^*)\right) + (2b^2 + 5b + 3)LR}{b\, m_S} .$$

Remark 4.6. *This theorem recovers the convergence rates for vanilla SVRG (first part, with $m_s = m$), as well as SVRG++ (second part, with $b = 2$).*

Remark 4.7. *Instead of Lemma 4.2 for convex functions, one could use Lemma 5.2 of Allen-Zhu and Yuan [4] to extend our results to strongly-convex sums of non-convex smooth functions, and also to extend the results of Allen-Zhu and Yuan [4] for this family of problems to FTRL(-Proximal) algorithms. The details are left for future work.*

4.9 Application to non-convex optimization

In this section we examine how far our approach can lead in solving non-convex optimization problem, and we identify some interesting classes of non-convex loss functions for which we can obtain performance guarantees without any extra effort.

Central to this extension is the decomposition of assumptions in our analysis: we are not using the convexity of f_t in Lemma 4.1 or Theorem 4.1, but only at the very last stage of the analysis, where convexity can ensure that Assumption 4.5 holds. Thus, the analysis easily extends to non-convex optimization problems where Assumption 4.5 either holds or could be replaced by another technique at the final stage of the analysis. In the rest of this section, we explore such classes of non-convex problems, which are also related to the Polyak-Łojasiewicz (PL) condition used in the non-convex optimization and learning community. For background and a summary of related work, consult Karimi *et al.* [51].

4.9.1 Stochastic optimization of star-convex functions

First, we explore the class of non-convex functions for which Assumption 4.5 directly holds. As it turns out, this is a much larger class of functions than convex functions. In particular, consider the so-called "star-convex" functions [79]:[17]

Definition 4.2 (Star-convex function)**.** *A function f is* star-convex *at a point x^* if and only if x^* is a global minimizer of f, and for all $\alpha \in [0,1]$ and all $x \in \operatorname{dom}(f)$:*

$$f(\alpha x^* + (1-\alpha)x) \le \alpha f(x^*) + (1-\alpha) f(x)\,. \tag{4.22}$$

A function is said to be star-convex when it is star-convex at some of its global minimizers.

The name "star-convex" comes from the fact that the sub-level sets $L_\beta = \{x \,:\, f(x) \le \beta\}$ of a function f that is star-convex at x^* are star-shaped with center x^* (recall that a set U is star-convex with center x if for any $y \in U$, the segment between x and y is included in U). However, note that there are functions whose sub-level sets are star-convex that are themselves not star-convex. In particular, functions that are increasing along the rays (IAR) started from their global minima have star-shaped sub-level sets and vice versa, but some of these functions (e.g., $f(x) = \sqrt{|x|}$, $x \in \mathbb{R}$) is clearly not star-convex. Recall that quasi-convex functions are those whose sub-level sets are convex. In one dimension a star-convex function is thus also necessarily quasi-convex. However, clearly, there are star-convex functions (such as $x \mapsto |x|\mathbb{I}\{|x| \le 1\} + 2|x|\mathbb{I}\{|x| > 1\}$, $x \in \mathbb{R}$) that are not convex and in multiple dimensions there are star-convex functions that are not quasi-convex (e.g., $x \mapsto \|x\|^2 g(\frac{x}{\|x\|_2^2})$ where $g(u)$ is, say, the sine of the angle of u with the unit vector e_1).

Star-convex functions arise in various optimization settings, often related to sums of squares [58], [79]. It is easy to see from the definitions that the set of star-convex functions

[17]We modify the definition so that it is relative to a given fixed global minimizer as this way we capture a larger class of functions and this is all we need.

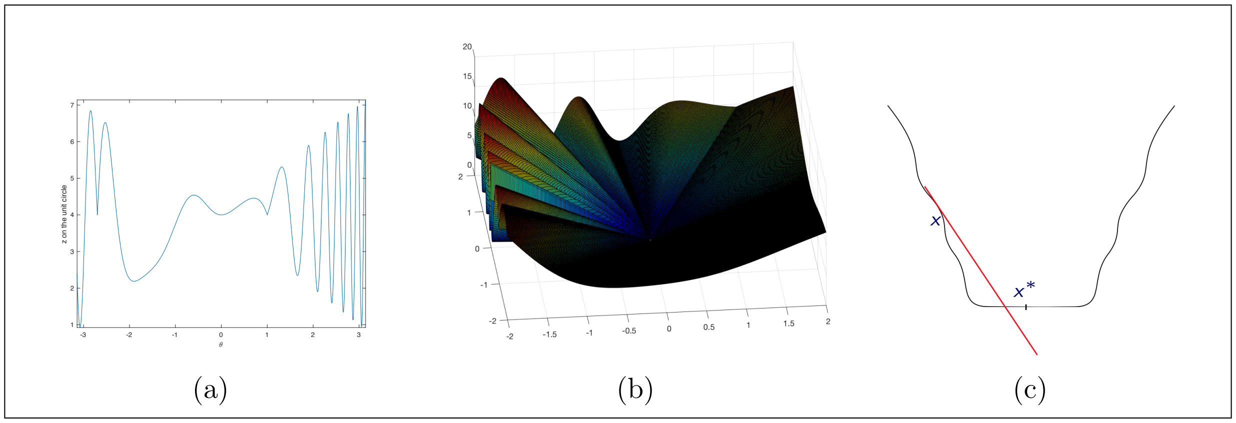


Figure 4.3: Star-convex functions: (a) z defined on the unit circle and (b) its star-convex extension to two dimensions. (c) The linear approximation of a star-convex function fitted at any point is a lower bound at the minimum.

is closed under non-singular affine domain transformations, addition (of functions having the same center) and multiplication by non-negative constants. Further, for $x \in \mathbb{R}^d$, $x \mapsto \prod_i |x_i|^{p_i}$ is star-convex (at zero) whenever $\sum_i p_i \geq 1$. It is also easy to see that for any positive function z defined on the unit sphere, $\|x\| z(x/\|x\|)$ is star-convex at 0 (see Figure 4.3a,b). For further properties and examples see Lee and Valiant [58].

We can immediately see that Assumption 4.5 holds for star-convex functions:

Lemma 4.3 (Non-negative Bregman divergence for star-convex functions)**.** *Let f be a directionally differentiable function with global optimum x^*. Then, f is star-convex at x^* if and only if for all $x \in \mathcal{H}$,*

$$\mathcal{B}_f(x^*, x) \geq 0\,.$$

The lemma essentially states that the linear approximation of a star-convex function f at any point x is a lower bound of the function at x^*. This is illustrated in Figure 4.3c.

Proof. Both directions are routine. For illustration we provide the proof of the forward direction. Assume without loss of generality that $x^* = 0$ and $f(x^*) = 0$. Then star-convexity at x^* is equivalent to having $f(\alpha x) \leq \alpha f(x)$ for any x and $\alpha \in [0, 1]$. Further, $\mathcal{B}_f(x^*, x) \geq 0$ is equivalent to $-f(x) - f'(x; -x) \geq 0$. Now, $f'(x; -x) = \lim_{\alpha \downarrow 0} \frac{f(x+\alpha(-x))-f(x)}{\alpha}$. Under star-convexity, $f(x + \alpha(-x)) = f((1-\alpha)x) \leq (1-\alpha) f(x)$. Hence, $f'(x; -x) \leq \lim_{\alpha \downarrow 0} \frac{(1-\alpha)f(x)-f(x)}{\alpha} = -f(x)$. □

Thus, Corollaries 4.6 and 4.7 apply to star-convex functions. In other words:

- For stochastic optimization of directionally-differentiable star-convex functions in Hilbert spaces, Ada-FTRL and Ada-MD converge to the global optimum *with the same rate as they converge for convex functions* (including fast rates due to other assumptions, e.g., smoothness).

Of course, a similar result holds for the online setting, too, but in this case the assumption that each f_t is star-convex w.r.t. the same center x^* becomes restrictive.

Remark 4.8. *Since the rate of regret depends on the norm of the gradients g_t, to get fast rates one needs to control these norms. This is trivial if f is Lipschitz-continuous. However, some star-convex functions are not Lipschitz, even arbitrarily close to the optima (e.g., $f(x,y) = (\sqrt{|x|} + \sqrt{|y|})^2$). For such functions, Lee and Valiant [58] propose alternative methods to gradient descent. However, it seems possible to control the norms in these settings using additional regularization (as in the normalized gradient descent method); see, e.g., the work of Hazan* et al. *[41], and the recent work of Levy [59]. Exploring this idea is left for future work.*

4.9.2 Beyond star-convex functions

Inspecting our proofs we may notice that Assumption 4.5 is unnecessarily restrictive: to maintain the same rate of growth for regret, it suffices for the sum of Bregman divergences to grow with the same rate as the rest of the bound, rather than being negative and hence dropped. This extends all of our results to another interesting class of non-convex functions which generalize star-convexity:

Definition 4.3 (τ-star-convexity, Hardt *et al.* [37])**.** *Let f be a directionally differentiable function f with global optimum x^*. Then f is* τ-star-convex[18] *on a set $\mathcal{X}$ at $x^* \in \mathcal{H}$ if there is $\tau > 0$ such that for all $x \in \mathcal{X} \cap \mathrm{dom}(f)$,*

$$\tau(f(x) - f(x^*)) \leq -f'(x; x^* - x)\,. \tag{4.23}$$

Note that by Lemma 4.3, star-convexity corresponds to the case when $\tau = 1$. Hardt *et al.* [37] demonstrated that an objective function that arises naturally in the identification of

[18] Hardt *et al.* [37] define the same concept under τ-weakly-quasi-convexity. However, per our previous discussion, it appears more appropriate to call this property τ-star-convexity. Especially since when $\tau = 1$ we get back star-convexity, which, as we have seen is not a weakening of quasi-convexity.

certain class of linear systems is τ-star-convex with some $\tau > 0$. For differentiable functions, (4.23) is equivalent to $f(x) - f(x^*) \le \frac{1}{\tau}\langle \nabla f(x), x - x^* \rangle$, so it is a simple generalization of the linear upper bound one typically uses to reduce online convex optimization to online linear optimization. Therefore, any regret bound that is proved via upper bounding linearized losses automatically extends to τ-star-convex functions. However, in general, it may require substantial work to identify what assumptions are used exactly in proving an upper bound on the linearized loss (e.g., [37] reproved the convergence guarantees for smooth SGD). The next lemma shows that our techniques can automatically separate the effects of different assumptions and provide fast regret rates under appropriate circumstances.

Lemma 4.4 (Basic regret bound under τ-star-convexity)**.** *Let f be locally directionally differentiable and τ-star-convex on a set $\mathcal{X}$ at x^*, $f_1 \cdots = f_T = f$. Then, for all $x_t \in \mathcal{X} \cap \operatorname{dom}(f_t)$ and $g_t \in \mathcal{H}$ $(t = 1, 2, \ldots, T)$,*

$$R_T(x^*) \le \frac{1}{\tau}\left(R_T^+(x^*) + \sum_{t=1}^{T} \langle g_t, x_t - x_{t+1} \rangle + \delta_t \right) .$$

Proof. The proof can be derived from the right-hand side of (4.3), but a shorter direct proof is also available: Add and subtract $\langle g_t, x^* - x_t \rangle$ to the right-hand side of (4.23). Noticing that $-f'(x_t; x^* - x_t) + \langle g_t, x^* - x_t \rangle = \delta_t$, summing up and using the definition $R_T^+(x^*) = \sum_t \langle g_t, x_{t+1} - x^* \rangle$ gives the result. □

Now since the regret was bounded through the expression in the parentheses of the previous display, Corollaries 4.6 and 4.7 apply. In other words:

- For stochastic optimization of directionally-differentiable τ-star-convex functions in Hilbert spaces, Ada-FTRL and Ada-MD enjoy $1/\tau$*-times the same regret as when they are applied to linearized loss functions*, including fast rates due to other assumptions, e.g., smoothness.

In the convex case the strong convexity of the losses (Assumption 4.7) implied that their Bregman divergences are non-negative (Assumption 4.5). The natural generalization of this leads to the following definition:

Definition 4.4 (τ-star-strong-convexity)**.** *Let f, r be directionally differentiable and let x^* be a global minimum of f. Then, f is τ-star-strongly-convex w.r.t. r if $S :=$*

$\mathrm{dom}(f) \cap \mathrm{dom}(r)$ *is non-empty and there exists* $\tau > 0$ *such that for all* $x \in S$ *and some minimizer* x^* *of* f,

$$\tau(f(x) - f(x^*)) \leq -f'(x; x^* - x) - \mathcal{B}_r(x^*, x)\,. \tag{4.24}$$

Replacing τ-star-convexity with τ-star-strong-convexity gives the following analogue of Lemma 4.4:

Lemma 4.5 (Basic regret bound under τ-star-strong-convexity)**.** *Let* f, r *be locally directionally differentiable. Assume that* f *is* τ*-star-strongly-convex w.r.t.* r *at* x^* *on a set* $\mathcal{X}$. *Then, for all* $x_t \in \mathcal{X} \cap \mathrm{dom}(f_t) \cap \mathrm{dom}(r)$ *and* $g_t \in \mathcal{H}$ *(*$t = 1, 2, \ldots, T$*),*

$$R_T(x^*) \leq \frac{1}{\tau}\left(R_T^+(x^*) + \sum_{t=1}^{T} \langle g_t, x_t - x_{t+1}\rangle + \delta_t - \mathcal{B}_r(x^*, x_t)\right)\,.$$

Proof. The proof follows the same step as that of Lemma 4.4, except that we need to use (4.24) instead of (4.23). □

It follows that the same manipulations as in Corollaries 4.6 and 4.7 imply:

- For stochastic optimization of directionally-differentiable τ-star-strongly-convex functions in Hilbert spaces, Ada-FTRL and Ada-MD converge to the global optimum *with* $1/\tau$*-times the same rate as they converge for strongly convex functions.*

It appears that τ-star-strong-convexity is related to the Polyak-Łojasiewicz (PL) inequality. Recall that a differentiable function f satisfies the PL inequality with constant $\mu > 0$ if

$$\mu(f(x) - f(x^*)) \leq \frac{1}{2}\|\nabla f(x)\|_2^2\,,$$

where x^* is a global minimizer of f. Proposed independently and simultaneously by [87] and [64], the PL inequality appears to play a fundamental role in the study of incremental gradient algorithms (see [51] and the references therein). As star-convexity, the PL inequality can also be satisfied by non-convex functions, partly explaining the prominent role it plays in the analysis of gradient methods. We can see that τ-star-strong-convexity implies the PL inequality when r is the squared Euclidean norm:

Lemma 4.6 (PL is implied by star-strong-convexity). *Let $r(x) = \frac{1}{2}\|x\|_2^2$ and let f be differentiable. If f is τ-star-strongly-convex w.r.t. r, then f also satisfies the PL inequality with $\mu = \tau$.*

Proof. Assume that f satisfies (4.24). We have

$$-f'(x; x - x^*) = \langle \nabla f(x), x^* - x \rangle \le \frac{1}{2}\left(\|\nabla f(x)\|^2 + \|x^* - x\|^2\right) ,$$

where the second step follows from the Fenchel-Young inequality. As it is well known, $B_r(x, y) = \frac{1}{2}\|x - y\|^2$. Thus, (4.24) implies that $\tau(f(x) - f(x^*)) \le \frac{1}{2}\|\nabla f(x)\|_2^2$. $\square$

Finally, note that the results can be combined with other algorithmic ideas, such as implicit-update and non-linearized learning; the same extensions of Corollaries 4.6 and 4.7, as discussed in Section 4.6, apply here as well. We can also do the same manipulations to obtain results similar to Section 4.5 for composite-objective non-convex learning. Note, however, that in this case, the star-convexity assumptions must hold with x^* being the minimizer of ℓ, not f, which makes them more restrictive than the non-composite case. We leave it for future work to handle composite-objective learning of star-convex functions in full generality.

We end this section by noting that there are interesting classes of non-convex problems other than the PL class; see, e.g., Karimi *et al.* [51]. A direction for future work is to explore whether these classes relate to specific conditions on Bregman divergences, and whether similar convergence results for general adaptive optimization are also possible under these function classes.

4.10 Discussion

In this section, we compare the results obtained in this paper to the previous attempts at unified analysis of adaptive FTRL and MD. A starting point of our work was the unifying treatment of online learning algorithms by McMahan [69], as well as the generalized adaptive FTRL analysis of Orabona *et al.* [82].

4.10.1 Comparison to the analysis of McMahan [69]

McMahan [69] also studied a unified, modular analysis of MD and FTRL algorithms (albeit with different modules), assuming that the regularizers p_t, q_t, r_t are convex, non-negative, and satisfy Assumption 4.8. Ada-FTRL and Ada-MD encompass all of the algorithms they considered. In particular, their Theorems 1 and 2 are special cases of Corollary 4.5 (recall that non-linearized FTRL, and in particular strongly-convex FTRL, are also special cases of Ada-FTRL; see Section 4.6). In addition, our analysis applies more generally to infinite-dimensional Hilbert spaces, our presentation of Ada-FTRL encompasses a larger set of algorithms, the relaxed assumptions under which we analyzed Ada-FTRL and Ada-MD remove certain practical limitations that existed in the work of McMahan [69], and our analysis captures a wider range of learning settings. We discuss these improvements below.

Importantly, McMahan [69] also provides a reduction from MD to a version of FTRL-Prox. This, in particular, illuminates important differences between MD and FTRL in composite-objective learning. We refer the reader to Section 6 of their paper. We decided to keep the presentation of the two algorithms separate to facilitate the relaxation of the assumptions on the regularizers; see Assumptions 4.1 and 4.2 and the discussion below.

Relaxing the assumptions on the regularizers

A central part of the modularity of our analysis comes from the flexibility of Assumptions 4.1 and 4.2 on the regularizers of Ada-FTRL and Ada-MD. In particular, unlike McMahan [69], we do not assume that the individual regularizers p_t, q_t, r_t are non-negative or convex.

This relaxation provides two benefits. First, with the non-negativity restriction removed, we can add arbitrary, possibly linear, components to the regularizers. As we showed above, this resulted in a simple recovery and analysis of optimistic FTRL and a new class of optimistic MD algorithms (Section 4.7), as well as a straightforward recovery of implicit and non-linearized updates, even for non-convex functions (Section 4.6).

Second, with the convexity assumption removed, Ada-FTRL and Ada-MD can accommodate algorithmic ideas such as non-decreasing regularization. For example, AdaDelay [101], an instance of Ada-MD for distributed delayed stochastic optimization, uses $r_{1:t} = \eta_t \| \cdot \|^2$, but η_t is not guaranteed to be non-decreasing, that is, r_t could be negative and non-convex (while $r_{0:t}$ still remains convex for all t). Now, note that MD and FTRL-Prox are closely related. Particularly, if the p_t are themselves Bregman divergences (as in proximal AdaGrad), then FTRL-Prox and MD have identical regret bounds. Therefore, the techniques of Sra *et al.* [101] for controlling the regularizer terms in the bound could be naturally applied, almost with no modification, to an FTRL-Prox version of AdaDelay. This extension to FTRL-Prox is interesting since, as mentioned before, composite FTRL-Prox with an L1 penalty tends to produce sparser results compared to Ada-MD ([68], [69], Section 6.2). Thus, while this variant of FTRL-Prox is a special case of Ada-FTRL (e.g., Corollary 4.5 applies), it was not clear how to analyze this algorithm under the assumptions made by McMahan [69]. We leave the detailed study of this extension for future work.

Finally, the choice to separate the proximal and non-proximal regularizers in Ada-FTRL provides certain conveniences. In particular, the q_t terms can take the role of incorporating information (such as composite terms) into Ada-FTRL, while the proximal part p_t remains intact. This precludes the need to provide a separate analysis for FTRL-Prox every time the structure of information changes (e.g., when implicit updates are added). Thus, unlike Section 5 of McMahan [69], we did not need to provide a separate analysis (their Theorem 10) for composite-objective FTRL-Prox. We also avoided the complications with composite optimistic FTRL-Prox as in Mohri and Yang [72]; see Section 4.7.

The regret decomposition and analysis of new learning settings

In comparison to McMahan [69], the analysis we provided exhibits much flexibility across learning settings. In particular, the regret decomposition given by Lemma 4.1 enabled us to accommodate a wide range of learning settings, and separate the effect of the learning setting from the forward regret of the algorithm. Building on this, for example, we provided a clean analysis of variational and variance-dependent bounds for smooth losses (and generalized them to adaptive algorithms). In addition, by encapsulating the effect of loss convexity into Assumption 4.5, we could generalize the analysis to certain non-convex classes.

4.10.2 Comparison to the analysis of Orabona *et al.* [82]

Orabona *et al.* [82] study a special case of Ada-FTRL where Assumption 4.8 holds. The main result of Orabona *et al.* [82], that is, their Lemma 1, can be thought of as playing the same role as (4.25). We emphasize, however, that their Lemma 1 is a quite general result. For example, with a few algebraic operations we could recover a special case of

Theorem 4.1 (including also the case of $p_t \neq 0$) from their Lemma 1, by setting $z_t = 0$ and moving the linear components to their f_t functions. Nevertheless, our analysis extends the work of Orabona *et al.* [82] to infinite-dimensional Hilbert-spaces and, more importantly, to Ada-MD. Furthermore, we demonstrated a principled way of mixing algorithmic ideas and incorporating information from the learning setting into FTRL and MD using the q_t functions, which in turn led to the discovery of the single-projection optimistic MD family of algorithms. Finally, the comments of Section 4.10.1 apply.

Importantly, the authors also provide a compact analysis of the Vovk-Azoury-Warmuth algorithm, as well as online binary classification algorithms. These results are essentially obtained from combining their Lemma 1 with interesting regret decompositions other than the one we presented in Lemma 4.1. It seems possible to combine their regret decompositions with our analysis to extend their result to Ada-MD algorithms, and to obtain refined bounds for smooth losses. We leave this direction for future work.

4.11 Conclusion and future work

We provided a generalized, unified and modular framework for analyzing online and stochastic optimization algorithms, and demonstrated its flexibility on several existing, as well as new, algorithms and learning settings. Our framework can be used together with other algorithmic ideas and learning settings, for example, adaptive delayed-feedback algorithms like AdaDelay [101], but exploring these problems are out of the scope of this work. There are many interesting questions related to non-convex optimization; while we showed that our results extend to the so-called τ-star(-strongly)-convex functions, which have already found some applications, it remains to be seen whether they also extend to other settings, such as optimization of quasi-convex functions, or functions that satisfy the Polyak-Łojasiewicz inequality. Exploring these and other applications of this framework is left for future work.

Appendices

4.A Formal statements and proofs for the standard results described in Table 4.2

Putting Lemma 4.1 and Theorem 4.1 together, for Ada-FTRL we obtain

$$\begin{aligned} R_T(x^*) \le\ & -\sum_{t=1}^T \mathcal{B}_{f_t}(x^*, x_t) + \sum_{t=0}^T (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^T (p_t(x^*) - p_t(x_t)) \\ & - \sum_{t=1}^T \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) + \sum_{t=1}^T \langle g_t, x_t - x_{t+1} \rangle + \sum_{t=1}^T \delta_t \,, \end{aligned} \tag{4.25}$$

whereas for Ada-MD,

$$\begin{aligned} R_T(x^*) \le\ & -\sum_{t=1}^T \mathcal{B}_{f_t}(x^*, x_t) + \sum_{t=0}^T (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^T \mathcal{B}_{p_t}(x^*, x_t) \\ & - \sum_{t=1}^T \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) + \sum_{t=1}^T \langle g_t, x_t - x_{t+1} \rangle + \sum_{t=1}^T \delta_t \,. \end{aligned} \tag{4.26}$$

Next we prove the concrete regret bounds, given in Table 4.2, based on the above. A schematic view of the proof ideas is given in Figure 4.2.

Corollary 4.5. *Consider the "Online Optimization" setting (Assumption 4.3), using* Ada-FTRL *(under Assumption 4.1) or* Ada-MD *(under Assumption 4.2). Suppose that Assumptions 4.5 and 4.8 hold. Then,*

(i) the regret of Ada-MD *is bounded as*

$$R_T(x^*) \le \sum_{t=0}^T (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^T \mathcal{B}_{p_t}(x^*, x_t) + \sum_{t=1}^T \frac{1}{2} \|g_t\|_{(t),*}^2 \,;$$

(ii) the regret of Ada-FTRL *is bounded as*

$$R_T(x^*) \le \sum_{t=0}^T (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^T (p_t(x^*) - p_t(x_t)) + \sum_{t=1}^T \frac{1}{2} \|g_t\|_{(t),*}^2 \,;$$

(iii) under Assumption 4.7, the regret of Ada-MD *is bounded as*

$$R_T(x^*) \le \sum_{t=0}^{T} (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^{T} \frac{1}{2}\|g_t\|^2_{(t),*}\,.$$

Proof. Note that by Assumption 4.3, we have

$$\delta_t \le 0\,, \tag{4.27}$$

for all $t = 1, 2, \dots, T$. In addition, by the Fenchel-Young inequality and Assumption 4.8,

$$\begin{aligned} \langle g_t, x_t - x_{t+1}\rangle &\le \frac{1}{2}\|x_t - x_{t+1}\|^2_{(t)} + \frac{1}{2}\|g_t\|^2_{(t),*} \\ &\le \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) + \frac{1}{2}\|g_t\|^2_{(t),*}\,. \end{aligned} \tag{4.28}$$

Putting (4.27), (4.28), and Assumption 4.5 into (4.25) and (4.26) and cancelling out the matching terms proves ((i)) and ((ii)). Finally, to prove ((iii)), we use Assumption 4.7 to cancel the $\mathcal{B}_{f_t}(x^*, x_t)$ terms with the $\mathcal{B}_{p_t}(x^*, x_t)$ terms (rather than dropping them by Assumption 4.5). □

Corollary 4.6. *Consider the "Stochastic Optimization" setting (Assumption 4.4), using* Ada-FTRL *(under Assumption 4.1) or* Ada-MD *(under Assumption 4.2). Suppose that Assumptions 4.5 and 4.8 hold. Then,*

(i) the regret of Ada-MD *is bounded as*

$$\mathbb{E}\{R_T(x^*)\} \le \mathbb{E}\left\{\sum_{t=0}^{T} (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^{T} \mathcal{B}_{p_t}(x^*, x_t) + \sum_{t=1}^{T} \frac{1}{2}\|g_t\|^2_{(t),*}\right\};$$

(ii) the regret of Ada-FTRL *is bounded as*

$$\mathbb{E}\{R_T(x^*)\} \le \mathbb{E}\left\{\sum_{t=0}^{T} (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T} \frac{1}{2}\|g_t\|^2_{(t),*}\right\};$$

(iii) under Assumption 4.7, the regret of Ada-MD *is bounded as*

$$\mathbb{E}\{R_T(x^*)\} \le \mathbb{E}\left\{\sum_{t=0}^{T} (q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^{T} \frac{1}{2}\|g_t\|^2_{(t),*}\right\}.$$

Proof. Let $f_t = f$ in Lemma 4.1 (hence in (4.25) and (4.26)), and note that by Assumption 4.4, we have

$$\mathbb{E}\{\delta_t\} = \mathbb{E}\{f'(x_t; x^* - x_t) - \langle \mathbb{E}\{g_t|x_t\}, x_t - x^*\rangle\} \le 0\,, \tag{4.29}$$

for all $t = 1, 2, \ldots, T$. Similar to the proof of Corollary 4.5, putting (4.29), (4.28), and Assumption 4.5 into (4.25) and (4.26) proves ((i)) and ((ii)). Finally, to prove ((iii)), one can use Assumption 4.7 to cancel the $\mathcal{B}_f(x^*, x_t)$ terms with the $\mathcal{B}_{p_t}(x^*, x_t)$ terms (rather than dropping them by Assumption 4.5). $\square$

Corollary 4.7. *Consider the "Stochastic Optimization" setting (Assumption 4.4), using* Ada-FTRL *(under Assumption 4.1) or* Ada-MD *(under Assumption 4.2). Suppose that Assumptions 4.5, 4.6 hold, and Assumption 4.8 holds with $r_{1:t} - \|\cdot\|^2/2$ in place of $r_{1:t}$.*[19] *Let $f^* := \inf_{x\in\mathcal{X}} f(x)$, and define $D := f(x_1) - f^*$ and $\sigma_t := g_t - \nabla f(x_t)$. Let $D_t := q_t(x^*) - q_t(x_{t+1})$. Then,*

(i) the regret of Ada-MD *is bounded as*

$$\mathbb{E}\{R_T(x^*)\} \le \mathbb{E}\left\{D_{0:T} + \sum_{t=1}^{T} \mathcal{B}_{p_t}(x^*, x_t) + \sum_{t=1}^{T} \frac{1}{2}\|\sigma_t\|^2_{(t),*} + D\right\};$$

(ii) the regret of Ada-FTRL *is bounded as*

$$\mathbb{E}\{R_T(x^*)\} \le \mathbb{E}\left\{D_{0:T} + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T} \frac{1}{2}\|\sigma_t\|^2_{(t),*} + D\right\};$$

(iii) under Assumption 4.7, the regret of Ada-MD *is bounded as*

$$\mathbb{E}\{R_T(x^*)\} \le \mathbb{E}\left\{\frac{1}{2}\|x^* - x_1\|^2 + D_{0:T} + \sum_{t=1}^{T} \frac{1}{2}\|\sigma_t\|^2_{(t),*} + D\right\}.$$

Proof. Note that for all $t = 1, 2, \ldots, T$, by Assumption 4.6 and the Fenchel-Young inequality,

$$\begin{aligned}\langle g_t, x_t - x_{t+1}\rangle &= f(x_t) - f(x_{t+1}) + \mathcal{B}_f(x_{t+1}, x_t) + \langle g_t - \nabla f(x_t), x_t - x_{t+1}\rangle \\ &\le f(x_t) - f(x_{t+1}) + \frac{1}{2}\|x_t - x_{t+1}\|^2 + \langle \sigma_t, x_t - x_{t+1}\rangle\end{aligned}$$

[19]The modification to Assumption 4.8 is equivalent to adding an extra $\|x\|^2/2$ regularizer to Ada-FTRL and Ada-MD.

$$\leq f(x_t) - f(x_{t+1}) + \frac{1}{2}\|x_t - x_{t+1}\|^2 + \frac{1}{2}\|\sigma_t\|^2_{(t),*} + \frac{1}{2}\|x_t - x_{t+1}\|^2_{(t)} \quad (4.30)$$

Putting (4.29), (4.30), and Assumption 4.5 into (4.25) and (4.26), telescoping the f terms, using $f(x_{T+1}) \geq f^*$, and canceling out the matching terms gives ((i)) and ((ii)). Finally, to prove ((iii)), one can use Assumption 4.7 to cancel the $\mathcal{B}_f(x^*, x_t)$ terms with the $\mathcal{B}_{p_t}(x^*, x_t)$ terms (rather than dropping them by Assumption 4.5). □

4.B Proofs for Section 4.5

Proof of Corollary 4.2. Define $\psi_0 := \psi_1$. Then, using our assumptions on ψ_t, we have

$$\begin{aligned}
R_T^{(\ell)}(x^*) &= R_T(x^*) + \sum_{t=1}^{T}(\psi_t(x_t) - \psi_t(x^*)) \\
&= R_T(x^*) + \sum_{t=1}^{T}(\psi_t(x_{t+1}) - \psi_t(x^*)) + \sum_{t=1}^{T}(\psi_t(x_t) - \psi_{t-1}(x_t)) \\
&\quad + \psi_1(x_1) - \psi_T(x_{T+1}) \\
&\leq R_T(x^*) + \sum_{t=1}^{T}(\psi_t(x_{t+1}) - \psi_t(x^*)) \\
&= R_T(x^*) + \sum_{t=1}^{T}\Big(q_t(x_{t+1}) - \tilde{q}_t(x_{t+1}) + \tilde{q}_t(x^*) - q_t(x^*)\Big).
\end{aligned}$$

The rest of the proof is as in Corollary 4.1, noting that $\tilde{q}_0 = q_0$. □

4.C Proofs for Section 4.7

Proof of Theorem 4.2. Starting from inequality (4.25), by the exact same manipulations as in Corollary 4.5:

$$\begin{aligned}
R_T(x^*) \leq\ & \sum_{t=0}^{T}(q_t(x^*) - q_t(x_{t+1})) + \sum_{t=1}^{T}\langle g_t, x_t - x_{t+1}\rangle \\
& + \sum_{t=1}^{T}(p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T} -\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) \\
=\ & \sum_{t=0}^{T}\langle \tilde{g}_{t+1} - \tilde{g}_t, x^* - x_{t+1}\rangle + \sum_{t=1}^{T}\langle g_t, x_t - x_{t+1}\rangle \\
& + \sum_{t=0}^{T}(\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T}(p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T} -\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t)
\end{aligned}$$

$$
\begin{aligned}
&= \langle \tilde{g}_{T+1}, x^* \rangle + \sum_{t=0}^{T} \langle \tilde{g}_{t+1}, -x_{t+1} \rangle + \sum_{t=1}^{T} \langle \tilde{g}_t, x_{t+1} \rangle + \sum_{t=1}^{T} \langle g_t, x_t - x_{t+1} \rangle \\
&\quad + \sum_{t=0}^{T} (\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T} -\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) \\
&= \langle \tilde{g}_{T+1}, x^* - x_{T+1} \rangle + \sum_{t=1}^{T} \langle g_t - \tilde{g}_t, x_t - x_{t+1} \rangle \\
&\quad + \sum_{t=0}^{T} (\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T} -\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) \qquad (4.31) \\
&\leq \langle \tilde{g}_{T+1}, x^* - x_{T+1} \rangle + \tilde{q}_T(x^*) - \tilde{q}_T(x_{T+1}) \\
&\quad + \sum_{t=0}^{T-1} (\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) + \sum_{t=1}^{T} \frac{1}{2} \| g_t - \tilde{g}_t \|_{(t),*}^2 ,
\end{aligned}
$$

using the Fenchel-Young for the second term, and Assumption 4.8 for the last term, in the final step. Finally, note that the left-hand side is independent of $\tilde{q}_T$ and $\tilde{g}_{T+1}$, and without loss of generality, we can set them to zero, which makes the first two terms of the right-hand side zero, hence finishing the proof. □

Proof of Theorem 4.3. Define $G_t = \| g_t - \tilde{g}_t \|_*^2$, and let $\lambda_t := \eta_t / 2$. Starting from (4.31), and using the fact that setting $\tilde{g}_{T+1} = 0$ does not affect the value of $R_T(x^*)$, we get

$$
\begin{aligned}
R_T(x^*) &\leq \sum_{t=1}^{T} \langle g_t - \tilde{g}_t, x_t - x_{t+1} \rangle + \sum_{t=1}^{T} -\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) \\
&\quad + \sum_{t=0}^{T} (\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) \\
&\leq \sum_{t=0}^{T} (\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) \\
&\quad + \sum_{t=1}^{T} -\frac{\eta_t}{2} \| x_t - x_{t+1} \|^2 + \sum_{t=1}^{T} \frac{\lambda_t}{2} \| x_t - x_{t+1} \|^2 + \sum_{t=1}^{T} \frac{1}{2\lambda_t} \| g_t - \tilde{g}_t \|_*^2 , \\
&\leq \sum_{t=0}^{T} (\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) \\
&\quad + \sum_{t=1}^{T} \frac{-\eta_t}{4} \| x_t - x_{t+1} \|^2 + \sum_{t=1}^{T} \frac{1}{\eta_t} G_t \\
&\leq \sum_{t=0}^{T} (\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t))
\end{aligned}
$$

$$+\sum_{t=1}^{T} \frac{-2L^2}{\eta_{t+1}} \|x_t - x_{t+1}\|^2 + \sum_{t=1}^{T} \frac{1}{\eta_t} G_t \,.$$

In the second inequality, we used Assumption 4.8 and the Fenchel-Young inequality. In the last inequality, we used the assumption $\eta_t \eta_{t+1} \geq 8L^2$. Now, let $x_0 := x_1$ and $f_0 := 0$, so that $\tilde{g}_1 = g_0 = \nabla f_0(x_0)$. Then, using ideas from Lemma 12 of Chiang *et al.* [18],

$$\begin{aligned}
\sum_{t=1}^{T} \frac{1}{\eta_t} G_t &= \sum_{t=1}^{T} \frac{1}{\eta_t} \|\nabla f_t(x_t) - \nabla f_{t-1}(x_{t-1})\|_*^2 \\
&\leq \sum_{t=1}^{T} 2\frac{1}{\eta_t} \|\nabla f_t(x_t) - \nabla f_{t-1}(x_t)\|_*^2 + \sum_{t=1}^{T} 2\frac{1}{\eta_t} \|\nabla f_{t-1}(x_t) - \nabla f_{t-1}(x_{t-1})\|_*^2 \\
&\leq 2\sum_{t=1}^{T} \frac{1}{\eta_t} \|\nabla f_t(x_t) - \nabla f_{t-1}(x_t)\|_*^2 + \sum_{t=2}^{T} \frac{2L^2}{\eta_t} \|x_t - x_{t-1}\|^2 \\
&\leq 2\sum_{t=1}^{T} \frac{1}{\eta_t} \|\nabla f_t(x_t) - \nabla f_{t-1}(x_t)\|_*^2 + \sum_{t=1}^{T} \frac{2L^2}{\eta_{t+1}} \|x_{t+1} - x_t\|^2 \,.
\end{aligned}$$

Note that to get the first inequality, we used the fact that $\|\cdot\|^2$ is convex for any norm, together with Jensen's inequality, so that $\|x+y\|^2 = 4\|x/2+y/2\|^2 \leq 4(\|x\|^2/2+\|y\|^2/2) = 2\|x\|^2 + 2\|y\|^2$. This completes the proof. □

Proof of Corollary 4.4. First, note that since ψ_t is convex, by definition, the regularizer $r_{0:t} = \sum_{t=1}^{t} \frac{\eta_t - \eta_{t-1}}{2} \|\cdot - x_t\|^2$ is η_t-strongly-convex w.r.t. the norm $\|\cdot\|$, satisfying Assumption 4.8. Furthermore, Assumption 4.5 is satisfied by the convexity of f_t. Also, by assumption, $\mathcal{X}$ is closed and $R < +\infty$, so the objectives are always bounded below and Assumption 4.1 holds.

Let $G_t = \|g_t - \tilde{g}_t\|_*^2$, and define $C := \eta_t - \eta\sqrt{G_{1:t}} = 4RL^2$. Starting as in Corollary 4.3, and following the same steps as in the proof of Theorem 4.3, we have

$$\begin{aligned}
R_T^{(\ell)}(x^*) &\leq \sum_{t=0}^{T} (\tilde{q}_t(x^*) - \tilde{q}_t(x_{t+1})) + \sum_{t=1}^{T} (p_t(x^*) - p_t(x_t)) - \sum_{t=1}^{T} \frac{\eta_t}{4} \|x_t - x_{t+1}\|^2 + \sum_{t=1}^{T} \frac{1}{\eta_t} G_t \\
&\leq \sum_{t=1}^{T} \frac{1}{2}(\eta_t - \eta_{t-1})R^2 + \sum_{t=1}^{T} \frac{-\eta_t}{4} \|x_t - x_{t+1}\|^2 + \sum_{t=1}^{T} \frac{1}{\eta\sqrt{G_{1:t}}} G_t \\
&\leq \frac{1}{2}\eta_T R^2 + \sum_{t=1}^{T} \frac{-C}{4} \|x_t - x_{t+1}\|^2 + \frac{2}{\eta}\sqrt{G_{1:T}} \\
&\leq \frac{1}{2} C R^2 + \left(\frac{2}{\eta} + \frac{\eta}{2} R^2\right) \sqrt{G_{1:T}} + \sum_{t=1}^{T} \frac{-C}{4} \|x_t - x_{t+1}\|^2
\end{aligned}$$

In the first line, we used, as in Theorem 4.3, the Fenchel-Young inequality with $\lambda_t = \eta_t/2$. In the second line, we dropped the $\tilde{q}_t = 0$ terms and used the definition of p_t and R, and the fact that $\eta_t \geq \eta_{t-1}$, to get the first term, and obtained the last term using the fact that $\eta_t \geq \eta\sqrt{G_{1:t}}$ by definition. In the third inequality, we let the η_t in the first term telescope, used the fact that $\eta_t > C$ in the second term, and Lemma 4.7 to get the last term. In the last line, we used the definition of η_T and grouped the $\sqrt{G_{1:T}}$ terms together.

Next, we use the inequalities $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ and $\sqrt{a} \leq \frac{1}{2} + a$ (for $a, b \geq 0$), as well as Jensen's inequality on $\|\cdot\|^2$ (as in the proof of Theorem 4.3) to bound $\sqrt{G_{1:T}}$ with $\sqrt{D_{\|\cdot\|}}$:

$$
\begin{aligned}
\sqrt{\sum_{t=1}^{T} G_t} &= \sqrt{\sum_{t=1}^{T} \|\nabla f_t(x_t) - \nabla f_{t-1}(x_{t-1})\|_*^2} \\
&\leq \sqrt{\sum_{t=1}^{T} 2\|\nabla f_t(x_t) - \nabla f_{t-1}(x_t)\|_*^2 + \sum_{t=1}^{T} 2\|\nabla f_{t-1}(x_t) - \nabla f_{t-1}(x_{t-1})\|_*^2} \\
&\leq \sqrt{2\sum_{t=1}^{T} \|\nabla f_t(x_t) - \nabla f_{t-1}(x_t)\|_*^2 + \sum_{t=1}^{T} 2L^2\|x_t - x_{t-1}\|^2} \\
&\leq \sqrt{2\sum_{t=1}^{T} \|\nabla f_t(x_t) - \nabla f_{t-1}(x_t)\|_*^2} + \sqrt{\sum_{t=1}^{T} 2L^2\|x_t - x_{t-1}\|^2} \\
&\leq \sqrt{2\sum_{t=1}^{T} \|\nabla f_t(x_t) - \nabla f_{t-1}(x_t)\|_*^2} + \frac{1}{2} + \sum_{t=1}^{T} 2L^2\|x_t - x_{t-1}\|^2 . \\
&= \sqrt{2\sum_{t=1}^{T} \|\nabla f_t(x_t) - \nabla f_{t-1}(x_t)\|_*^2} + \frac{1}{2} + \sum_{t=1}^{T} 2L^2\|x_{t+1} - x_t\|^2 ,
\end{aligned}
$$

where the last line follows, as in the proof of Theorem 4.3, by defining $x_0 = x_1$ and adding the extra positive term $2L^2\|x_{T+1} - x_T\|^2$. Putting back into the previous inequality,

$$
\begin{aligned}
R_T^{(\ell)}(x^*) \leq{}& \frac{1}{2}CR^2 + \sum_{t=1}^{T} \frac{-C}{4}\|x_t - x_{t+1}\|^2 \\
&+ \left(\frac{2}{\eta} + \frac{\eta}{2}R^2\right)\left(\sqrt{2D_{\|\cdot\|}} + \frac{1}{2} + \sum_{t=1}^{T} 2L^2\|x_{t+1} - x_t\|^2\right)
\end{aligned}
$$

$$= \frac{1}{2}CR^2 + \sum_{t=1}^{T} \frac{-C}{4}\|x_t - x_{t+1}\|^2$$
$$+ 2R\sqrt{2D_{\|\cdot\|}} + R + \sum_{t=1}^{T} 4RL^2\|x_{t+1} - x_t\|^2$$
$$= \frac{1}{2}4R^3L^2 + R + 2R\sqrt{2D_{\|\cdot\|}}\,.$$

In the first equality, we used $\eta = 2/R$ while in the last one we used that $C = 4RL^2$ by definition. This completes the proof. □

Lemma 4.7 (Lemma 4 of [69])**.** *For any non-negative numbers $a_1, a_2, \ldots, a_T$ with $a_1 > 0$,*

$$\sum_{t=1}^{T} \frac{a_t}{\sqrt{\sum_{s=1}^{t} a_s}} \le 2\sqrt{\sum_{t=1}^{T} a_t}\,.$$

4.D Proofs for Section 4.8

Proof of Theorem 4.4. As promised, we will use Corollary 4.7 to prove the theorem. First notice that defining the norm in the corollary as $\sqrt{L}\|\cdot\|$ and the norms $\|\cdot\|_{(t)} = \sqrt{cL}\|\cdot\|$ (with dual norm $\|\cdot\|_{(t),*} = \frac{1}{\sqrt{cL}}\|\cdot\|$), the conditions of the corollary are satisfied. Therefore, by Corollary 4.7, after moving the composite ϕ terms and D to the left-hand side, we get

$$\mathbb{E}\left\{\sum_{t=1}^{T} \ell(x_{t+1}) - \ell(x^*)\right\} \le \mathbb{E}\left\{\frac{(1+c)L}{2}\|x^* - x_1\|^2 + \sum_{t=1}^{T} \frac{1}{2cL}\|\sigma_t\|_{(t),*}^2\right\},$$

for Ada-MD, and

$$\mathbb{E}\left\{\sum_{t=1}^{T} \ell(x_{t+1}) - \ell(x^*)\right\} \le \mathbb{E}\left\{\frac{(1+c)L}{2}\|x^*\|^2 + \sum_{t=1}^{T} \frac{1}{2cL}\|\sigma_t\|_{(t),*}^2\right\},$$

for Ada-FTRL. Let $R_T^\ell := \sum_{t=1}^{T} \ell(x_{t+1}) - \ell(x^*)$. Then, by Lemma 4.2 and the definition of the dual norm $\|\cdot\|_{(t),*}$,

$$\mathbb{E}\{R_T^\ell\} \le (1+c)LR + \sum_{t=1}^{T} \mathbb{E}\left\{\frac{4L\ \mathbb{E}\{\ell(x_t) - \ell(x^*) + \ell(\tilde{x}_s) - \ell(x^*)\}}{2cL}\right\}$$
$$= (1+c)LR + \frac{2}{c}\mathbb{E}\{R_T^\ell\} + \sum_{t=1}^{T} \mathbb{E}\left\{\frac{4L\ \mathbb{E}\{\ell(x_t) - \ell(x_{t+1}) + \ell(\tilde{x}_s) - \ell(x^*)\}}{2cL}\right\}.$$
$$= (1+c)LR + \frac{2}{c}\mathbb{E}\{R_T^\ell\} + \frac{2}{c}\mathbb{E}\{\ell(x_1) - \ell(x_{T+1})\} + \frac{2}{c}\sum_{t=1}^{T} \mathbb{E}\{\ell(\tilde{x}_s) - \ell(x^*)\}\,.$$

Next, we bound the last summation above. Let T_s denote the set of indices t such that g_t is calculated in the s-th epoch, and let $\tilde{x}_s$ be the first iterate used (in g_t) in epoch s. We have:

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}\{\ell(\tilde{x}_s) - \ell(x^*)\} &= \sum_{s=1}^{S} \sum_{t \in T_s} \mathbb{E}\{\ell(\tilde{x}_s) - \ell(x^*)\} \\
&\le m_1 \left(\ell(x_1) - \ell(x^*)\right) + \sum_{s=2}^{S} \mathbb{E}\left\{ m_s \left(\frac{1}{m_{s-1}} \sum_{t \in T_{s-1}} (\ell(x_t) - \ell(x^*)) \right) \right\} \\
&= m_1 \left(\ell(x_1) - \ell(x^*)\right) + \sum_{s=1}^{S-1} \mathbb{E}\left\{ \frac{m_{s+1}}{m_s} \sum_{t \in T_s} (\ell(x_t) - \ell(x^*)) \right\} \\
&\le m_1 \left(\ell(x_1) - \ell(x^*)\right) + \sum_{s=1}^{S-1} \mathbb{E}\left\{ b \sum_{t \in T_s} (\ell(x_t) - \ell(x^*)) \right\} \qquad (4.32) \\
&\le m_1 \left(\ell(x_1) - \ell(x^*)\right) + b \sum_{s=1}^{S} \sum_{t \in T_s} \mathbb{E}\{\ell(x_t) - \ell(x^*)\} \\
&= m_1 \left(\ell(x_1) - \ell(x^*)\right) + b \sum_{t=1}^{T} \mathbb{E}\{\ell(x_t) - \ell(x^*)\} \\
&= m_1 \left(\ell(x_1) - \ell(x^*)\right) + b\mathbb{E}\{\ell(x_1) - \ell(x_{T+1})\} + b\mathbb{E}\big\{R_T^\ell\big\} \\
&\le (b + m_1) \left(\ell(x_1) - \ell(x^*)\right) + b\mathbb{E}\big\{R_T^\ell\big\}.
\end{aligned}$$

Putting together,

$$\mathbb{E}\big\{R_T^\ell\big\} \le (1+c)LR + \frac{2+2b}{c}\mathbb{E}\big\{R_T^\ell\big\} + \frac{2+2b+2m_1}{c}\big(\ell(x_1) - \ell(x^*)\big).$$

Re-arranging and applying Jensen's inequality proves the first part of the theorem. To prove the second part, we start from (4.32):

$$\begin{aligned}
&m_1 \left(\ell(x_1) - \ell(x^*)\right) + \sum_{s=1}^{S-1} \mathbb{E}\left\{ b \sum_{t \in T_s} \ell(x_t) - \ell(x^*) \right\} \\
&\quad \le (m_1 + b) \left(\ell(x_1) - \ell(x^*)\right) + b \sum_{s=1}^{S-1} \sum_{t \in T_s} \mathbb{E}\{\ell(x_{t+1}) - \ell(x^*)\}.
\end{aligned}$$

Now, if $c = 2 + 2b$, we have $1 - 2/c = 2b/c$. Thus,

$$\begin{aligned}
&\left(1 - \frac{2}{c}\right) \sum_{s=1}^{S} \sum_{t \in T_s} \mathbb{E}\{\ell(x_{t+1}) - \ell(x^*)\} \\
&= \left(1 - \frac{2}{c}\right) \mathbb{E}\big\{R_T^\ell\big\}
\end{aligned}$$

$$\leq (1+c)LR + \frac{2+2b+2m_1}{c}\big(\ell(x_1)-\ell(x^*)\big) + \frac{2b}{c}\sum_{s=1}^{S-1}\sum_{t\in T_s}\mathbb{E}\{\ell(x_{t+1})-\ell(x^*)\}\,.$$

Rearranging gives

$$\left(1-\frac{2}{c}\right)\sum_{t\in T_S}\mathbb{E}\{\ell(x_{t+1})-\ell(x^*)\} \leq (1+c)LR + \frac{2+2b+2m_1}{c}\big(\ell(x_1)-\ell(x^*)\big)\,.$$

Dividing both sides by $(1-2/c)m_S$ and using Jensen's inequality on ℓ completes the proof. □

4.E Technical results

In this appendix, we have gathered some technical results required in our proofs. The first lemma states that the Bregman divergence is invariant under addition of affine functions.

Lemma 4.8. *Let $f : \mathcal{H} \to \overline{\mathbb{R}}$ be proper, and let $x, y \in \operatorname{dom}(f)$. Suppose that $v \in \mathcal{H}$, and $w \in \mathbb{R}$, and let $g : \mathcal{H} \to \overline{\mathbb{R}}$ be given by $g(\cdot) = f(\cdot) + \langle v, \cdot\rangle + w$. Then,*

(i) g is proper, with $\operatorname{dom}(g) = \operatorname{dom}(f)$.

(ii) For any $z \in \mathcal{H}$, the derivative $g'(x; z)$ exists in $[-\infty, +\infty]$ if and only if $f'(x; z)$ exists in $[-\infty, +\infty]$, in which case

$$g'(x;z) = f'(x;z) + \langle v, z\rangle\,.$$

(iii) If $f'(x; y-x)$ or $g'(x; y-x)$ exist, then $\mathcal{B}_g(y, x) = \mathcal{B}_f(y, x)$.

Proof. That g is proper and $\operatorname{dom}(f) = \operatorname{dom}(g)$ is immediate since $\operatorname{dom}(\langle v, \cdot\rangle) = \mathcal{H}$ and $w \in \mathbb{R}$. Then $x, y \in \operatorname{dom}(g)$, and for any $z \in \mathcal{H}$, if either of $f'(x; z)$ or $g'(x; z)$ exist in $[-\infty, +\infty]$,

$$f'(x;z) + \langle v, z\rangle = \lim_{\alpha\downarrow 0}\frac{f(x+\alpha z) + \langle v, x+\alpha z\rangle + w - f(x) - \langle v, x\rangle - w}{\alpha} = g'(x; v)\,,$$

which proves the second part of the lemma. Letting $z = y - x$ and using the definition of $\mathcal{B}_g$ gives $\mathcal{B}_f(y, x) = \mathcal{B}_g(y, x)$. □

The next proposition gathers useful results based on Proposition 17.2 of Bauschke and Combettes [6].

Proposition 4.1. *Let f be proper and convex, and let $x, y \in \operatorname{dom}(f)$ and $z \in \mathcal{H}$. Then,*

(i) $f'(x; z)$ exists in $[-\infty, +\infty]$ and

$$f'(x; z) = \inf_{\alpha \in (0,+\infty)} \frac{f(x + \alpha z) - f(x)}{\alpha} \,.$$

(ii) $f'(x; y - x) < +\infty$.

(iii) $\mathcal{B}_f(y, x) \geq 0$.

Proof. Part ((i)) is proved in Proposition 17.2(ii) of Bauschke and Combettes [6]. Also, by their Proposition 17.2(iii),

$$f'(x; y - x) + f(x) \leq f(y)\,,$$

proving part ((ii)) since $f(y)$ and $f(x)$ are both real numbers. Part ((iii)) then simply follows from the same equation, with the Bregman divergence being real and non-negative when $f'(x; y - x)$ is real-valued, and $+\infty$ when $f'(x; y - x) = -\infty$. □

The next lemma is useful for decomposing Bregman divergences.

Lemma 4.9. *Let $r : \mathcal{H} \to \overline{\mathbb{R}}$ and $q : \mathcal{H} \to \overline{\mathbb{R}}$ be directionally differentiable (hence also proper). Let $S := \operatorname{dom}(r) \cap \operatorname{dom}(q)$, suppose $S \neq \emptyset$, and let $x, y \in S$. Suppose that at least one of the two limits $q'(x; y - x)$ and $r'(x; y - x)$ is finite. Then,*

$$\mathcal{B}_r(y, x) - \mathcal{B}_q(y, x) = \mathcal{B}_p(y, x)\,,$$

where $p : \mathcal{H} \to \overline{\mathbb{R}}$ is given by

$$p(z) := \begin{cases} r(z) - q(z) & z \in \operatorname{dom}(r) \cup \operatorname{dom}(q)\,, \\ +\infty & \textit{otherwise}\,. \end{cases}$$

Proof. By the assumption that at least one of them is finite, we can take the difference of the two limits $q'(x; y - x)$ and $r'(x; y - x)$ to obtain

$$\begin{aligned} -r'(x; y - x) + q'(x; y - x) &= \lim_{\alpha \downarrow 0} \frac{-r(x + \alpha(y - x)) + r(x)}{\alpha} + \lim_{\alpha \downarrow 0} \frac{q(x + \alpha(y - x)) - q(x)}{\alpha} \\ &= \lim_{\alpha \downarrow 0} \frac{-p(x + \alpha(y - x)) + p(x)}{\alpha} = -p'(x; y - x)\,. \qquad (4.33) \end{aligned}$$

In the derivation above, we have used that at most one of $r(x+\alpha(y-x))$ and $q(x+\alpha(y-x))$ can remain infinite as $\alpha \downarrow 0$. Formally, there exists an $\epsilon > 0$ such that for all $\alpha < \epsilon$, the summation $-r(x+\alpha(y-x))+q(x+\alpha(y-x))$ is well defined, and is equal, by definition, to $-p(x+\alpha(y-x))$. Adding the real-valued equation $r(y)-r(x)-q(y)+q(x)=p(y)-p(x)$ to (4.33) completes the proof. $\square$

Note that, in light of Proposition 4.1, the above lemma holds if both q and r are proper convex functions, given the condition that at least one of the directional derivatives is finite (i.e., not equal to $-\infty$). The latter condition is needed even if p is also a proper convex function (note that this requires that $\mathrm{dom}(r) \subset \mathrm{dom}(q)$). In this case, Proposition 4.1 (ii) shows that the sum of the limits $p'(x; y-x)$ and $q'(x; y-x)$ is well-defined for any $x, y \in \mathrm{dom}(p) \cap \mathrm{dom}(q)$, and is equal to $r'(x; y-x)$. Therefore, since the function values are finite, we get $\mathcal{B}_p(y,x)+\mathcal{B}_q(y,x)=\mathcal{B}_r(y,x)$. On the other hand, $\mathcal{B}_r(y,x)-\mathcal{B}_q(y,x)=\mathcal{B}_p(y,x)$ only holds if the left-hand side is well-defined, which happens exactly if at least one of the Bregman divergences (necessarily $\mathcal{B}_q(y,x)$) is finite (equivalently, at least one of the directional derivatives is finite). An example illustrating this situation is when all functions are constant multiples of $-\sqrt{x}\,\mathbb{I}\{x \geq 0\}+\infty\,\mathbb{I}\{x<0\}$, defined over the reals, in which case the Bregman divergences become $+\infty$ for $x=0$ and $y>0$.

4.F Proof of Theorem 4.1

In this section, we provide a detailed proof of Theorem 4.1. First, we prove generalized versions of two lemmas that have appeared in several previous work; see, e.g., the work of Dekel *et al.* [24] and the references therein.

The first lemma is used for Ada-FTRL.

Lemma 4.10. *Let $g \in \mathcal{H}$ and consider a proper, directionally differentiable function $r : \mathcal{H} \to \overline{\mathbb{R}}$. Define $S = \mathrm{dom}(r)$, and let $\mathcal{X} \subset \mathcal{H}$ be a convex set such that $\mathcal{X} \cap S \neq \emptyset$. Further assume that $\arg\min_{x\in\mathcal{X}} \langle g, x\rangle + r(x)$ is non-empty. Then, for any $x^+ \in \arg\min_{x\in\mathcal{X}} \langle g, x\rangle + r(x)$ and any $x \in \mathcal{X} \cap S$,*

$$+\infty > \langle g, x-x^+\rangle + r(x) - r(x^+) \geq \mathcal{B}_r(x, x^+)\,. \tag{4.34}$$

Proof. Let $h : \mathcal{H} \to \overline{\mathbb{R}}$ be given by $h(\cdot) = \langle g, \cdot\rangle + r(\cdot)$, so that $x^+ \in \arg\min_{x\in\mathcal{X}} h(x)$. Note that by Lemma 4.8, $\mathrm{dom}(h) = S$ and h is directionally differentiable with $h'(x; z) =$

$\langle g, z\rangle + r'(x; z)$ for all $x \in S$ and $z \in \mathcal{H}$. Also note that $x^+ \in \mathcal{X} \cap S$ by definition. Since $x, x^+ \in \mathcal{X}$, and $\mathcal{X}$ is convex, for all $\alpha \in [0, 1]$, we have $x^+ + \alpha(x - x^+) \in \mathcal{X}$. Therefore, the optimality of x^+ over $\mathcal{X}$ implies that for all $\alpha \in (0, 1)$,

$$\frac{h(x^+ + \alpha(x - x^+)) - h(x^+)}{\alpha} \geq 0\,.$$

Thus, $0 \leq h'(x^+; x - x^+) = \langle g, x - x^+\rangle + r'(x^+; x - x^+)$, and therefore $+\infty > \langle g, x - x^+\rangle \geq -r'(x^+; x - x^+)$. Adding the real number $r(x) - r(x^+)$ to the sides completes the proof. $\square$

The second lemma is used for Ada-MD.

Lemma 4.11. *Let $\mathcal{X}, S, g$ and r be as in Lemma 4.10. Let $y \in S \cap \mathcal{X}$ be such that $r'(y; \cdot - y)$ is real-valued and concave on S, that is, for all $x_1, x_2 \in S$ and all $\alpha \in [0, 1]$ for which $x_\alpha := x_1 + \alpha(x_2 - x_1) \in S$,*

$$+\infty > r'(y; x_\alpha - y) \geq \alpha r'(y; x_2 - y) + (1 - \alpha) r'(y; x_1 - y) > -\infty\,.$$

Let $q : \mathcal{H} \to \overline{\mathbb{R}}$ be proper and directionally differentiable, with $S_q := S \cap \mathcal{X} \cap \operatorname{dom}(q) \neq \emptyset$. Assume that $\mathcal{X}^+ := \arg\min_{x \in \mathcal{X}} \langle g, x\rangle + q(x) + \mathcal{B}_r(x, y)$ is non-empty, and the associated optimal value is finite. Then, for any $x^+ \in \mathcal{X}^+$ and any $x \in S_q$,

$$+\infty > \langle g, x - x^+\rangle + q(x) - q(x^+) + \mathcal{B}_r(x, y) - \mathcal{B}_r(x^+, y) \geq \mathcal{B}_{r+q}(x, x^+)\,. \qquad (4.35)$$

Proof. Let $h : \mathcal{H} \to \overline{\mathbb{R}}$ be given by $h(\cdot) = \langle g, \cdot\rangle + q + \mathcal{B}_r(\cdot, y)$, so that $x^+ \in \arg\min_{x \in \mathcal{X}} h(x)$. Note that by assumption, $h(x^+) < +\infty$. In addition, $\operatorname{dom}(h) \subset S \cap \operatorname{dom}(q)$. Thus, $x^+ \in S_q$.

Now, fix $\alpha \in (0, 1)$, and let $x_\alpha = x^+ + \alpha(x - x^+)$. If $x_\alpha \in S_q$, then $q(x_\alpha)$ and $r(x_\alpha)$ are real-valued, and by the optimality of x^+ over $\mathcal{X}$ and the concavity of $r'(y; \cdot - y)$ over S,

$$\begin{aligned}
0 &\leq h(x_\alpha) - h(x^+) = q(x_\alpha) - q(x^+) + \langle g, x^+ + \alpha(x - x^+) - x^+\rangle + \mathcal{B}_r(x_\alpha, y) - \mathcal{B}_r(x^+, y) \\
&= q(x_\alpha) - q(x^+) + \alpha\langle g, x - x^+\rangle + r(x_\alpha) - r(x^+) \\
&\quad - r'(y; x_\alpha - y) + r'(y; x^+ - y) \\
&\leq q(x_\alpha) - q(x^+) + \alpha\langle g, x - x^+\rangle + r(x_\alpha) - r(x^+) \\
&\quad - \big((1 - \alpha) r'(y; x^+ - y) + \alpha r'(y; x - y)\big) + r'(y; x^+ - y) \\
&= q(x_\alpha) - q(x^+) + \alpha\langle g, x - x^+\rangle + r(x_\alpha) - r(x^+) + \alpha\left(r'(y; x^+ - y) - r'(y; x - y)\right)\,,
\end{aligned}$$

Suppose, on the other hand, that $x_\alpha \notin S_q$. Then, given that by the assumption of convexity of $\mathcal{X}$, $x_\alpha \in \mathcal{X}$, we must have $x_\alpha \notin S \cap \mathrm{dom}(q)$, so that $(r+q)(x_\alpha) = +\infty$. In addition, $r'(y; \cdot - y)$ is real-valued over S and $x, x^+ \in S$, so $r'(y; x^+ - y) - r'(y; x - y)$ is real-valued. Putting this together, we will again have that for $x_\alpha \notin S_q$,

$$0 \le q(x_\alpha) - q(x^+) + \alpha\langle g, x - x^+\rangle + r(x_\alpha) - r(x^+) + \alpha\left(r'(y; x^+ - y) - r'(y; x - y)\right) ,$$

Thus, dividing by the positive α, for all $\alpha \in (0,1)$, we have

$$0 \le \langle g, x - x^+\rangle + \frac{q(x_\alpha) - q(x^+) + r(x_\alpha) - r(x^+)}{\alpha} - r'(y; x - y) + r'(y; x^+ - y) .$$

Taking infimum over α, we obtain

$$\begin{aligned}0 &\le \langle g, x - x^+\rangle - r'(y; x - y) + r'(y; x^+ - y) + \inf_{\alpha\in(0,1)} \frac{q(x_\alpha) - q(x^+) + r(x_\alpha) - r(x^+)}{\alpha} \\ &\le \langle g, x - x^+\rangle - r'(y; x - y) + r'(y; x^+ - y) + (r+q)'(x^+; x - x^+) ,\end{aligned}$$

using directional differentiability of $q + r$ in the final step. Adding the real-valued equation $0 = q(x) - q(x^+) + r(x) - r(y) + r(y) - r(x^+) + (r+q)(x^+) - (r+q)(x)$, using the definition of Bregman divergence, and rearranging terms completes the proof. $\square$

We can now prove Theorem 4.1.

Proof of Theorem 4.1. First consider Ada-FTRL. For $t = 0, 1, \ldots, T$, let $h_t^{\mathrm{ftrl}} : \mathcal{H} \to \overline{\mathbb{R}}$ be given by $h_t^{\mathrm{ftrl}}(\cdot) := \langle g_{1:t}, \cdot\rangle + q_t(\cdot) + r_{1:t}(\cdot)$, recalling that $c_{i:j} \equiv 0$ whenever $i > j$. Let $S_t = \mathrm{dom}(r_{1:t})$. By Assumption 4.1, for $t = 1, 2, \ldots, T$,

$$-\infty < h_{t-1}^{\mathrm{ftrl}}(x_t) = \langle g_{1:t-1}, x_t\rangle + q_{t-1}(x_t) + r_{1:t-1}(x_t) < +\infty .$$

Therefore, $x_t \in \mathrm{dom}(r_{1:t-1} + q_{t-1})$. In addition, by (4.4), $x_t \in \mathrm{dom}(p_t)$. Thus, $x_t \in \mathrm{dom}(r_{1:t-1} + q_{t-1} + p_t)$, that is, $x_t \in \mathrm{dom}(r_{1:t}) = S_t$. Furthermore, $h_t^{\mathrm{ftrl}}(x_{t+1}) < +\infty$, so $x_{t+1} \in \mathrm{dom}(q_t) \cap \mathrm{dom}(r_{1:t})$. Thus, $x_t, x_{t+1} \in \mathcal{X} \cap S_t \subset \cap_{s=1}^t \left(\mathrm{dom}(q_{s-1}) \cap \mathrm{dom}(p_s)\right)$.

Now, for any $t = 1, 2, \ldots, T$, since x_t minimizes p_t over $\mathcal{X}$, if we add p_t to the objective of the optimization above, we will still have

$$x_t \in \operatorname*{arg\,min}_{x\in\mathcal{X}} h_{t-1}^{\mathrm{ftrl}}(x) + p_t(x) = \operatorname*{arg\,min}_{x\in\mathcal{X}} \langle g_{1:t-1}, x\rangle + r_{1:t}(x) .$$

By Assumption 4.1, $r_{1:t}$ is directionally differentiable. Therefore, for any $t = 1, 2, \ldots, T$, we can apply Lemma 4.10 with $g \leftarrow g_{1:t-1}$, $r \leftarrow r_{1:t}$, $x^+ \leftarrow x_t$, and $x \leftarrow x_{t+1}$, to obtain

$$\begin{aligned}
-\infty &< \langle g_{1:t-1}, x_t - x_{t+1} \rangle + p_{1:t}(x_t) - p_{1:t}(x_{t+1}) + q_{0:t-1}(x_t) - q_{0:t-1}(x_{t+1}) \\
&\le -\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t)\,.
\end{aligned}$$

In the inequality above, the right-hand side cannot be equal to $-\infty$ (by Lemma 4.10), and all other terms are real-valued. Thus, we can sum up this inequality over $t = 1, 2, \ldots, T$, to obtain

$$\begin{aligned}
&-\sum_{t=1}^{T} \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) \\
\ge\ & \sum_{t=1}^{T} \langle g_{1:t-1}, x_t \rangle - \sum_{t=1}^{T} \langle g_{1:t-1}, x_{t+1} \rangle + \sum_{t=1}^{T} p_{1:t}(x_t) - \sum_{t=1}^{T} p_{1:t}(x_{t+1}) \\
& + \sum_{t=1}^{T} q_{0:t-1}(x_t) - \sum_{t=1}^{T} q_{0:t-1}(x_{t+1}) \\
=\ & \sum_{t=0}^{T-1} \langle g_{1:t}, x_{t+1} \rangle - \sum_{t=1}^{T} \langle g_{1:t-1}, x_{t+1} \rangle + \sum_{t=1}^{T} p_{1:t}(x_t) - \sum_{t=2}^{T+1} p_{1:t-1}(x_t) \\
& + \sum_{t=0}^{T-1} q_{0:t}(x_{t+1}) - \sum_{t=0}^{T} q_{0:t-1}(x_{t+1}) \\
=\ & \sum_{t=1}^{T-1} \langle g_{1:t}, x_{t+1} \rangle - \sum_{t=1}^{T} \langle g_{1:t-1}, x_{t+1} \rangle + \sum_{t=1}^{T} p_{1:t}(x_t) - \sum_{t=1}^{T+1} p_{1:t-1}(x_t) \\
& + \sum_{t=0}^{T-1} q_{0:t}(x_{t+1}) - \sum_{t=0}^{T} q_{0:t-1}(x_{t+1}) \\
=\ & -\langle g_{1:T}, x_{T+1} \rangle + \sum_{t=1}^{T} \langle g_t, x_{t+1} \rangle + \sum_{t=1}^{T} p_t(x_t) - p_{1:T}(x_{T+1}) - q_{0:T}(x_{T+1}) + \sum_{t=0}^{T} q_t(x_{t+1}) \\
=\ & \sum_{t=1}^{T} \langle g_t, x_{t+1} \rangle + \sum_{t=1}^{T} p_t(x_t) + \sum_{t=0}^{T} q_t(x_{t+1}) - \Big(\langle g_{1:T}, x_{T+1} \rangle + p_{1:T}(x_{T+1}) + q_{0:T}(x_{T+1}) \Big) \\
\ge\ & \sum_{t=1}^{T} \langle g_t, x_{t+1} \rangle + \sum_{t=1}^{T} p_t(x_t) + \sum_{t=0}^{T} q_t(x_{t+1}) - \Big(\langle g_{1:T}, x^* \rangle + p_{1:T}(x^*) + q_{0:T}(x^*) \Big) \\
=\ & R_T^+(x^*) + \sum_{t=1}^{T} p_t(x_t) + \sum_{t=0}^{T} q_t(x_{t+1}) - p_{1:T}(x^*) - q_{0:T}(x^*)\,,
\end{aligned}$$

using, in the last inequality, the optimality of x_{T+1} over $\mathcal{X}$, as well as the fact that p_t, q_t are proper and all terms on the right-hand side not involving x^* are real-valued (hence the term in the parentheses involving x^* is well-defined and can be added to the rest of the expression). Now, if $x^* \notin \operatorname{dom}(p_{1:T} + q_{0:T})$, the bound of Theorem 4.1 holds trivially (recalling that the Bregman divergences cannot be $+\infty$). Otherwise, $(p_{1:T} + q_{0:T})(x^*)$ is real-valued, and rearranging completes the proof for Ada-FTRL.

For Ada-MD, we start by presenting the implications of Assumption 4.2.

To simplify notation, let $x_0 := g_0 := 0$, and define $h_t^{\mathrm{md}} := \langle g_t, x\rangle + q_t(x) + \mathcal{B}_{r_{1:t}}(x, x_t)$ and $S_t = \operatorname{dom}(r_{1:t})$ for $t = 0, 1, \ldots, T$ (so that $S_0 = \operatorname{dom}(r_{1:0}) = \mathcal{H}$). Then, by Assumption 4.2, $h_t^{\mathrm{md}}(x_{t+1}) < +\infty$ for all $t = 0, 1, \ldots, T$, so $x_{t+1} \in \mathcal{X} \cap \operatorname{dom}(q_t) \cap S_t$. Thus, given that $r'_{1:t}(x_t; \cdot - x_t)$ is real-valued on S_t, and $x_t \in S_t$ by assumption, $\mathcal{B}_{r_{1:t}}(x_{t+1}, x_t)$ is also real-valued.

Now, note that by the optimality of x_{T+1}, and because $h_T^{\mathrm{md}}(x_{T+1})$ is finite, for all $x^* \in \mathcal{X}$,

$$\langle g_T, x_{T+1} - x^*\rangle \le q_T(x^*) - q_T(x_{T+1}) + \mathcal{B}_{r_{1:T}}(x^*, x_T) - \mathcal{B}_{r_{1:T}}(x_{T+1}, x_T)\,. \tag{4.36}$$

Next, fix $t \in \{0, 1, 2, \ldots, T-1\}$ and suppose that $p_{t+1}(x^*)$ is finite-valued. Then, by the definition of p_{t+1}, we have $x^* \in \mathcal{X} \cap \operatorname{dom}(q_t)$ and $x^* \in \operatorname{dom}(r_{t+1}) = \operatorname{dom}(r_{1:t+1}) \subset S_t$. Furthermore, by the argument above, $x_{t+1} \in \mathcal{X} \cap S_t \cap \operatorname{dom}(q_t)$ and $x_t \in S_t$. Thus, for all $t = 0, 1, \ldots, T-1$, we can apply Lemma 4.11 with $g \leftarrow g_t$, $r \leftarrow r_{1:t}$, $q \leftarrow q_t$, $y \leftarrow x_t$, $x^+ \leftarrow x_{t+1}$, and $x \leftarrow x^*$, to obtain

$$\begin{aligned}\langle g_t, x_{t+1} - x^*\rangle \le{}& q_t(x^*) - q_t(x_{t+1}) + \mathcal{B}_{r_{1:t}}(x^*, x_t) - \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t)\\ &- \mathcal{B}_{r_{1:t}+q_t}(x^*, x_{t+1})\,.\end{aligned} \tag{4.37}$$

Note that this also implies that the right-hand side above cannot be $-\infty$, and only the last Bregman divergence term could be infinite. Now, since $r'_{1:t+1}(x_{t+1}; \cdot - x_{t+1})$ is real-valued on S_{t+1}, $r_{1:t} + q_t$ is directionally differentiable, and $x^* \in S_{t+1}$, by Lemma 4.9 we have

$$\mathcal{B}_{r_{1:t+1}}(x^*, x_{t+1}) - \mathcal{B}_{r_{1:t}+q_t}(x^*, x_{t+1}) = \mathcal{B}_{p_{t+1}}(x^*, x_{t+1})\,.$$

In particular, this implies that $\mathcal{B}_{p_{t+1}}(x^*, x_{t+1})$ cannot be $-\infty$. Moving the (real-valued) first term to the right-hand side, and substituting into (4.37), we have

$$\begin{aligned}\langle g_t, x_{t+1} - x^*\rangle \le{}& q_t(x^*) - q_t(x_{t+1}) + \mathcal{B}_{r_{1:t}}(x^*, x_t) - \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) + \mathcal{B}_{p_{t+1}}(x^*, x_{t+1})\\ &- \mathcal{B}_{r_{1:t+1}}(x^*, x_{t+1})\,.\end{aligned}$$

In light of the above, if $p_{t+1}(x^*)$ is finite-valued for all $t = 0, 1, 2, \ldots, T-1$, then summing up the above inequality for all these ts and adding (4.36), we obtain

$$\begin{aligned}
\sum_{t=0}^{T} \langle g_t, x_{t+1} - x^* \rangle \leq & \sum_{t=0}^{T-1} \big(q_t(x^*) - q_t(x_{t+1})\big) + \sum_{t=0}^{T-1} \mathcal{B}_{p_{t+1}}(x^*, x_{t+1}) \\
& + \sum_{t=0}^{T-1} \mathcal{B}_{r_{1:t}}(x^*, x_t) - \sum_{t=0}^{T-1} \mathcal{B}_{r_{1:t+1}}(x^*, x_{t+1}) - \sum_{t=0}^{T-1} \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) \\
& + q_T(x^*) - q_T(x_{T+1}) + \mathcal{B}_{r_{1:T}}(x^*, x_T) - \mathcal{B}_{r_{1:T}}(x_{T+1}, x_T) \\
= & \sum_{t=0}^{T} \big(q_t(x^*) - q_t(x_{t+1})\big) + \sum_{t=0}^{T-1} \mathcal{B}_{p_{t+1}}(x^*, x_{t+1}) \\
& + \sum_{t=1}^{T} \mathcal{B}_{r_{1:t}}(x^*, x_t) - \sum_{t=0}^{T-1} \mathcal{B}_{r_{1:t+1}}(x^*, x_{t+1}) - \sum_{t=1}^{T} \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t) \\
= & \sum_{t=0}^{T} \big(q_t(x^*) - q_t(x_{t+1})\big) + \sum_{t=1}^{T} \mathcal{B}_{p_t}(x^*, x_t) - \sum_{t=1}^{T} \mathcal{B}_{r_{1:t}}(x_{t+1}, x_t)\,,
\end{aligned}$$

and (4.9) holds.

On the other hand, if $p_{t+1}(x^*)$ is infinite for at least one t in $\{0, 1, 2, \ldots, T-1\}$, then $\mathcal{B}_{p_{t+1}}(x^*, x_{t+1}) = +\infty$ by definition. Therefore, the right-hand side of (4.9) will be $+\infty$, given that by the argument above, $\mathcal{B}_{p_{t+1}}(x^*, x_{t+1})$ cannot be equal to $-\infty$ if $p_{t+1}(x^*)$ is finite-valued. Thus, in this case as well, the bound of (4.9) holds trivially, completing the proof. $\square$

Chapter 5

Generically-Constrained Asynchronous Composite Optimization

Abstract

We present two new algorithms, AsynCADA and HedgeHog, for asynchronous sparse online and stochastic optimization. AsynCADA is the first asynchronous stochastic optimization algorithm with finite-time data-dependent convergence guarantees for generic convex constraints that, in addition: (a) allows for proximal (i.e., composite-objective) updates and adaptive step-sizes; (b) enjoys any-time convergence guarantees without requiring an exact global clock; and (c) when the data is sufficiently sparse, its convergence rate for (non-)smooth, (non-)strongly-convex, and even a limited class of non-convex objectives matches the corresponding serial rate, implying a theoretical "linear speed-up".

The second algorithm, HedgeHog, is an asynchronous parallel version of the Exponentiated Gradient (EG) algorithm for optimization over the probability simplex (a.k.a. Hedge in online learning), and the first asynchronous algorithm without SGD-style updates enjoying linear speed-ups under sparsity.

Unlike previous work, AsynCADA and HedgeHog and their convergence and speed-up analyses are not limited to individual coordinate-wise (i.e., "box-shaped") constraints or smooth and strongly-convex objectives. Underlying both results is a generic analysis framework that is of independent interest and further applicable to distributed and delayed feedback optimization.

5.1 Introduction

Many modern machine learning methods are based on iteratively optimizing a regularized objective. Given a convex, non-empty set of feasible model parameters $\mathcal{X} \subset \mathbb{R}^d$, a differentiable *loss* function $f : \mathbb{R}^d \to \mathbb{R}$, and a convex (possibly non-differentiable) *regularizer* function $\phi : \mathbb{R}^d \to \mathbb{R}$, these methods seek the parameter vector $x^* \in \mathcal{X}$ that minimizes $f + \phi$ (assuming a minimizer exists):

$$x^* = \arg\min_{x \in \mathcal{X}} f(x) + \phi(x)\,, \tag{5.1}$$

In particular, empirical risk minimization (ERM) methods such as (regularized) least-squares, logistic regression, LASSO, and support vector machines solve an optimization problem of form (5.1); in these cases, $f(x) = \frac{1}{m}\sum_{i=1}^{m} F(x, \xi_i)$ is the average of the loss $F(x, \xi_i)$ of the model parameter x on the given training data $\xi_1, \xi_2, \ldots, \xi_m$ and $\phi(x)$ is a norm (or a combination of norms) on $\mathbb{R}^d$ (e.g., $F(x, \xi) = \log(1 + \exp(x^\top \xi))$ and $\phi(x) = \frac{1}{2}\|x\|_2^2$ in linear logistic regression [30]).

To bring the power of modern parallel computing architectures to such optimization problems, several papers in the past decade have studied parallel variants of the stochastic optimization algorithms applied to these problems. Of particular interest are asynchronous lock-free algorithms, starting with Recht *et al.* [92], who showed that if τ processes run sothcastic gradient descent (SGD) and apply their updates to the same shared iterate without locking, then the overall algorithm (called Hogwild!) converges after the same amount of work (i.e., same number of updates) as serial SGD, up to a multiplicative factor that increases with the number of concurrent processes and decreases with the sparsity of the problem. Thus, if the problem is sparse enough, this penalty can be considered a constant, and the algorithm achieves the same rate, but in $1/\tau$ the time, as serial SGD; this phenomenon is referred to as *linear speed-up* through parallelization. Several follow-up work (see e.g., [10]–[12], [21], [27]–[29], [56], [57], [62], [65], [80], [85], [86], [91], [94]–[96], [102], [106] and the references therein) have demonstrated linear speed-ups for methods based on (block-)coordinate descent (BCD), as well as other variants of SGD such as SVRG [43], SAGA [23], AdaGrad [26], [70], and SGD with a time-decaying step-size. Despite the great advances, however, several problems remain open.[1]

[1] In this paper, we do not further consider BCD-based methods, for two main reasons: a) in general, a

First, the existing convergence guarantees concern SGD when the constraint set $\mathcal{X}$ is box-shaped, that is, a Cartesian product of (block-)coordinate-wise constraints $\mathcal{X} = \times_{i=1}^{d} \mathcal{X}_i$. This leaves it unclear whether existing techniques apply to stochastic optimization algorithms that operate on non-box-shaped constraints (such as projected SGD working on the ℓ_2 ball), or algorithms that use a non-Euclidean regularizer, such as the Exponentiated Gradient (EG) algorithm working on the probability simplex (see, e.g., [38], [97]).

Second, with the exception of the works of Duchi *et al.* [27] and Pan *et al.* [84] (which still require box-shaped constraints), the existing analyses demonstrating the fast speed-ups are limited to strongly-convex (or Polyak-Łojasiewicz) objectives. Thus, so far it had remained unclear whether a similar speed-up analysis is possible if the objective is simply convex or smooth [65], or if we are in the closely-related online-learning setting where the objective can change over time.

Third, with the exception of the work of Pedregosa *et al.* [85] (which still requires box-shaped constraints, block-separable ϕ and strongly-convex f), the existing analyses do not take advantage of the structure of problem (5.1). In particular, when ϕ is "simple to optimize" over $\mathcal{X}$ (formally defined as having access to a proximal operator oracle, as we make precise in the sequel), serial algorithms such as Proximal-SGD take advantage of this property to achieve considerably faster convergence rates. Asynchronous variants of the Proximal-SGD algorithm with such faster rates have so far been unavailable for non-strongly-convex objectives and non-box constraints.

5.1.1 Contributions

In this paper we address the aforementioned problems and present algorithms that are applicable to general convex constraint sets, not just box-shaped $\mathcal{X}$, but still achieve linear speed-ups (under sparsity) for non-smooth and non-strongly-convex (as well as

BCD update may unnecessarily slow down the convergence of the algorithm by focusing only on a single coordinate of the gradient information, especially in the sparse-data problems we consider in this paper (see, e.g., Pedregosa *et al.* [85, Appendix F]); and b) BCD algorithms typically apply only to box-shaped constraints, which is what our algorithms are designed to be able to avoid. We would like to note, however, that our stochastic gradient oracle set-up (Section 5.2) does allow for building an unbiased gradient estimate using only one randomly-selected (block-)coordinate, as done in BCD methods. Nevertheless, the literature on parallel asynchronous BCD algorithms is vast, including especially algorithms for proximal, non-strongly-convex, and non-convex optimization; see, e.g., [10]–[12], [21], [28], [29], [62], [86], [91], [94]–[96], [102], [106] and the references therein.

Algorithm	$\mathcal{X}$	Nonsmooth	Smooth f	Strongly-convex	Smooth f + Strongly-convex
SGD / DA	$\mathbb{R}^d$	[27], [84] ✓	[84] ✓	[84] ✓	[57], [65], [80], [84], [92] ✓
SGD	□	[27], [84]	[84]	[84]	[57], [65], [80], [84], [92]
DA	◯	✓	✓	✓	✓
AG / DA	□	[27], [84] ✓	[84] ✓	[84] ✓	[84] ✓
AG / DA	◯	✓	✓	✓	✓
Prox-DA	◯	✓	✓	✓	✓
Prox-AG	◯	✓	✓	✓	✓
Hedge/EG	△	✓	✓	✓	✓

Table 5.1: Convex / star-convex optimization settings under which sufficient sparsity results in linear speed-up. Previous work appear under the algorithm / problem combinations they cover. A ✓ indicates a setting covered by the results in this paper. The symbols □, △, and ◯ indicate, respectively, the case when the constraint set is box-shaped, the probability simplex, or any convex constraint set with a projection oracle. AG stands for AdaGrad and DA for dual-averaging, while Prox-AG and Prox-DA denote the variant of the algorithm using the proximal operator of ϕ.

smooth or strongly convex) objectives, and even a specific class of non-convex problems. This is achieved through our new asynchronous optimization algorithm, AsynCADA, which generalizes the Async-AdaGrad (and Async-DA) algorithm of Duchi *et al.* [27] to proximal updates and its data-dependent bound to arbitrary constraint sets. Instantiations of AsynCADA under different settings are given in Table 5.1. Indeed, the results are obtained by a more general analysis framework, built on the work of Duchi *et al.* [27], that yields data-dependent convergence guarantees for a generic class of adaptive, composite-objective online optimization algorithms undergoing perturbations to their "state". We further use this framework to derive the first asynchronous online and stochastic optimization algorithm with non-box constraints that uses non-Euclidean regularizers. In particular, we present and analyze HedgeHog, the parallel asynchronous variant of the EG algorithm, also known as Hedge in online linear optimization [38], [97], and show that it enjoys similar parallel speed-up regimes as AsynCADA. The results are derived for the more general setting of noisy online optimization, and the generic framework is of independent interest, in particular in the related settings of distributed and delayed-feedback learning.

5.1.2 Notation and definitions

We use τ_* to denote a global upper bound on the number of concurrent updates in AsynCADA and HedgeHog, such that each concurrent iteration of the algorithms can overlap in time with at most τ_* other iterations. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$, $\mathbb{I}\{\mathcal{E}\}$ for the indicator of an event $\mathcal{E}$, and $\sigma(\mathcal{H})$ to denote the sigma-field generated by a set $\mathcal{H}$ of random variables. The j-th coordinate of a vector $a \in \mathbb{R}^d$ is denoted $a^{(j)}$. For $\alpha \in \mathbb{R}^d$ with positive entries, $\|\cdot\|_\alpha$ denotes the α-weighted Euclidean norm, given by $\|x\|_\alpha^2 = \frac{1}{2}\sum_{j=1}^{d} \alpha^{(j)} \left(x^{(j)}\right)^2$, and $\|\cdot\|_{\alpha,*}$ its dual. We use $(a_t)_{t=i}^{j}$ to denote a sequence $a_i, a_{i+1}, \ldots, a_j$ and $a_{i:j}$ to denote its sum $\sum_{t=i}^{j} a_t$, with $a_{i:j} := 0$ if $i > j$. Given a differentiable function $h : \mathbb{R}^d \to \mathbb{R}$, the *Bregman divergence* in h of $y \in \mathbb{R}^d$ from $x \in \mathbb{R}^d$ is given by

$$\mathcal{B}_h(y, x) := h(y) - h(x) - \langle \nabla h(x), y - x \rangle .$$

It can be shown that a differentiable function is convex if and only if $\mathcal{B}_h(x, y) \geq 0$ for all $x, y \in \mathbb{R}^d$. The function $h : \mathbb{R}^d \to \mathbb{R}$ is *μ-strongly convex* w.r.t. a norm $\|.\|$ on $\mathbb{R}^d$ if and only if for all $x, y \in \mathbb{R}^d$ $\mathcal{B}_h(x, y) \geq \frac{\mu}{2}\|x - y\|^2$, and *smooth* w.r.t. a norm $\|.\|$ if and only if for all $x, y \in \mathbb{R}^d$, $|\mathcal{B}_h(x, y)| \leq \frac{1}{2}\|x - y\|^2$. A differentiable function f is *star-convex* if and only if there exists a global minimizer x^* of f such that for all $x \in \mathbb{R}^d$, $\mathcal{B}_f(x^*, x) \geq 0$.

5.2 Problem setting: noisy online optimization

We consider a generic iterative optimization setting that enables us to study both online learning and stochastic composite optimization. The problem consists of a (known) constraint set $\mathcal{X}$ and a (known) convex (possibly non-differentiable) function ϕ, as well as differentiable functions $f_1, f_2, \ldots$ about which an algorithm learns iteratively. At each iteration $t = 1, 2, \ldots$, the algorithm picks an iterate $x_t \in \mathcal{X}$, and observes an unbiased estimate $g_t \in \mathbb{R}^d$ of the gradient $\nabla f_t(x_t)$: $\mathbb{E}\{g_t | x_t\} = \nabla f_t(x_t)$. The goal is to minimize the composite-objective online regret after T iterations, given by

$$R_T^{(f+\phi)} = \sum_{t=1}^{T} \left(f_t(x_t) + \phi(x_t) - f(x_T^*) - \phi(x_T^*)\right) ,$$

where $x_T^* = \arg\min_{x \in \mathcal{X}} \left\{\sum_{t=1}^{T} (f_t(x) + \phi(x))\right\}$. In the absence of noise (i.e., when $g_t = \nabla f_t(x_t)$), this reduces to the (composite-objective) online (convex) optimization setting

[38], [97].

Stochastic optimization, online regret, and iterate averaging. If $f_t = f$ for all $t = 1, 2, \dots$, we recover the stochastic optimization setting, with the algorithm aiming to minimize the composite objective $f + \phi$ over $\mathcal{X}$ while receiving noisy estimates of $\nabla f(x)$. The algorithm's online regret can then be used to control the optimization risk: Since $f_t \equiv f$, we have $x^*_T = x^* = \arg\min_{x \in \mathcal{X}} \{f(x) + \phi(x)\}$, and by Jensen's inequality, if f is convex and $\bar{x}_T = \frac{1}{T} x_{1:T}$ is the average iterate,

$$f(\bar{x}_T) + \phi(\bar{x}_T) - f(x^*) - \phi(x^*) \le \frac{1}{T} R_T^{(f+\phi)} .$$

In addition, if f is non-convex but $\bar{x}_T$ is selected uniformly at random from $x_1, \dots, x_T$, then the above bound holds in expectation. As such, in the rest of the paper we study the optimization risk through the lens of online regret.

Stochastic first-order oracle. Throughout the paper, we assume that at time t, the noisy gradient estimate g_t is given by a randomized first-order oracle[2] $g_t : \mathbb{R}^d \times \Xi \to \mathbb{R}^d$, where Ξ is some space of random variables, and there exists a sequence $(\xi_t)_{t=1}^T$ of independent elements from Ξ, with distribution $\mathbb{P}_\Xi$, such that $\int_\Xi g_t(x, \xi) d\mathbb{P}_\Xi(\xi) = \nabla f_t(x)$ for all $x \in \mathcal{X}$.

For example, in the finite-sum stochastic optimization case when $f = \sum_i^N f_i$, selecting one f_i uniformly at random to estimate the gradient corresponds to $\mathbb{P}_\Xi$ being the uniform distribution on $\Xi = \{1, 2, \dots, N\}$ and $g_t(x, \xi_t) = \nabla f_{\xi_t}(x)$, whereas selecting a mini-batch of f_i's correponds to Ξ being the set of subsets (of a fixed or varying size) of $\{1, 2, \dots, N\}$ and $g_t(x, \xi_t) = \frac{1}{|\xi_t|} \sum_{i \in \xi_t} \nabla f_i(x)$. This also covers variance-reduced gradient estimates as formed, e.g., by SAGA and SVRG, in which case g_t is built using information from the previous rounds.[3]

[2]With a slight abuse of notation, $g_t(x, \xi)$ (with arguments x, ξ) is from now on used to denote the oracle at time t evaluated at x, ξ, where as g_t (without arguments) denotes the observed noisy gradient $g_t(x_t, \xi_t)$.

[3] Note that in this case ξ_t remains an independent sequence, even though g_t changes with the history.

5.3 AsynCADA: Asynchronous Composite Adaptive Dual Averaging

In this section, we introduce and analyze AsynCADA for asynchronous noisy online optimization. AsynCADA consists of τ processes running in parallel (e.g., threads on the same physical machine or computing nodes distributed over a network accessing a shared data store). The processes can access a shared memory, consisting of a *dual* vector $z \in \mathbb{R}^d$ to store the sum of observed gradient estimates g_t, a *step-size* vector $\eta \in \mathbb{R}^d$, and an integer t, referred to as the *clock*, to track the number of iterations completed at each point in time. The processes run copies of Algorithm 4 concurrently.

Algorithm 4: AsynCADA: Asynchronous Composite Adaptive Dual Averaging

```
1  repeat
2      $\hat{\eta} \leftarrow$ a full (lock-free) read of the shared step-sizes $\eta$
3      $\hat{z} \leftarrow$ a full (lock-free) read of the shared dual vector $z$
4      $t \leftarrow t + 1$                                   // atomic read-increment
5      $\hat{t} \leftarrow t + \gamma$                        // denote $\hat{z}_{t-1} = \hat{z}$, $\hat{\eta}_t = \hat{\eta}, \hat{t}_t = \hat{t}$
6      Receive $\xi_t$
7      Compute the next iterate: $x_t \leftarrow \mathbf{prox}(\hat{t}_t\phi, -\hat{z}_{t-1}, \hat{\eta}_t)$   // prox defined in (5.2)
8      Obtain the noisy gradient estimate: $g_t \leftarrow g_t(x_t, \xi_t)$
9      for $j$ such that $g_t^{(j)} \neq 0$ do $z^{(j)} \leftarrow z^{(j)} + g_t^{(j)}$   // atomic update
10     Update the shared step-size vector $\eta$
11 until terminated
```

Inconsistent reads. The processes access the shared memory without necessarily acquiring a lock: as in previous Hogwild!-style algorithms [56], [57], [65], [85], [92], we only assume that operations on single coordinates of z and η, as well as on t', are atomic. This in particular means that the values of $\hat{z}$ or $\hat{\eta}$ read by a process may not correspond to an actual state of z or η at any given point in time, as different processes can modify the coordinates in parallel while the read is taking place.

Proximal operator oracle. We assume that there exists a generalized *proximal operator* oracle **prox** over $\mathcal{X}$, invoked in Line 7 of Algorithm 4, that given a function ψ and vectors z and η, returns

$$\mathbf{prox}(\psi, z, \eta) := \arg\min_{x \in \mathcal{X}} \psi(x) + \frac{1}{2} \left\| x - \eta^{-1} \odot z \right\|_{\eta}^{2}, \tag{5.2}$$

where η^{-1} denotes the element-wise inverse of η and $\odot$ denotes element-wise multiplication. When η is the same for all coordinates (in which case we simply treat it as a scalar), this reduces to $\mathbf{prox}_\psi(z, \eta) = \arg\min_{x\in\mathcal{X}} \psi(x) + \frac{\eta}{2}\|x - z/\eta\|^2$, which is the standard proximal operator; the generalized version (5.2) makes it possible to use coordinate-wise step-sizes as in AdaGrad [26], [70]. Finally, at iteration t AsynCADA invokes **prox** with $\psi \leftarrow (t+\gamma)\phi$. Similarly, serial proximal DA algorithms [109] call **prox** with $\psi \leftarrow t\phi$, whereas the conventional Proximal-SGD algorithm (based on Mirror-Descent) invokes the proximal operator with $\psi \leftarrow \phi$ irrespective of the iteration; see the paper of Xiao [109, Sections 5 and 6] for a detailed discussion of this phenomenon.

The role of γ. AsynCADA uses an over-estimates $\hat{t}_t$ of the current global clock t by an additional γ. This over-estimation enables us to better handle the effect of asynchrony when composite objectives are involved; see Section 5.3.1. AsynCADA can nevertheless be run without γ (i.e., with $\gamma = 0$).[4]

Exact vs estimated clock. AsynCADA as given in Algorithm 4 maintains the global clock t exactly. However, this option may not be desirable (or available) in certain asynchronous computing scenarios. For example, if the processes are distributed over a network, then maintaining an exact global clock amounts to changing the pattern of asynchrony and delaying the computations by repeated calls over a network. To mitigate this requirement, in Section 5.A we provide AsynCADA(ρ), a version of AsynCADA in which the processes update the global clock only every ρ iterations. AsynCADA as presented in Algorithm 4 is equivalent to AsynCADA(ρ) with $\rho = 1$, and both algorithms enjoy the same rate of convergence and linear speed-up.

Obviously, when $\phi \equiv 0$ and t is not used for setting the step-sizes η either, there is no need to maintain t physically, and Line 4 can be omitted in Algorithm 4.

Updating the step-sizes η: In Line 10 of Algorithm 4, the step-size η has to be updated based on the information received. The exact way this is done depends on the specific

[4] The analysis is still possible, and straightforward, without the bias γ; however, it results in an extra additive term of order $\mathcal{O}(\tau_*^2\Phi)$ where $\Phi = \sup_{x,y\in\mathcal{X}} \{\phi(x) - \phi(y)\}$ is the diameter of $\mathcal{X}$ w.r.t. ϕ. This term does not diminish with p_* and may be unnecessarily large, affecting convergence in the early stages of the optimization process.

step-size schedule. In particular, we consider two situations: First, when the step-size is either constant or a simple function of t (or $\hat{t}_t$ in case of AsynCADA(ρ)), and second, when diagonal AdaGrad step-sizes are used. In the first case, the vector η need not be kept in the shared memory explicitly, and Lines 2 and 10 can be omitted. In the second case, following [27], we store the sum of squared gradients in the shared η, i.e., Line 10 is implemented as follows:

```
10* for $j$ such that $g_t^{(j)} \neq 0$ do  $\left(\eta^{(j)}\right)^2 \leftarrow \left(\eta^{(j)}\right)^2 + \left(g_t^{(j)}\right)^2$        // atomic update
```

Note that in this case, we are storing the square of η in the shared memory, so a square root operation needs to be applied after reading the shared memory in Line 2 to retrieve η.

Forming the output $\bar{x}_T$ for stochastic optimization: For stochastic optimization, the algorithm needs to output the average (or randomized) iterate $\bar{x}_T$ at the end. However, this needs no further coordination between the processes. To form the average iterate, it suffices for each process to keep a local running sum of the iterates it produces and the number of updates it makes. At the end, $\bar{x}_T$ is built from these sums and the total number of updates. Alternatively, we can return a random iterate as $\bar{x}_T$ by terminating the algorithm, with probability $1/T$, after calculating x in Line 7.

5.3.1 Analysis of AsynCADA

To analyze AsynCADA, we assume the sampling of ξ_t in Line 6 is independent of the past:

Assumption 5.1 (Independence of ξ_t)**.** *For all $t = 1, 2, \ldots, T$, the t-th sample ξ_t is independent of the history $\hat{\mathcal{H}}_t := \left\{(\xi_s, \hat{z}_s, \hat{\eta}_{s+1})_{s=1}^{t-1}\right\}$.*

This in turn implies that ξ_t is independent of x_t as well as x_s and ξ_s for all $s < t$.

For general (non-box-shaped) $\mathcal{X}$, Assumption 5.1 is plausible, as AsynCADA *needs* to read z (and η) completely and independently of ξ_t. If $\mathcal{X}$ is box-shaped and ϕ is coordinate-separable, however, the values of $x_t^{(j)}$ for different coordinates j can be calculated independently. In this, case, the algorithm may first sample ξ_t, and then only read the relevant coordinates j from z (and η) for which g_t may be non-zero, as calculating other values of $x_t^{(j)}$ is unnecessary for calculating g_t. As mentioned by Mania *et al.* [65], this violates Assumption 5.1. This is because multiple other processes are updating z and η, and the updates that are included the value read for $\hat{z}_{t-1}$ (and $\hat{\eta}_t$) would then depend on ξ_t. Previous papers either assume that this independence holds in their analysis, e.g., by enforcing a full read of z and η, [56], [57], [65], [85], or rely on the smoothness of the objective to bound the effect of the possible change in the read values [65, Appendix A]. It seems possible to adapt the argument of Mania *et al.* [65, Appendix A] to AsynCADA for box-shaped $\mathcal{X}$, by comparing x_t to the iterate that would have been created based on the content of the shared memory right before the start of the execution of the t-th iteration. This makes the analysis more complicated, and is not necessary when $\mathcal{X}$ is not box-shaped; hence, we do not further pursue this construction in this paper.

Sparsity of the gradient estimates

For $t \in [T]$ and $j \in [d]$, we use $p_{t,j}$ to denote the probability that the j-th coordinate of g_t is non-zero given the history $\hat{\mathcal{H}}_t$, that is,

$$p_{t,j} = \mathbb{P}\left\{ g_t^{(j)} \neq 0 \middle| \hat{\mathcal{H}}_t \right\}.$$

Let p_* to denote an upper-bound on $p_{t,j}$ for all $t \in [T]$ and $j \in [d]$. We use p_* as a measure of the sparsity of the problem.[5]

Non-adaptive and time-decaying step-sizes

We start the convergence analysis of AsynCADA by first studying the case when η_t is either a constant, or varies only as a function of the estimated iteration count $\hat{t}_t$. Recall that each concurrent iteration of the algorithms can overlap in time with at most τ_* other iterations. The next theorem is proved in Section 5.B, where a similar result is also given for AsynCADA(ρ) (Theorem 5.4).

Theorem 5.1. *Suppose that either all $f_t, t \in [T]$ are convex, or $\phi \equiv 0$ and $f_t \equiv f$ for some star-convex function f. Consider* AsynCADA *running under Assumption 5.1 for $T > \tau_*^2$ updates, using $\gamma = 2\tau_*^2$. Let $\eta_0 > 0$. Then:*

[5] In stochastic optimization with a finite-sum objective $f = \sum_{i=1}^m f_i$, where $g_t = \nabla f_{\xi_t}(x_t)$ and $\xi_t \in [m]$ is an index at time t sampled uniformly at random and independently of the history, one could measure the sparsity of the problem through a "conflict graph" [57], [65], [85], [92], which is a bi-partite graph with $f_i, i \in [m]$ on the left and coordinates $j \in [d]$ on the right, and an edge between f_i and coordinate j if $\nabla f_i(x)^{(j)}$ can be non-zero for some $x \in \mathcal{X}$. In this graph, let δ_j denote the degree of the node corresponding to coordinate j and Δ_r be the largest $\delta_j, j \in [d]$. Then, it is straightforward to see that $p_{t,j} \leq \delta_j/m$. Thus, $p_* = \Delta_r/m$ is a valid upper-bound, and gives the sparsity measure used, e.g., by Leblond *et al.* [57] and Pedregosa *et al.* [85].

(i) If $\mathbb{E}\{\|g_t\|_2^2\} \le G_*^2$ for all $t \in [T]$, then using a fixed $\eta_t = \eta_0\sqrt{T}$ or a time-varying $\eta_t = \eta_0\sqrt{\hat{t}_t}$,

$$\frac{1}{T}\mathbb{E}\Big\{R_T^{(f+\phi)}\Big\} \le \frac{1}{\sqrt{T}}\left(\eta_0\|x^*\|_2^2 + \frac{2(1+p_*\tau_*^2)}{\eta_0}G_*^2\right) . \tag{5.3}$$

(ii) If $f = \mathbb{E}_{\xi\sim\mathbb{P}_\Xi}\{F(x,\xi)\}$ and there exists $l \in \mathbb{R}^d$ with positive entries such that for all $\xi \in \Xi$, $F(\cdot,\xi)$ is convex and 1-smooth w.r.t. the norm $\|\cdot\|_l$, then given a constant $c_0 > 8(1+p_*\tau_*^2)$ and using a fixed $\eta_{t,i} = c_0 l_i + \eta_0\sqrt{T}$ or a time-varying $\eta_{t,i} = c_0 l_i + \eta_0\sqrt{\hat{t}_t}$,

$$\frac{1}{T}\mathbb{E}\Big\{R_T^{(f+\phi)}\Big\} \le \frac{c_0\|x^*\|_l^2}{T} + \frac{2}{\sqrt{T}}\left(\eta_0\|x^*\|_2^2 + \frac{4(1+p_*\tau_*^2)}{\eta_0}\sigma_*^2\right) , \tag{5.4}$$

where $\sigma_*^2 = \mathbb{E}\{\|g(x^*,\cdot)\|_2^2\}$.

(iii) If ϕ is μ-strongly-convex and $\mathbb{E}\{\|g_t\|_2^2\} \le G_*^2$ for all $t \in [T]$, then using $\eta_t \equiv 0$ or, equivalently, $\mathbf{prox}(\hat{t}_t\phi, -z, 0) := \arg\min_{x\in\mathcal{X}} \hat{t}_t\phi(x) + \langle z, x\rangle = \nabla\phi^*(-z/\hat{t}_t)$,

$$\frac{1}{T}\mathbb{E}\Big\{R_T^{(f+\phi)}\Big\} \le \frac{(1+p_*\tau_*^2)G_*^2(1+\log(T))}{\mu T} , \tag{5.5}$$

Remark 5.1. *In all cases, the bounds with $\tau_* = 0$ are the same as the bounds in the serial setting. Assuming $c = p_*\tau_*^2$ is constant, the bounds match, up to a constant factor, the serial bounds for adaptive FTRL algorithms under the same settings [47], and we have a theoretical linear speed-up.*

Remark 5.2. *Note that Eq.* (5.5) *holds for all time steps, and converges to zero as T grows, without the knowledge of T or epoch-based updates. In case of* AsynCADA*(ρ), the algorithm does not maintain an exact clock either. To our knowledge, this makes* AsynCADA*(ρ) the first Hogwild!-style algorithm with an any-time guarantee without maintaining a global clock.*

Remark 5.3. *The bound G_* that we are assuming, in the strongly-convex case, is on $(f+\phi) - 1/2\|\cdot\|^2$, i.e., not on the whole objective but rather only on the non-strongly-convex part. As such, like Nguyen* et al. *[80], our result does not rely on the incompatible assumption that the objective is Lipschitz and strongly-convex at the same time. However, unlike Nguyen* et al. *[80],* AsynCADA*(ρ) handles generic $\mathcal{X}$ and does not require an exact global clock.*

AdaGrad step-sizes

In the case of AdaGrad step-size setting (as given by Line 10*), a data-dependent bound can be derived which matches the bound for the serial AdaGrad algorithm [26] up to the same constant factors and linear speed-up criteria as for Theorem 5.1. The derivation depends on using Theorem 5.5 with r_t being the non-perturbed, ideal (diagonal) AdaGrad regularizer and $\hat{r}_t$ being formed using the value of the shared vector η^2 of coordinate-wise squared gradients, while also over-estimating each coordinate of η^2 by an additional $\tau_* G_*^2$ factor. In particular, the only difference to the proof of Theorem 5.1 is bounding the Δ_t/ν_t terms arising from Theorem 5.5. It is straightforward to show that these terms enjoy a bound of the same order as the rest of the regret, and do not change the rate of convergence. The proof is very similar to the derivations done by Duchi *et al.* [27], in particular using the inequality $\sqrt{a+b} - \sqrt{a} \le b/(2\sqrt{a})$ to bound Δ_t, which we do not repeat here.

5.4 HedgeHog: Hogwild-Style Hedge

Next, we present HedgeHog, the first asynchronous version of the EG algorithm. The parallelization scheme is very similar to AsynCADA, the difference being that EG uses multiplicative updates rather than additive SGD-style updates. We focus only on the case of $\phi \equiv 0$. Each processes runs Lines 3–10 of Algorithm 5 concurrently with the other processes, sharing the dual vector z.

Algorithm 5: HedgeHog!: Asynchronous Stochastic Exponentiated Gradient.

```
   Input: Step size $\eta$
1  Initialization
2      Let $z \leftarrow 0$ be the shared sum of observed gradient estimates
3  repeat in parallel by each process
4      $\hat{z} \leftarrow$ a full lock-free read of the shared dual vector $z$        // $t \leftarrow t+1$, denote $\hat{z}_{t-1} = \hat{z}$
5      Receive $\xi_t$
6      Compute the next iterate: $w_t^{(i)} \leftarrow \exp\left(-\hat{z}_{t-1}^{(i)}/\eta\right)$,     $i = 1, 2, \ldots, d$
7      Normalize: $x_t \leftarrow w_t/\|w_t\|_1$
8      Obtain the noisy gradient estimate: $g_t \leftarrow g_t(x_t, \xi_t)$
9      for $j$ such that $g_t^{(j)} \neq 0$ do $z^{(j)} \leftarrow z^{(j)} + g_t^{(j)}$        // atomic update
10 until terminated
```

As in AsynCADA(ρ), we index the iterations by the time they finish the reading of z in Line 4 of HedgeHog ("after-read" labeling [56]). Similarly, we use $\hat{\mathcal{H}}_t = \{(\xi_s, \hat{z}_s)_{s=1}^{t-1}\}$ to denote the history of HedgeHog at time t, and use $\hat{\mathcal{H}}_t$ to define the sparsity measure p_* as in Section 5.3.1. Then, we have the following regret bound for HedgeHog.

Theorem 5.2. *Let $\mathcal{X}$ be the probability simplex $\mathcal{X} = \{x | x^{(j)} > 0, \|x\|_1 = 1\}$, and suppose that either f_t are all convex, or $f_t \equiv f$ for a star-convex f. Assume that for all $t \in [T]$, the sampling of ξ_t in Line 5 of* HedgeHog *is independent of the history $\hat{\mathcal{H}}_t$. Then, after T updates,* HedgeHog *satisfies*

$$\mathbb{E}\Big\{R_T^{(f)}\Big\} \leq \eta \log(d) + \sum_{t=1}^{T} \mathbb{E}\Big\{\frac{1 + \sqrt{p_*}\tau_*}{2\eta} \|g_t\|_\infty^2\Big\}.$$

Remark 5.4. *As in the case of* AsynCADA*, as long as $\sqrt{p_*}\tau_*$ is a constant, the rate above matches the worst-case rate of serial EG up to constant factors, implying a linear speed-up. In particular, given an upper-bound G_* on $\mathbb{E}\{\|g_t\|_\infty\}$ and setting $\eta = G_*/\sqrt{T\log(d)}$, we recover the well-known $\mathcal{O}(G_*\sqrt{T\log(d)})$ rate for EG [38], but in the parallel asynchronous setting.*

5.5 The framework: serial optimization with perturbed state

In this section, we present the generic framework underlying the analysis of AsynCADA and HedgeHog. The framework allows us to study the effect of perturbations in the state of a (serial) dual-averaging algorithm on its regret.

Serial algorithm. We focus on a family of algorithms, known as Adaptive Follow-the-Regularized-Leader (Ada-FTRL) [47], [69], [82], that generalize regularized dual-averaging algorithms [109]. The Ada-FTRL algorithm uses a sequence of regularizer functions $r_0, r_1, r_2, \ldots$. At time $t = 1, 2, \ldots$, given the previous feedback $g_s \in \mathbb{R}^d, s \in [t-1]$, Ada-FTRL selects the next point x_t such that

$$x_t \in \operatorname*{arg\,min}_{x\in\mathcal{X}} \langle z_{t-1}, x\rangle + t\phi(x) + r_{0:t-1}(x)\,, \tag{5.6}$$

where $z_{t-1} = g_{1:t-1}$ is the sum of the past feedback. We refer to $(z_t, t, r_{0:t})$ as the *state* of the algorithm at time t, noting that apart from tie-breaking in (5.6), this state determines x_t.

It is straightforward to verify that if $\eta_t^{(i)}, i \in [d]$ are positive step-sizes (possibly adaptively tuned as in AdaGrad [26], [70]), then Ada-FTRL with $r_{0:t-1} = \frac{1}{2}\|\cdot\|_{\eta_t}^2$ reduces to $x_t = \mathbf{prox}(t\phi, -z_{t-1}, \eta_t)$, with $\mathbf{prox}$ given by (5.2). In addition, when $\phi = 0$, $\mathcal{X}$ is the probability simplex, and $\eta > 0$, Ada-FTRL with the negentropy regularizer $r_{0:t-1}(x) = r_0(x) = \eta \sum_{i=1}^{d} x_i \log(x_i)$ recovers the update $x_t^{(i)} = C_t \exp(-z_{t-1}^{(i)}/\eta)$ of the EG algorithm, where $C_t = 1/\sum_{j=1} \exp(-z_{t-1}^{(j)}/\eta)$ is the constant normalizing x_t to lie in $\mathcal{X}$. Other choices of r_t recover algorithms such as the p-norm update; we refer to Hazan [38], McMahan [69], Orabona *et al.* [82], and Shalev-Shwartz [97] for further examples.

Analysis of Ada-FTRL Ada-FTRL and its special cases have been extensively studied in the literature [16], [38], [47], [69], [82], [97]. In particular, it has been shown that under specific conditions on r_t and ϕ, which we discuss in detail in Appendix 5.E, Ada-FTRL enjoys the following bound.

Theorem 5.3 (Regret of Ada-FTRL). *For any $x^* \in \mathcal{X}$ and any sequence of vectors $(g_t)_{t=1}^T$ in $\mathbb{R}^d$, using any sequence of regularizers $r_0, r_1, \ldots, r_T$ that are* admissible *(Definition 5.2 in Appendix 5.E) w.r.t. a sequence of norms $\|\cdot\|_{(t)}$, the iterates $(x_t)_{t=1}^T$ generated by* Ada-FTRL *satisfy*

$$\sum_{t=1}^{T} \left(\langle g_t, x_t - x^* \rangle + \phi(x_t) - \phi(x^*)\right) \leq r_{0:T}(x^*) - \sum_{t=0}^{T} r_t(x_{t+1}) + \sum_{t=1}^{T} \frac{1}{2}\|g_t\|_{(t,*)}^2 . \tag{5.7}$$

Importantly, this bound holds for *any* feedback sequence g_t irrespective of the way it is generated, and serves as a solid basis to derive bounds under different assumptions on f, ϕ, and r_t [47], [82].

Perturbed Ada-FTRL. Next, we show that Theorem 5.3 also provides the basis to analyze Ada-FTRL with perturbed states. Specifically, suppose that instead of (5.6), the iterate x_t is given by

$$x_t \in \arg\min_{x \in \mathcal{X}} \langle \hat{z}_{t-1}, x \rangle + (\hat{t}_t)\phi(x) + \hat{r}_{0:t-1}(x), \qquad t = 1, 2, \ldots, \tag{5.8}$$

where $\hat{z}_{t-1}$ denotes a *perturbed* version of the dual vector z_{t-1}, $\hat{t}_t$ denotes a perturbed version of Ada-FTRL's counter maintaining the number of iterations t, and $\hat{r}_{0:t-1}$ denotes a perturbed version of the regularizer $r_{0:t-1}$. Then, we can analyze the regret of the Perturbed-ADA-FTRL update (5.8) by comparing x_t to the "ideal" iterate $\tilde{x}_t$, given by

$$\tilde{x}_t := \operatorname*{arg\,min}_{x \in \mathcal{X}} \langle z_{t-1}, x \rangle + t\phi(x) + r_{0:t-1}(x), \qquad t = 1, 2, \ldots. \tag{5.9}$$

Since $(\tilde{x}_t)_{t=1}^T$ is given by a non-perturbed ADA-FTRL update, it enjoys the bound of Theorem 5.3. The crucial observation of Duchi *et al.* [27] (who studied the special case of (5.8) where $\phi = 0$, $\mathcal{X}$ is box-shaped, and $\hat{r}_t = r_t$) was that the regret of Perturbed-ADA-FTRL is related to the linear regret of $\tilde{x}_t$. When ϕ may be non-zero, we capture this relation by the next lemma, proved in Section 5.D:

Lemma 5.1 (Perturbation penalty of ADA-FTRL). *Consider any sequences $(x_t)_{t=1}^T$ and $(\tilde{x}_t)_{t=1}^T$ in $\mathcal{X}$, and any sequence $(g_t)_{t=1}^T$ in $\mathbb{R}^d$. Then, the regret $R_T^{(f+\phi)}$ of the sequence $(x_t)_{t=1}^T$ satisfies*

$$R_T^{(f+\phi)} = \sum_{t=1}^T \left(\langle g_t, \tilde{x}_t - x^* \rangle + \phi(\tilde{x}_t) - \phi(x^*) \right) + \tilde{\epsilon}_{1:T} + \delta_{1:T} - B_{1:T} \,, \tag{5.10}$$

where $\tilde{\epsilon}_t = \langle g_t, x_t - \tilde{x}_t \rangle + \phi(x_t) - \phi(\tilde{x}_t)$, $\delta_t = \langle \nabla f_t(x_t) - g_t, x_t - x^* \rangle$ and $B_t = \mathcal{B}_{f_t}(x^*, x_t)$.

When g_t is an unbiased estimate of $\nabla f_t(x_t)$ and f_t are (star-)convex, the terms $-B_{1:T}$ and $\delta_{1:T}$ are non-positive in expectation (c.f. Theorem 5.5), and for $\tilde{x}_t$ given by (5.9), the first summation is bounded by Theorem 5.3. As such, to bound the regret of Perturbed-ADA-FTRL, it only remains to control the "perturbation penalty" terms $\tilde{\epsilon}_t$ capturing the difference in the composite linear loss $\langle g_t, \cdot \rangle + \phi$ between x_t and $\tilde{x}_t$. In Section 5.D, we use the stability of ADA-FTRL algorithms (Lemma 5.4) to control $\tilde{\epsilon}_{1:T}$, under a specific perturbation structure that captures the evolution of the state of asynchronous dual-averaging algorithms like ASYNCADA and HEDGEHOG. Unlike Duchi *et al.* [27], our derivation applies to *any* convex constraint set $\mathcal{X}$ and, crucially, to ADA-FTRL updates incorporating a non-zero ϕ and a perturbed counter $\hat{t}_t$.

5.6 Conclusion, limitations, and future work

We presented and analyzed ASYNCADA, a parallel asynchronous online optimization algorithm with composite, adaptive updates, and global convergence rates under generic convex constraints and convex composite objectives which can be smooth, non-smooth, or non-strongly-convex. We also showed a similar global convergence for the so-called "star-convex" class of non-convex functions. Under all of the aforementioned settings, we showed that AsynCADA enjoys linear speed-ups when the data is sparse. We also derived and analyzed HedgeHog, the first Hogwild-style asynchronous variant of the Exponentiated Gradient algorithm working on the probability simplex, and showed that HedgeHog enjoyed similar linear speed-ups. To derive and analyze AsynCADA and HedgeHog, we showed that the idea of perturbed iterates, used previously in the analysis of asynchronous SGD algorithms, naturally extends to generic dual-averaging algorithms, in the form of a perturbation in the "state" of the algorithm. Then, building on the previous work of Duchi *et al.* [27], we studied a unified framework for analyzing generic adaptive dual-averaging algorithms for composite-objective noisy online optimization (including AsynCADA and HedgeHog as special cases). Possible directions for future research include applying the analysis to other problem settings, such as multi-armed bandits. In addition, it remains an open problem whether such an analysis is obtainable for constrained adaptive Mirror-Descent without further restrictions on the regularizers r_t (e.g., smoothness of the MD regularizer seems to help). Finally, the derivation of such data-dependent bounds for the final (rather than the average) iterate in stochastic optimization, without the usual strong-convexity and smoothness assumptions, remains an interesting open problem.

Appendices

5.A AsynCADA(ρ): AsynCADA with inexact clock

In this section, we present AsynCADA(ρ), a more general version of AsynCADA that maintains the global clock sparsely. In the context of AsynCADA(ρ), we use t' to denote the *clock* variable in the shared memory, and use t to denote the virtual iteration index as we specify below. The processes run copies of Algorithm 6 concurrently. Each process is also equipped with an internal counter t'' and a function `MaintainClock` to control the updating of the global clock t'.

Similar notes as in AsynCADA apply regarding the maintenance of the step-size η and the formation of the average iterate. Note, however, that unlike AsynCADA, step-sizes changing with time need to use $\hat{t}_t$ rather than t, since the latter is not available anymore. As Theorem 5.4 in Section 5.B shows, this has a negligible effect on the convergence guarantees.

Algorithm 6: AsynCADA(ρ): AsynCADA with inexact clock

```
   Input: clock update frequency $\rho$
1  Initialize internal local counter $t'' \leftarrow 0$
2  repeat
3      $\hat{\eta} \leftarrow$ a full (lock-free) read of the shared step-sizes $\eta$
4      $\hat{z} \leftarrow$ a full (lock-free) read of the shared dual vector $z$
5      $\hat{t} \leftarrow$ MaintainClock()          // $t \leftarrow t+1$, denote $\hat{z}_{t-1} = \hat{z}$, $\hat{\eta}_t = \hat{\eta}, \hat{t}_t = \hat{t}$
6      Receive $\xi_t$
7      Compute the next iterate: $x_t \leftarrow \mathbf{prox}(\hat{t}_t\phi, -\hat{z}_{t-1}, \hat{\eta}_t)$   // prox defined in (5.2)
8      Obtain the noisy gradient estimate: $g_t \leftarrow g_t(x_t, \xi_t)$
9      for $j$ such that $g_t^{(j)} \neq 0$ do $z^{(j)} \leftarrow z^{(j)} + g_t^{(j)}$          // atomic update
10     Update the shared step-size vector $\eta$
11 until terminated
```

Algorithm 7: Maintaining the local and global iteration counters

```
1  Function MaintainClock()
2      Let $\gamma > t''\tau_*$
3      $t'' \leftarrow t'' + 1$               // count number of iterations by this process
4      if $t'' \geq \rho$ then             // update global clock every $\rho$ local iterations
5          $t'' \leftarrow 0$
6          $t' \leftarrow t' + \rho$                          // atomic read-increment
7      end if
8      return $t' + \gamma$    // use the value of $t'$ read in Line 6 if executed; otherwise
        read $t'$
9  end
```

Indexing the iterates Unlike AsynCADA, in AsynCADA(ρ) the iterates are not physically indexed by the global clock t. As such, at each point in time we define a virtual count of the number of iterations undertaken so far, and then come up with an actual estimate of this virtual global clock. To that end, we use the "after-read" iteration indexing proposed by Leblond *et al.* [57]: we define the t-th iteration to be the one corresponding to the t-th completion of reading the shared memory (which happens by reading t' in Line 6 or 8 of `MaintainClock`), before the execution of Line 6. This ensures that $\hat{z}_{t-1}$ contains only updates made by processes $s < t$, which proves useful in the analysis.

Estimating the clock. In AsynCADA(ρ), the processes share share an integer t' to estimate the (virtual) iteration count t, which is updated by each process every ρ iterations. In particular, in each iteration a process makes one call to the function `MaintainClock`

(Algorithm 7), which increments its local counter t'' of the number of updates made by that process since it last updated the global clock; then, after every ρ local updates, `MaintainClock` increments the shared global clock estimate t' by ρ (and resets t'' for that process). Note that in this way, t' is always an under-estimate of t, and AsynCADA (Algorithm 4) is recovered when $\rho = 1$. Again, when $\phi \equiv 0$ and t is not used for setting the step-sizes η either, there is no need to maintain t' physically, and the call to `MaintainClock` can be omitted in Algorithm 6.

5.B Proofs for AsynCADA and AsynCADA(ρ)

We start the analysis by a lemma on the time estimates formed by AsynCADA(ρ).

Lemma 5.2 (Time estimate of AsynCADA(ρ)). *Suppose AsynCADA(ρ) is run for T iterations with any $\rho \geq 1$, using $\gamma \geq \rho\tau_* + \tau_*^2$. Then, the estimated clock $\hat{t}_t$ is non-increasing with t, i.e., for all $s, t \in [T]$ with $s < t$, we have $\hat{t}_s \leq \hat{t}_t$. In addition, for all $t \in [T]$, we have $\hat{t}_t > t + \tau_*^2$.*

Proof. Fix $s < t \in [T]$, and note that the value of t' read in `MaintainClock` in the s-th iteration cannot be greater than the value of t' read in the the t-th iteration. Specifically, t' can only increase over (physical) time, and the iterations are indexed by the time they make their last reading of the shared memory before the update in Line 7 of AsynCADA(ρ), which is the reading (and possibly incrementing) of t' in Line 8 (respectively, Line 6) of `MaintainClock`. Thus, the reading of t' in iteration $s < t$ necessarily has happened before that of t, leading to a smaller value of t'. As all the processes are adding the same fixed value of γ to t' to obtain $\hat{t}$, this implies $\hat{t}_s \leq \hat{t}_t$.

To see that $\hat{t}_t > t$, fix t and let t' be the value of the global clock estimate at the end of the call to `MaintainClock` in the t-th iteration. Then, we have $t' > t - \rho\boldsymbol{\tau} \geq t - \rho\tau_*$ because by construction, there have been at most ρ updates in each of the $\boldsymbol{\tau}$ processors since the last update of t' by each processor, and $\tau_* \geq \boldsymbol{\tau}$ by assumption (since the first $\boldsymbol{\tau}$ iterations overlap with each other in time). As such, $\hat{t}_t = t' + \gamma \geq t' + \rho\tau_* + \tau_*^2 > t + \tau_*^2$. $\square$

Proof of Theorem 5.1. The theorem follows immediately from the convergence bound of AsynCADA(ρ) with $\rho = 1$, given by Theorem 5.4 below. $\square$

To analyze AsynCADA(ρ), we need to make a slightly modified version of Assumption 5.1, since $\hat{t}_t$ is now a non-deterministic part of the state of the algorithm:

Assumption 5.2 (Independence of ξ_t)**.** *For all $t = 1, 2, \ldots, T$, the t-th sample ξ_t is independent of the history $\hat{\mathcal{H}}_t = \left\{ \left(\xi_s, \hat{z}_s, \hat{t}_s, \hat{\eta}_{s+1}\right)_{s=1}^{t-1} \right\}$.*

Theorem 5.4. *Suppose that either all $f_t, t \in [T]$ are convex, or $\phi \equiv 0$ and $f_t \equiv f$ for some star-convex function f. Consider* AsynCADA*(ρ) running under Assumption 5.2 for $T > \tau_*^2$ updates, using $\gamma = 2\tau_*^2$ and any $\rho \leq \tau_*$ in* `MaintainClock`*. Let $\eta_0 > 0$. Then:*

(i) If $\mathbb{E}\{\|g_t\|_2^2\} \leq G_*^2$ for all $t \in [T]$, then using a fixed $\eta_t = \eta_0\sqrt{T}$ or a time-varying $\eta_t = \eta_0\sqrt{\hat{t}_t}$,

$$\frac{1}{T}\mathbb{E}\left\{ R_T^{(f+\phi)} \right\} \leq \frac{1}{\sqrt{T}}\left(\eta_0\|x^*\|_2^2 + \frac{2(1+p_*\tau_*^2)}{\eta_0}G_*^2 \right) . \tag{5.11}$$

(ii) If for all $\xi \in \Xi$, $F(\cdot, \xi)$ is convex and 1-smooth w.r.t. a norm $\|\cdot\|_l$, then given a constant $c_0 > 8(1 + p_*\tau_*^2)$ and using a fixed $\eta_{t,i} = c_0 l_i + \eta_0\sqrt{T}$ or a time-varying $\eta_{t,i} = c_0 l_i + \eta_0\sqrt{\hat{t}_t}$,

$$\frac{1}{T}\mathbb{E}\left\{ R_T^{(f+\phi)} \right\} \leq \frac{c_0\|x^*\|_l^2}{T} + \frac{2}{\sqrt{T}}\left(\eta_0\|x^*\|_2^2 + \frac{4(1+p_*\tau_*^2)}{\eta_0}\sigma_*^2 \right) , \tag{5.12}$$

where $\sigma_*^2 = \mathbb{E}\{\|g(x^*, \cdot)\|_2^2\}$.

(iii) If ϕ is μ-strongly-convex and $\mathbb{E}\{\|g_t\|_2^2\} \leq G_*^2$ for all $t \in [T]$, then using $\eta_t \equiv 0$ or, equivalently, $\mathbf{prox}(\hat{t}_t\phi, -z, 0) := \arg\min_{x\in\mathcal{X}} \hat{t}_t\phi(x) + \langle z, x\rangle = \nabla\phi^*(-z/\hat{t}_t)$,

$$\frac{1}{T}\mathbb{E}\left\{ R_T^{(f+\phi)} \right\} \leq \frac{(1+p_*\tau_*^2)G_*^2(1+\log(T))}{\mu T} , \tag{5.13}$$

Proof. We cast AsynCADA(ρ) in the Perturbed-Ada-FTRL framework of Section 5.5:

(i) Thanks to the after-read time-indexing discussed above, $\hat{z}_t$ in AsynCADA(ρ) cannot include any coordinate updates from g_s for $s > t$ since by construction, the reading of z in t has finished before calculating of g_s is started. As such, $\hat{z}_{t-1}$ and z_{t-1} are related to each other by (5.15) for all $j \in [d]$ and $t \in [T]$.

(ii) In addition, letting $r_{0:t} = \hat{r}_{0:t} = \frac{1}{2}\|\cdot\|_{\eta_t}$, it is easy to see that the AsynCADA(ρ) update $x_t \leftarrow \mathbf{prox}(\hat{t}_t\phi, -\hat{z}_{t-1}, \hat{\eta}_t)$ is equivalent to the perturbed Ada-FTRL update (5.8).

(iii) Furthermore, by Lemma 5.2, $\hat{t}_t$ is non-decreasing with, and greater than, t. Thus, η_t is also non-decreasing with t, and $\hat{t}_t$, $r_{0:t}$ and $\hat{r}_{0:t}$ satisfy Assumption 5.3 with norms $\|\cdot\|_t = \|\cdot\|_{\eta_t}$.

(iv) Finally, Assumption 5.2 ensures that Assumption 5.4 holds.

Therefore, letting $\nu_t = \hat{t}_t - t$, applying Theorem 5.5, and noting that $\Delta_t = 0$ by construction, we get

$$\mathbb{E}\left\{R_T^{(f+\phi)}(x^*)\right\} \leq \mathbb{E}\left\{r_{0:T}(x^*) + \sum_{t=1}^{T} \frac{1 + p_*\nu_t + \sum_{s:t\in O_s} \frac{\tau_s}{\nu_s}}{2} \|g_t\|_{(t,*)}^2 - B_{1:T}\right\}.$$

Next, the assumption that either f_t is convex, or $f_t \equiv f$ for a star-convex f implies $\mathcal{B}_f(x^*, x_t) \geq 0$; hence, the $B_{1:T}$ terms can be dropped. Also, by construction, $\hat{t}_t \leq t + \gamma$, as t' always under-estimates t. Together with Lemma 5.2 and since $\gamma = 2\tau_*^2$, this implies $\tau_*^2 < \nu_t \leq 2\tau_*^2$, and we have

$$\mathbb{E}\left\{R_T^{(f+\phi)}(x^*)\right\} \leq \mathbb{E}\left\{r_{0:T}(x^*) + \sum_{t=1}^{T} \frac{1 + 2p_*\tau_*^2 + 1}{2} \|g_t\|_{(t,*)}^2\right\}. \tag{5.14}$$

Using the definition of G_*, the fact that $r_{0:T} = \frac{\sqrt{\hat{t}_T}}{2}\|\cdot\|^2 \leq \frac{\sqrt{T+\gamma}}{2}\|\cdot\|^2 \leq \frac{\sqrt{3T}}{2}\|\cdot\|^2$, the expansion $\|\cdot\|_{(t,*)}^2 = \frac{1}{\eta_t}\|\cdot\|^2$, and the well-known bound $\sum_{t=1}^{T}(\sqrt{t})^{-1} \leq 2\sqrt{T}$ [69], we get (5.11).

To get (5.12), we continue from (5.14) but instead upper-bound $\|g_t\|$ as follows:

$$\begin{aligned} \frac{1}{2}\|g_t\|_{(t,*)}^2 &\leq \|\nabla F(x_t, \xi_t) - \nabla F(x^*, \xi_t)\|_{(t,*)}^2 + \|\nabla F(x^*, \xi_t)\|_{(t,*)}^2 \\ &\frac{1}{c_0}\|\nabla F(x_t, \xi_t) - \nabla F(x^*, \xi_t)\|_{l,*}^2 + \frac{1}{\eta_0\sqrt{t}}\|\nabla F(x^*, \xi_t)\|^2, \end{aligned}$$

where the last step follows by the definition of $\eta_{t,i}$, which in particular implies $\|\cdot\|_{(t)}^2 \geq c_0\|\cdot\|_l^2$ and $\|\cdot\|_{(t)}^2 \geq \eta_0\sqrt{\hat{t}_t}\|\cdot\|^2 \geq \eta_0\sqrt{t}\|\cdot\|^2$ (similarly, in case of a fixed step size, $\|\cdot\|_{(t)}^2 \geq \eta_0\sqrt{T}\|\cdot\|^2 \geq \eta_0\sqrt{t}\|\cdot\|^2$). Putting back into (5.14), we obtain

$$\begin{aligned} \mathbb{E}\left\{R_T^{(f+\phi)}(x^*)\right\} \leq \mathbb{E}&\left\{\frac{c_0}{2}\|x^*\|_l + \frac{\eta_0\sqrt{3T}}{2}\|x^*\|^2 + \sum_{t=1}^{T} \frac{2 + 2p_*\tau_*^2}{\eta_0\sqrt{t}}\|\nabla F(x^*, \xi_t)\|^2\right\} \\ &+ \mathbb{E}\left\{\sum_{t=1}^{T} \frac{2 + 2p_*\tau_*^2}{c_0}\|\nabla F(x_t, \xi_t) - \nabla F(x^*, \xi_t)\|_{\ell,*}^2\right\} \end{aligned}$$

$$
\begin{aligned}
&\le \mathbb{E}\left\{\frac{c_0}{2}\|x^*\|_l + \frac{\eta_0\sqrt{3T}}{2}\|x^*\|^2 + \sum_{t=1}^{T}\frac{2+2p_*\tau_*^2}{\eta_0\sqrt{t}}\sigma_*^2\right\} \\
&\quad + \mathbb{E}\left\{\sum_{t=1}^{T}\frac{1}{4}\|\nabla F(x_t,\xi_t) - \nabla F(x^*,\xi_t)\|_{l,*}^2\right\} \\
&\le \mathbb{E}\left\{\frac{c_0}{2}\|x^*\|_l + \frac{\eta_0\sqrt{3T}}{2}\|x^*\|^2 + \frac{2(2+2p_*\tau_*^2)}{\eta_0}\sigma_*^2\sqrt{T}\right\} \\
&\quad + \mathbb{E}\left\{\sum_{t=1}^{T}\frac{1}{2}(F(x_t,\xi_t) - F(x^*,\xi_t) - \langle\nabla F(x^*,\xi_t), x_t - x^*\rangle)\right\} \\
&= \mathbb{E}\left\{\frac{c_0}{2}\|x^*\|_l + \frac{\eta_0\sqrt{3T}}{2}\|x^*\|^2 + \frac{4(1+p_*\tau_*^2)}{\eta_0}\sigma_*^2\sqrt{T}\right\} \\
&\quad + \mathbb{E}\left\{\sum_{t=1}^{T}\frac{1}{2}(f(x_t) - f(x^*) - \langle\nabla f(x^*), x_t - x^*\rangle)\right\} \\
&= \mathbb{E}\left\{\frac{c_0}{2}\|x^*\|_l + \frac{\eta_0\sqrt{3T}}{2}\|x^*\|^2 + \frac{4(1+p_*\tau_*^2)}{\eta_0}\sigma_*^2\sqrt{T}\right\} \\
&\quad + \mathbb{E}\left\{\sum_{t=1}^{T}\frac{1}{2}(f(x_t) - f(x^*) - \langle\phi'(x^*), x^* - x_t\rangle)\right\} \\
&= \mathbb{E}\left\{\frac{c_0}{2}\|x^*\|_l + \frac{\eta_0\sqrt{3T}}{2}\|x^*\|^2 + \frac{4(1+p_*\tau_*^2)}{\eta_0}\sigma_*^2\sqrt{T}\right\} \\
&\quad + \mathbb{E}\left\{\sum_{t=1}^{T}\frac{1}{2}(f(x_t) - f(x^*) + \phi(x_t) - \phi(x^*))\right\}.
\end{aligned}
$$

Here, the first inequality follows by the definition of $\|\cdot\|_{(t,*)}$ and the fact that $\eta_T = \eta_0\sqrt{\hat{t}_T} \le \eta_0\sqrt{T+\gamma} \le \eta_0\sqrt{3T}$, the second inequality follows by the definition of c_0 and σ, the third follows by smoothness of F [78] and the bound $\sum_{t=1}^{T}(\sqrt{t})^{-1} \le 2\sqrt{T}$ [69], the fourth by the independence of ξ_t from the history, the fifth by the optimality of x^* (where $\phi'(x^*)$ denotes the sub-gradient of ϕ for which $\phi'(x^*) + \nabla f(x^*) = 0$), and the last line follows by convexity of ϕ. Moving the last term to the l.h.s. and multiplying the sides by 2 completes the proof of (5.12).

To prove (5.13), note that $t\phi$ is $t\mu$-strongly-convex by assumption, and thus the sequence of regularizers $r_t = \hat{r}_t = 0$ still satisfy Assumption 5.3 with the norms $\|\cdot\|_{(t,*)}^2 = \mu t\|\cdot\|^2$. Thus, (5.14) implies

$$
\mathbb{E}\left\{R_T^{(f+\phi)}(x^*)\right\} \le \mathbb{E}\left\{\sum_{t=1}^{T}\frac{2(1+p_*\tau_*^2)}{2\mu t}\|g_t\|^2\right\}
$$

$$\leq \frac{2(1+p_*\tau_*^2)}{2\mu} G_*^2(1+\log(T))\,,$$

where in the last step we have used the bound $\sum_{t=1}^T (1/t) \leq 1 + \log(T)$, completing the proof. □

5.C Proofs for HedgeHog

Proof of Theorem 5.2. As in the case of AsynCADA(ρ), the proof follows by casting HedgeHog as Perturbed-Ada-FTRL. In particular, the same relation between $\hat{z}_{t-1}$ and z_{t-1} holds, and it is easy to see that with the after-read time-indexing the HedgeHog update corresponds to Perturbed-Ada-FTRL with the regularizer $r_{0:t}(x) = r(x) + \eta \ln(d)$ where $r(x) = \eta \sum_{i=1}^d x^{(i)} \log\left(x^{(i)}\right)$, which is 1-strongly-convex w.r.t. the ℓ_1 norm [97]. Note also that we can assume any value for $\hat{t}_t > t$, including $\hat{t}_t = t + \nu_t$ for any $\nu_t > 0$, as we don't use $\hat{t}_t$ in the update and hence don't need to be able to compute it. Then, Assumption 5.3 is satisfied with $\|\cdot\|_t = \sum_{i=1}^d |x^{(i)}|$ being the ℓ_1 norm, with $\|\cdot\|_{(t,*)} = \|\cdot\|_\infty$. Then, applying Theorem 5.5 and noting $\phi = 0, \Delta_t = 0, B_{1:T} \geq 0$, we have

$$\mathbb{E}\left\{R_T^{(f)}(x^*)\right\} \leq r(x^*) + \eta\log(d) + \sum_{t=1}^T \mathbb{E}\left\{\frac{1 + p_*\nu_t + \sum_{s:t\in O_s}\frac{\tau_s}{\nu_s}}{2}\|g_t\|_{(t,*)}^2\right\},$$

for any ν_t determined by $\hat{\mathcal{H}}_t$. In particular, letting $\nu_t = \tau_*/\sqrt{p_*}$, recalling that τ_s and $|\{s : t \in O_s\}|$ cannot be larger than τ_*, and noting that $r(x^*) \leq 0$ for any $x^* \in \mathcal{X}$ completes the proof. □

5.D Proofs for the generic framework

Proof of Lemma 5.1. The proof follows in the same way as in the serial setting [47]. For $t \in [T]$,

$$\begin{aligned} f_t(x_t) - f_t(x^*) &= \langle \nabla f_t(x_t), x_t - x^* \rangle - \mathcal{B}_{f_t}(x^*, x_t) \\ &= \langle g_t, x_t - x^* \rangle + \langle \nabla f_t(x_t) - g_t, x_t - x^* \rangle - \mathcal{B}_{f_t}(x^*, x_t) \\ &= \langle g_t, \tilde{x}_t - x^* \rangle + \langle g_t, x_t - \tilde{x}_t \rangle + \delta_t - B_t \\ &= \langle g_t, \tilde{x}_t - x^* \rangle + \phi(\tilde{x}_t) - \phi(x_t) + \tilde{\epsilon}_t + \delta_t - B_t \end{aligned}$$

Adding $\phi(x_t) - \phi(x^*)$ to both sides and summing over t completes the proof. □

Perturbation structure. We assume that the difference of $\hat{z}_{t-1}$ and z_{t-1} is that zero or more coordinates $g_s^{(j)}$ from the past feedback vectors $g_s, s \in [t-1]$, can be missing from (i.e., not added in) the perturbed dual vector $\hat{z}_{t-1}$. Formally, for all $t \in [T]$ and $j \in [d]$,

$$\hat{z}_{t-1}^{(j)} = g_{1:t-1}^{(j)} - \sum_{s \in O_{t,j}} g_s^{(j)} \,, \tag{5.15}$$

where $O_{t,j}$ is the subset of the past indices $[t-1]$ corresponding to the missing updates at the j-th coordinate. Written in a more compact form,

$$\hat{z}_{t-1} = g_{1:t-1} - \sum_{s \in O_t} I_{t,s} g_s \,, \tag{5.16}$$

where $O_t = \cup_j O_{t,j}$ is the set of all time steps with missing information at time t, and $I_{t,s}, s \in [t-1]$, are diagonal $d \times d$ matrices with $I_{t,s}^{(j,j)} = 1$ if $g_s^{(j)}$ is missing from $\hat{z}_{t-1}$ and 0 otherwise. We define $\tau_{t,j} = |O_{t,j}|$ and $\tau_t = |O_t|$ to denote, respectively, the total number of missing updates to the j-th coordinate of $\hat{z}_{t-1}$, and to the whole vector $\hat{z}_{t-1}$. Similarly, we assume that the time-counter $\hat{t}_t$ may not be equal to t, and the cumulative regularizers $r_{0:t}$ and $\hat{r}_{0:t}$, can be different, with the latter using only some of the past updates made to $r_{0:t}$. However, the exact perturbation in $\hat{t}_t$ and $\hat{r}_{0:t}$ depends on the specifics of the algorithm. Our analysis isolates these perturbations in individual terms, which we can subsequently study on a case-by-case basis. We make the following assumption on $\hat{t}_t$ and the sequence of actual regularizers $(\hat{r}_t)_{t=0}^T$ and ideal regularizers $(r_t)_{t=0}^T$.

Assumption 5.3. *The regularizers $r_t, \hat{r}_t, t = 0, 1, \ldots, T$, are admissible* Ada-FTRL *regularizers (Definition 5.2) with the same sequence of norms $\|\cdot\|_{(t)}$, and the sequence of norms is non-decreasing: $\|\cdot\|_{(t)} \geq \|\cdot\|_{(t-1)}$ for all $t = 1, 2, \ldots, T$. Finally, $r_t \geq 0, t = 0, 1, 2, \ldots, T$, and $\hat{t}_t > t, t = 1, 2, \ldots, T$.*

Intuitively, Assumption 5.3 states that the regularizers $\hat{r}_t$ are not fundamentally different from the regularizers r_t as far as the basic properties of Ada-FTRL are concerned. In particular, the assumption is satisfied if $(r_t)_{t=0}^T$ is admissible with a non-decreasing sequence of norms and the perturbation increases the curvature, that is, $\hat{r}_{0:t-1} - r_{0:t-1}$ is convex. Finally, the assumption $\hat{t}_t > t$ helps us in providing bounds for composite-objective learning, as will become clear later.

Independence assumption. Similarly to the standard serial setting, we will assume that the outcome ξ_t at time t is independent of the history that determines x_t. In the case of perturbed ADA-FTRL, we define the history to depend on the actual states the *perturbed* ADA-FTRL algorithm has gone through:

Definition 5.1 (History of the perturbed game). *For $t = 1, 2, \ldots, T$, the* history of the perturbed game *up to time t is defined as*

$$\hat{\mathcal{H}}_t = \left\{ \left(\xi_s, \hat{z}_s, \hat{t}_s, \hat{r}_{0:s} \right)_{s=1}^{t-1} \right\},$$

where $\hat{z}_s$, $\hat{r}_{0:s}, \hat{t}_s$ are the dual vector, regularizer and time-counter used by the $(s+1)$-th perturbed ADA-FTRL *update.*

We assume that the stochastic outcomes are independent of the history:

Assumption 5.4 (Independence of ξ_t). *For all $t = 1, 2, \ldots, T$, the t-th sample ξ_t is independent of the history $\hat{\mathcal{H}}_t$.*

This in turn means that ξ_t is independent of x_t as well as x_s and ξ_s for all $s < t$.

We call a norm $\|\cdot\|$ a *weighted q-norm* if there exists $q > 0$ and $a_j, j \in [d]$ such that for all $x \in \mathbb{R}^d$,

$$\|x\| = \left(\sum_{j=1}^{d} a_j \left| x^{(j)} \right|^q \right)^{1/q} . \tag{5.17}$$

The next theorem describes a generic data-dependent bound on the regret of perturbed ADA-FTRL. For the theorem, for all $t \in [T]$, we define $\Delta_t := r_{0:t-1}(x_t) - r_{0:t-1}(\tilde{x}_t) + \hat{r}_{0:t-1}(\tilde{x}_t) - \hat{r}_{0:t-1}(x_t)$, and let $\nu_t := \hat{t}_t - t$ with the $\hat{t}_t$ used in the Perturbed-ADA-FTRL update (5.8).

Theorem 5.5. *Suppose that Perturbed-*ADA-FTRL *is run under Assumption 5.4, and Assumption 5.3 holds such that for each $t \in [T]$, $\|\cdot\|_{(t)}$ is a weighted q-norm (Eq. (5.17)) with $q = 1$ or $q = 2$. Then, the regret of Perturbed-*ADA-FTRL *satisfies*

$$\mathbb{E}\left\{ R_T^{(f+\phi)} \right\} \le \mathbb{E}\left\{ r_{0:T}(x^*) + \sum_{t=1}^{T} \left(\frac{1 + p_* \nu_t + \sum_{s:t \in O_s} \frac{\tau_s}{\nu_s}}{2} \|g_t\|_{(t,*)}^2 + \frac{\Delta_t}{\nu_t} \right) - B_{1:T} \right\} .$$

5.D.1 Proof of Theorem 5.5

First, we upper-bound $\tilde{\epsilon}_t$ in terms of the difference between $\tilde{x}_t$ and x_t.

Lemma 5.3. *Consider Perturbed-*ADA-FTRL *under the conditions of Theorem 5.5. Let $\beta_t \in \mathbb{R}^d$ be given by $\beta_t^{(j)} = \mathbb{I}\Big\{g_t^{(j)} \neq 0\Big\}$, and use $\odot$ to denote element-wise vector multiplication. Then,*

- *For any positive real number c_t and any norm $\|\cdot\|$, we have*

$$\tilde{\epsilon}_t + \phi(\tilde{x}_t) - \phi(x_t) \ \le \frac{c_t}{2}\|g_t\|_*^2 + \frac{1}{2c_t}\|\beta_t \odot (x_t - \tilde{x}_t)\|^2 \,,$$

- *In the stochastic setting under Assumption 5.4, for any $c_t > 0$ and any norm $\|\cdot\|$,*

$$\mathbb{E}\{\tilde{\epsilon}_t + \phi(\tilde{x}_t) - \phi(x_t)\} \ \le \mathbb{E}\bigg\{\frac{c_t}{2}\,\|\nabla f_t(x_t)\|_*^2 + \frac{1}{2c_t}\|x_t - \tilde{x}_t\|^2\bigg\} \,.$$

- *Under Assumption 5.4, for any $q \ge 1$, any weighted q-norm $\|\cdot\|$ determined by the history $\hat{\mathcal{H}}_t$, and any positive scalar $c_t \in \sigma(\hat{\mathcal{H}}_t)$,*

$$\mathbb{E}\{\tilde{\epsilon}_t + \phi(\tilde{x}_t) - \phi(x_t)\} \ \le \mathbb{E}\bigg\{\frac{c_t}{2}\|g_t\|_*^2\bigg\} + p_*^{(1/q)}\mathbb{E}\bigg\{\frac{1}{2c_t}\|(x_t - \tilde{x}_t)\|^2\bigg\} \,,$$

  where p_* is a global upper-bound on $\mathbb{P}\Big\{g_t^{(j)} \neq 0|\hat{\mathcal{H}}_t\Big\}$. In case of $q = 2$, the bound still holds if $p_*^{1/2}$ is replaced with p_*.

Proof of Lemma 5.3. To get the first inequality, note that $g_t = \beta_t \odot g_t$ by definition. The bound then follows by the Fenchel-Young inequality.

To get the second bound, note that $x_t, \tilde{x}_t \in \sigma(\hat{\mathcal{H}}_t)$ by construction, so by Assumption 5.4,

$$\mathbb{E}\{\langle g_t - \nabla f_t(x_t), x_t - \tilde{x}_t\rangle\} = \mathbb{E}\Big\{\langle \mathbb{E}\Big\{g_t - \nabla f_t(x_t)|\hat{\mathcal{H}}_t\Big\}, x_t - \tilde{x}_t\rangle\Big\} = 0 \,.$$

Thus, $\mathbb{E}\{\tilde{\epsilon}_t + \phi(\tilde{x}_t) - \phi(x_t)\} = \mathbb{E}\{\langle \nabla f_t(x_t), x_t - \tilde{x}_t\rangle\}$, and the result follows by the Fenchel-Young inequality.

To get the third bound, we first start with the simpler case of $q = 2$, using $a \in \sigma(\hat{\mathcal{H}}_t)$ to denote the associated weighting vector, then apply the first inequality and take expectation of the terms $\|\beta_t \odot (x_t - \tilde{x}_t)\|^2$. Note that by construction, $x_t, \tilde{x}_t \in \sigma(\hat{\mathcal{H}}_t)$. Furthermore, by assumption, $c_t, a \in \sigma(\hat{\mathcal{H}}_t)$. Hence,

$$\mathbb{E}\bigg\{\frac{1}{2c_t}\|\beta_t \odot (x_t - \tilde{x}_t)\|^2\bigg\} \ = \mathbb{E}\bigg\{\mathbb{E}\bigg\{\frac{1}{2c_t}\|\beta_t \odot (x_t - \tilde{x}_t)\|^2|\hat{\mathcal{H}}_t\bigg\}\bigg\}$$

$$
\begin{aligned}
&= \mathbb{E}\left\{\sum_{j=1}^{d} \mathbb{E}\left\{\frac{1}{2c_t} a^{(j)} \beta_t^{(j)} (x_t^{(j)} - \tilde{x}_t^{(j)})^2 | \hat{\mathcal{H}}_t\right\}\right\} \\
&= \mathbb{E}\left\{\sum_{j=1}^{d} \mathbb{E}\left\{\mathbb{I}\left\{g_t^{(j)} \neq 0\right\} | \hat{\mathcal{H}}_t\right\} \frac{1}{2c_t} a^{(j)} (x_t^{(j)} - \tilde{x}_t^{(j)})^2\right\} \\
&= \mathbb{E}\left\{\sum_{j=1}^{d} p_{t,j} \frac{1}{2c_t} a^{(j)} (x_t^{(j)} - \tilde{x}_t^{(j)})^2\right\} \\
&\leq \left(\max_{j \in [d]} p_{t,j}\right) \mathbb{E}\left\{\frac{1}{2c_t} \sum_{j=1}^{d} a^{(j)} (x_t^{(j)} - \tilde{x}_t^{(j)})^2\right\},
\end{aligned}
$$

completing the proof.

To get the bound for any $q \geq 1$, first note that when $q \in [1, \infty)$, the function $h : [0, \infty) \to \mathbb{R}$ given by $h(x) := x^{1/q}$ (with $h(0) := 0$) is concave for all $x > 0$. Thus, by Jensen's inequality, $\mathbb{E}\{h(X)\} \leq h(\mathbb{E}\{X\})$ for any non-negative random variable X. Next, we let the q-norm in question be given by Eq. (5.17), with $a \in \sigma(\hat{\mathcal{H}}_t)$ denoting the associated weighting vector, and continue as in the case of $q = 2$ above:

$$
\begin{aligned}
\mathbb{E}\left\{\frac{1}{2c_t} \|\beta_t \odot (x_t - \tilde{x}_t)\|^2\right\} &= \mathbb{E}\left\{\mathbb{E}\left\{\frac{1}{2c_t} \|\beta_t \odot (x_t - \tilde{x}_t)\|^2 \middle| \hat{\mathcal{H}}_t\right\}\right\} \\
&= \mathbb{E}\left\{\frac{1}{2c_t} \mathbb{E}\left\{\left(\sum_{j=1}^{d} a^{(j)} \beta_t^{(j)} \left|x_t^{(j)} - \tilde{x}_t^{(j)}\right|^q\right)^{2/q} \middle| \hat{\mathcal{H}}_t\right\}\right\} \\
&\leq \mathbb{E}\left\{\frac{1}{2c_t} \left(\mathbb{E}\left\{\left(\sum_{j=1}^{d} a^{(j)} \beta_t^{(j)} \left|x_t^{(j)} - \tilde{x}_t^{(j)}\right|^q\right)^2 \middle| \hat{\mathcal{H}}_t\right\}\right)^{1/q}\right\},
\end{aligned}
$$

where the last inequality follows since $\mathbb{E}\left\{h(X) | \hat{\mathcal{H}}_t\right\} \leq h\left(\mathbb{E}\left\{X | \hat{\mathcal{H}}_t\right\}\right)$ by the concavity of h as argued above, where $X = \left(\sum_{j=1}^{d} a^{(j)} \beta_t^{(j)} \left|x_t^{(j)} - \tilde{x}_t^{(j)}\right|^q\right)^2$. On the other hand, since h is also increasing, we can bound $h\left(\mathbb{E}\left\{X | \hat{\mathcal{H}}_t\right\}\right)$ by first upper-bounding $\mathbb{E}\left\{X | \hat{\mathcal{H}}_t\right\}$. In particular,

$$
\begin{aligned}
\mathbb{E}\left\{X | \hat{\mathcal{H}}_t\right\} &= \mathbb{E}\left\{\left(\sum_{j=1}^{d} a^{(j)} \beta_t^{(j)} \left|x_t^{(j)} - \tilde{x}_t^{(j)}\right|^q\right)^2 \middle| \hat{\mathcal{H}}_t\right\} \\
&= \mathbb{E}\left\{\sum_{j=1}^{d} a^{(j)} \beta_t^{(j)} \left|x_t^{(j)} - \tilde{x}_t^{(j)}\right|^q \left(\sum_{i=1}^{d} a^{(i)} \beta_t^{(i)} \left|x_t^{(i)} - \tilde{x}_t^{(i)}\right|^q\right) \middle| \hat{\mathcal{H}}_t\right\} \\
&\leq \mathbb{E}\left\{\sum_{j=1}^{d} a^{(j)} \beta_t^{(j)} \left|x_t^{(j)} - \tilde{x}_t^{(j)}\right|^q \left(\sum_{i=1}^{d} a^{(i)} \left|x_t^{(i)} - \tilde{x}_t^{(i)}\right|^q\right) \middle| \hat{\mathcal{H}}_t\right\}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{j=1}^{d} a^{(j)} \mathbb{E}\Big\{ \beta_t^{(j)} | \hat{\mathcal{H}}_t \Big\} \left| x_t^{(j)} - \tilde{x}_t^{(j)} \right|^q \left(\sum_{i=1}^{d} a^{(i)} \left| x_t^{(i)} - \tilde{x}_t^{(i)} \right|^q \right) \\
&\le p_* \sum_{j=1}^{d} a^{(j)} \left| x_t^{(j)} - \tilde{x}_t^{(j)} \right|^q \left(\sum_{i=1}^{d} a^{(i)} \left| x_t^{(i)} - \tilde{x}_t^{(i)} \right|^q \right) \\
&= p_* \|x_t - \tilde{x}_t\|^{2q} .
\end{aligned}
$$

Thus, $h\left(\mathbb{E}\Big\{X|\hat{\mathcal{H}}_t\Big\}\right) \le h\left(p_* \|x_t - \tilde{x}_t\|^{2q}\right) = p_*^{1/q} \|x_t - \tilde{x}_t\|^2$. Thus,

$$
\begin{aligned}
\mathbb{E}\left\{ \frac{1}{2c_t} \|\beta_t \odot (x_t - \tilde{x}_t)\|^2 \right\} &\le \mathbb{E}\left\{ \frac{1}{2c_t} h\left(\mathbb{E}\Big\{X|\hat{\mathcal{H}}_t\Big\}\right) \right\} \\
&\le \mathbb{E}\left\{ \frac{1}{2c_t} p_*^{1/q} \|x_t - \tilde{x}_t\|^2 \right\} ,
\end{aligned}
$$

completing the proof of the third bound. □

Thus, controlling the regret in perturbed optimization reduces to picking a suitable norm $\|\cdot\|$ and applying Lemma 5.3 at each time step t, and then controlling the differences $x_t - \tilde{x}_t$. To that end, we use the stability of Ada-FTRL updates, that is, that the difference of two Ada-FTRL iterates is controlled by the difference in the two states of the algorithm resulting in the iterates. The following lemma provides this stability bound.

Lemma 5.4. *Let $(x_t)_{t=1}^T$ and $(\tilde{x}_t)_{t=1}^T$ be given by updates* (5.8) *and* (5.9)*, respectively, and suppose that Assumption 5.3 holds. Define $\Delta_t = r_{0:t-1}(x_t) - r_{0:t-1}(\tilde{x}_t) + \hat{r}_{0:t-1}(\tilde{x}_t) - \hat{r}_{0:t-1}(x_t)$ for $t = 1, 2, \ldots, T$. Then, for all $t = 1, 2, \ldots, T+1$,*

$$
\frac{1}{2}\|x_t - \tilde{x}_t\|_{(t)}^2 \le \frac{1}{2} \left\| \sum_{s \in O_t} I_{t,s} g_s \right\|_{(t,*)}^2 + (t - \hat{t}_t)\left(\phi(x_t) - \phi(\tilde{x}_t)\right) + \Delta_t . \qquad (5.18)
$$

Proof of Lemma 5.4. Since both $(r_t)_{t=1}^T$ and $(\hat{r}_t)_{t=1}^T$ are admissible, the Ada-FTRL margin lemma [47, Lemma 24 (Appendix F)] applied to the update (5.8) implies that for all $t = 1, 2, \ldots, T$,

$$
\langle \hat{z}_{t-1}, \tilde{x}_t - x_t \rangle + \hat{t}\left(\phi(\tilde{x}_t) - \phi(x_t)\right) + \hat{r}_{0:t-1}(\tilde{x}_t) - \hat{r}_{0:t-1}(x_t) \ge \mathcal{B}_{\hat{t}\phi + \hat{r}_{0:t-1}}(\tilde{x}_t, x_t) ,
$$

while for update (5.9) we have

$$
\langle z_{t-1}, x_t - \tilde{x}_t \rangle + t\left(\phi(x_t) - \phi(\tilde{x}_t)\right) + r_{0:t-1}(x_t) - r_{0:t-1}(\tilde{x}_t) \ge \mathcal{B}_{t\phi + r_{0:t-1}}(x_t, \tilde{x}_t) .
$$

By the strong convexity of $t\phi + r_{0:t-1}$ and $t\phi + \hat{r}_{0:t-1}$ w.r.t. $\|\cdot\|_{(t)}$, convexity of ϕ, and the fact that $\hat{t} > t > 0$ (so that $\hat{t}_t\phi + r_{0:t-1}$ is also strongly-convex w.r.t. $\|\cdot\|_{(t)}$), we have $\mathcal{B}_{\hat{t}\phi+\hat{r}_{0:t-1}}(\tilde{x}_t, x_t) \geq \frac{1}{2}\|x_t - \tilde{x}_t\|_{(t)}^2$ and $\mathcal{B}_{t\phi+r_{0:t-1}}(x_t, \tilde{x}_t) \geq \frac{1}{2}\|x_t - \tilde{x}_t\|_{(t)}^2$. Adding the above,

$$\begin{aligned}
\frac{1}{2}\|x_t - \tilde{x}_t\|_{(t)}^2 &\leq -\frac{1}{2}\|x_t - \tilde{x}_t\|_{(t)}^2 + \langle z_{t-1} - \hat{z}_{t-1}, x_t - \tilde{x}_t\rangle + (t - \hat{t})\left(\phi(x_t) - \phi(\tilde{x}_t)\right) \\
&\quad + (r_{0:t-1}(x_t) - \hat{r}_{0:t-1}(x_t)) - (r_{0:t-1}(\tilde{x}_t) - \hat{r}_{0:t-1}(\tilde{x}_t)) \\
&= -\frac{1}{2}\|x_t - \tilde{x}_t\|_{(t)}^2 + \langle \sum_{s\in O_t} I_{t,s} g_s, x_t - \tilde{x}_t\rangle + (t - \hat{t})\left(\phi(x_t) - \phi(\tilde{x}_t)\right) + \Delta_t \\
&\leq \frac{1}{2}\left\|\sum_{s\in O_t} I_{t,s} g_s\right\|_{(t,*)}^2 + (t - \hat{t})\left(\phi(x_t) - \phi(\tilde{x}_t)\right) + \Delta_t\,, \qquad (5.19)
\end{aligned}$$

where in the last step we have used the Fenchel-Young inequality, completing the proof. □

We can now prove the theorem.

Proof of Theorem 5.5. For $t = 1, 2, \ldots, T$, recall that the imaginary iterate $\tilde{x}_t$ is defined by Eq. (5.9)

$$\tilde{x}_t = \operatorname*{arg\,min}_{x\in\mathcal{X}} \langle g_{1:t-1}, x\rangle + (t+1)\phi(x) + r_{0:t-1}(x)\,,$$

and note that in addition to the difference between $r_{0:t-1}$ and $\hat{r}_{0:t-1}$, the actual iterate x_t and the imaginary iterate $\tilde{x}_t$ have a difference of $\nu_t\phi(x)$ in their regularization.

Starting from the regret decomposition, and using the linear regret of the imaginary iterate $\tilde{x}_t$, as well as the fact that r_t are non-negative by Assumption 5.3, we have

$$\begin{aligned}
R_T^{(f+\phi)}(x^*) &\leq \sum_{t=1}^{T}\langle g_t, \tilde{x}_t - x^*\rangle + \tilde{\epsilon}_{1:T} + \delta_{1:T} - B_{1:T} + \sum_{t=1}^{T}\left(\phi(\tilde{x}_t) - \phi(x^*)\right) \\
&\leq r_{0:T}(x^*) - \sum_{t=0}^{T} r_t(\tilde{x}_{t+1}) + \sum_{t=1}^{T}\frac{1}{2}\|g_t\|_{(t,*)}^2 + \tilde{\epsilon}_{1:T} + \delta_{1:T} - B_{1:T} \\
&\leq r_{0:T}(x^*) + \sum_{t=1}^{T}\frac{1}{2}\|g_t\|_{(t,*)}^2 + \tilde{\epsilon}_{1:T} + \delta_{1:T} - B_{1:T}\,. \qquad (5.20)
\end{aligned}$$

In the above, the first inequality follows by Lemma 5.1. The second inequality follows by bounding the linear regret $\sum_{t=1}^{T}\langle g_t, \tilde{x}_t - x^*\rangle$ using Theorem 5.3, and the third by dropping the non-negative terms $r_t(\tilde{x}_{t+1})$.

Next, we bound the penalty terms $\tilde{\epsilon}_{1:T}$. For each $t = 1, 2, \ldots, T$, using the fact that $\nu_t > 0$ by Assumption 5.3 and $\nu_t \in \sigma(\hat{\mathcal{H}}_t)$ by definition, we have

$$\begin{aligned}
\mathbb{E}\{\tilde{\epsilon}_t + \phi(\tilde{x}_t) - \phi(x_t)\} &\le \mathbb{E}\bigg\{\frac{p_*\nu_t}{2}\|g_t\|^2_{(t,*)} + \frac{1}{2\nu_t}\|(x_t - \tilde{x}_t)\|^2_{(t)}\bigg\} \\
&\le \mathbb{E}\bigg\{\frac{p_*\nu_t}{2}\|g_t\|^2_{(t,*)}\bigg\} \\
&\quad + \mathbb{E}\left\{\frac{1}{2\nu_t}\left(\bigg\|\sum_{s\in O_t} I_{t,s}g_s\bigg\|^2_{(t,*)} + 2(\nu_t\phi(\tilde{x}_t) - \nu_t\phi(x_t) + \Delta_t)\right)\right\} \\
&\le \mathbb{E}\bigg\{\frac{p_*\nu_t}{2}\|g_t\|^2_{(t,*)} + \sum_{s\in O_t}\frac{\tau_t}{2\nu_t}\|I_{t,s}g_s\|^2_{(t,*)} + \frac{\Delta_t}{\nu_t} + \phi(\tilde{x}_t) - \phi(x_t)\bigg\} \\
&\le \mathbb{E}\bigg\{\frac{p_*\nu_t}{2}\|g_t\|^2_{(t,*)} + \sum_{s\in O_t}\frac{\tau_t}{2\nu_t}\|g_s\|^2_{(t,*)} + \frac{\Delta_t}{\nu_t} + \phi(\tilde{x}_t) - \phi(x_t)\bigg\} \\
&\le \mathbb{E}\bigg\{\frac{p_*\nu_t}{2}\|g_t\|^2_{(t,*)} + \sum_{s\in O_t}\frac{\tau_t}{2\nu_t}\|g_s\|^2_{(s,*)} + \frac{\Delta_t}{\nu_t} + \phi(\tilde{x}_t) - \phi(x_t)\bigg\}.
\end{aligned} \tag{5.21}$$

The first inequality above uses Lemma 5.3 with $c_t = p_*\nu_t$ (using the assumption of $\|\cdot\|_{(t)}$ being a weighted q-norm with $q = 1$ or $q = 2$), the second follows by Lemma 5.4, the third uses the convexity of the norms $\|\cdot\|^2_{(t,*)}$ and Jensen's inequality, the forth follows because $I_{t,s}$ is a $\{0,1\}$-valued diagonal matrix and $\|\cdot\|_{(t)}$ is a weighted q-norm, and hence $\|I_{t,s}g_s\|_{(t,*)} \le \|g_s\|_{(t,*)}$, and the last line follows because $s \in O_t$ implies $s \le t$ by construction, and for $s \le t$, the dual norms satisfy $\|\cdot\|_{(t,*)} \le \|\cdot\|_{(s,*)}$ by Assumption 5.3. Summing the second term on the r.h.s. of (5.21), for $t = 1, 2, \ldots, T$, we get

$$\sum_{t=1}^{T}\sum_{s\in O_t}\frac{\tau_t}{2\nu_t}\|g_s\|^2_{(s,*)} = \sum_{s=1}^{T}\left(\sum_{t:s\in O_t}\frac{\tau_t}{2\nu_t}\right)\|g_s\|^2_{(s,*)}\ , \tag{5.22}$$

Thus, summing (5.21) over t, combining with (5.22), and noting that the terms $\phi(x_t)-\phi(\tilde{x}_t)$ cancel from the sides of the asyncrony penalty bounds (5.21), we get

$$\mathbb{E}\Big\{R_T^{(f+\phi)}(x^*)\Big\} \le \mathbb{E}\left\{r_{0:T}(x^*) + \sum_{t=1}^{T}\frac{1 + p_*\nu_t + \sum_{s:t\in O_s}\frac{\tau_s}{\nu_s}}{2}\|g_t\|^2_{(t,*)} + \frac{\Delta_t}{\nu_t} + \delta_{1:T}\right\}.$$

Finally, noting that $x_t \in \sigma(\mathcal{H}_t)$ by definition, by Assumption 5.4 it follows that $\mathbb{E}\{\delta_t|\mathcal{H}_t\} = 0$ in the stochastic setting. This completes the proof. □

5.E Extra details for the analysis of serial Ada-FTRL

A typical proxy for bounding the regret of serial optimization algorithms is *linearizing* the loss, and studying the linearized regret [16], [38], [97]. In particular, we define the linearized forward regret

$$R_T^+(x^*) = \sum_{t=1}^{T} \langle g_t, x_{t+1} - x^* \rangle ,$$

and carry out the analysis in two steps: a decomposition of the regret in terms of the forward regret R_T^+, followed by a bound on R_T^+. To that end, we need the following assumption.

Definition 5.2 (Admissible regularizers.). *A sequence of regularizer functions $(r_t)_{t=0}^T$ is "admissible" for* Ada-FTRL *if and only if all r_t are defined on a common convex domain $S \subset \mathbb{R}^d$, the intersection $\mathcal{X} \cap S$ is non-empty, and there exists a sequence of norms $\big(\|\cdot\|_{(t)}\big)_{t=1}^T$ such that for all $t = 1, 2, \ldots, T$, the cumulative regularizer $t\phi + r_{0:t-1} : S \to \mathbb{R}$ is lower-semi-continuous and* 1*-strongly-convex w.r.t. $\|\cdot\|_{(t)}$.*

As shown by Lemma 5.5, admissible regularizers guarantee that the Ada-FTRL updates (5.6) are well-defined, that is, there exists some $x_{t+1} \in \mathcal{X}$ that satisfies (5.6), and the associated optimal value is finite.

Lemma 5.5 (Well-posed Ada-FTRL). *For all $t = 0, 1, \ldots, T$, the argmin sets that define x_{t+1} in the* Ada-FTRL *updates* (5.6) *are non-empty, and their optimal values are finite.*

Proof. Fix $t \in [T]$, and consider the extended-value function $h_t = \langle z_{t-1}, \cdot \rangle + r_{0:t-1} + \mathcal{I}_{x \in S \cap \mathcal{X}}$, which is proper, l.s.c. and convex by construction. In addition, since $r_{0:t-1}$ is strongly-convex over S, then h_t is l.s.c. and 1-strong-convex on $\mathbb{R}^d$. The result then follows by Proposition 17.26 of [6], noting that x_t will be the corresponding minimizer by definition. □

Then, we have the following bound on the regret of Ada-FTRL.

Theorem 5.6 (Forward regret of Ada-FTRL, [47].). *For any $x^* \in \mathcal{X}$ and for any sequence of linear losses $\langle g_t, \cdot \rangle, t = 1, 2, \ldots, T$, and using any sequence of admissible regularizers $r_0, r_1, \ldots, r_T$, the forward regret of* Ada-FTRL *satisfies*

$$R_T^+(x^*) \leq r_{0:T}(x^*) - \sum_{t=0}^{T} r_t(x_{t+1}) + \sum_{t=1}^{T} (\phi(x^*) - \phi(x_t)) - \sum_{t=1}^{T} \mathcal{B}_{r_{0:t-1}}(x_{t+1}, x_t) . \quad (5.23)$$

Theorem 5.3 follows as a direct consequence of the above theorem by using the strong convexity of $r_{0:t-1}$ and the Fenchel-Young inequality; see [47] for further details.

Conclusion

We studied three problems in fast cross-validation, distributed optimization, and parallel asynchronous optimization with online algorithms.

First, we studied fast cross-validation of online algorithms. We provided a novel algorithm, TreeCV, that organized the computation in a tree-structure to avoid wasteful re-training in k-fold CV.

Second, we studied online algorithms for adaptive distributed optimization. We provided a generic meta-algorithm, Solid, to extend standard online optimization algorithms, and their analysis, to the distributed setting, and used it to provide a delay-adaptive step-size tuning scheme for AdaGrad algorithms.

Third, we provided a refined, modular analysis of online optimization algorithms, derived new optimistic MD and adaptive composite-objective scale-free FTRL algorithms with variational bounds, and extended generic online adaptive FTRL algorithms to perturbed-feedback and asynchronous composite-objective optimization settings with arbitrary constraint sets. Based on this extension, we presented and analyzed AsynCADA, a new asynchronous, generically-constrained composite optimization algorithm, and Hedge-Hog, an asynchronous variant of the Hedge / Exponentiated-Gradient algorithm. For both algorithms, we demonstrated linear speed-ups under a range of problem settings, expanding the range of problem settings and algorithmic techniques adoptable to asynchronous optimization.

An immediate direction for future research is applying the analysis framework presented in Chapters 4 and 5 to saddle-point formulations of reinforcement-learning algorithms. In that case, too, a straightforward regret-decomposition relates the dual gap of the problem to the online regret [74], enabling us to extend the serial, parallel and distributed algorithms discussed here to reinforcement learning.

References

[1] A. Agarwal and J. Duchi, “Distributed delayed stochastic optimization,” in *Advances in Neural Information Processing Systems 24 (NIPS)*, J. Shawe-Taylor *et al.*, Eds., 2011, pp. 873–881.

[2] A. Agarwal *et al.*, “Information-theoretic lower bounds on the oracle complexity of convex optimization,” in *Advances in Neural Information Processing Systems*, 2009, pp. 1–9.

[3] Z. Allen-Zhu, “Natasha 2: Faster non-convex optimization than sgd,” in *Advances in Neural Information Processing Systems 31*, 2018, pp. 2675–2686.

[4] Z. Allen-Zhu and Y. Yuan, “Improved SVRG for Non-Strongly-Convex or Sum-of-Non-Convex Objectives,” in *Proceedings of the 33rd International Conference on Machine Learning*, ser. ICML ’16, 2016.

[5] S. An *et al.*, “Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression,” *Pattern Recognition*, Part Special Issue on Visual Information Processing, vol. 40, no. 8, pp. 2154–2162, Aug. 2007.

[6] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science & Business Media, 2011.

[7] A. Beck and M. Teboulle, “Mirror descent and nonlinear projected subgradient methods for convex optimization,” *Operations Research Letters*, vol. 31, no. 3, pp. 167–175, 2003.

[8] D. P. Bertsekas and S. E. Shreve, *Stochastic optimal control: The discrete time case*. Academic Press New York, 1978, vol. 23.

[9] S. Bhojanapalli *et al.*, “Global optimality of local search for low rank matrix recovery,” in *Advances in Neural Information Processing Systems 29*, 2016, pp. 3873–3881.

[10] L. Cannelli *et al.*, “Asynchronous parallel algorithms for nonconvex optimization,” *arXiv preprint arXiv:1607.04818*, 2016.

[11] ——, “Asynchronous parallel algorithms for nonconvex big-data optimization. part i: Model and convergence,” *arXiv preprint arXiv:1607.04818*, 2017.

[12] ——, “Asynchronous parallel algorithms for nonconvex big-data optimization. part ii: Complexity and numerical results,” *arXiv preprint arXiv:1701.04900*, 2017.

[13] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," *Advances in neural information processing systems*, pp. 409–415, 2001.

[14] G. Cawley, "Leave-One-Out Cross-Validation Based Model Selection Criteria for Weighted LS-SVMs," in *International Joint Conference on Neural Networks, 2006. IJCNN '06*, 2006, pp. 1661–1668.

[15] N. Cesa-Bianchi *et al.*, "On the generalization ability of on-line learning algorithms," *Information Theory, IEEE Transactions on*, vol. 50, no. 9, pp. 2050–2057, 2004.

[16] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. New York, NY, USA: Cambridge University Press, 2006.

[17] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 27:1–27:27, 3 2011, Datasets available at http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/.

[18] C.-K. Chiang *et al.*, "Online optimization with gradual variations," in *Conference on Learning Theory*, 2012.

[19] A. Christmann and A. v. Messem, "Bouligand derivatives and robustness of support vector machines for regression," *The Journal of Machine Learning Research*, vol. 9, pp. 915–936, 2008.

[20] K. L. Clarkson *et al.*, "Sublinear optimization for machine learning," *Journal of the ACM*, vol. 59, no. 5, 23:1–23:49, Nov. 2012.

[21] D. Davis *et al.*, "The sound of apalm clapping: Faster nonsmooth nonconvex optimization with stochastic asynchronous palm," in *Advances in Neural Information Processing Systems*, 2016, pp. 226–234.

[22] M. Debruyne *et al.*, "Model selection in kernel based regression using the influence function," *Journal of Machine Learning Research*, vol. 9, no. 10, 2008.

[23] A. Defazio *et al.*, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in neural information processing systems*, 2014, pp. 1646–1654.

[24] O. Dekel *et al.*, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 165–202, 2012.

[25] J. C. Duchi *et al.*, "Composite objective mirror descent," in *Conference on Learning Theory*, 2010, pp. 14–26.

[26] J. Duchi *et al.*, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, Jul. 2011.

[27] J. Duchi *et al.*, "Estimation, optimization, and parallelism when data is sparse," in *Advances in Neural Information Processing Systems*, 2013, pp. 2832–2840.

[28] F. Facchinei *et al.*, "Parallel selective algorithms for nonconvex big data optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 7, pp. 1874–1889, 2015.

[29] O. Fercoq and P. Richtárik, "Optimization in high dimensions via accelerated, parallel, and proximal coordinate descent," *SIAM Review*, vol. 58, no. 4, pp. 739–771, 2016.

[30] J. Friedman *et al.*, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[31] R. Ge *et al.*, "Matrix completion has no spurious local minimum," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 2973–2981.

[32] A. Girard, "A fast 'Monte-Carlo cross-validation' procedure for large least squares problems with noisy data," en, *Numerische Mathematik*, vol. 56, no. 1, pp. 1–23, Jan. 1989.

[33] G. H. Golub and U. von Matt, "Generalized Cross-Validation for Large-Scale Problems," *Journal of Computational and Graphical Statistics*, vol. 6, no. 1, pp. 1–34, Mar. 1997.

[34] G. H. Golub *et al.*, "Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter," *Technometrics*, vol. 21, no. 2, pp. 215–223, May 1979.

[35] I. Goodfellow *et al.*, *Deep learning*. MIT Press, 2016.

[36] N. Görnitz *et al.*, "Toward supervised anomaly detection," *Journal of Artificial Intelligence Research*, vol. 46, no. 1, pp. 235–262, 2013.

[37] M. Hardt *et al.*, "Gradient descent learns linear dynamical systems," *arXiv preprint arXiv:1609.05191*, 2016.

[38] E. Hazan, "Introduction to online convex optimization," *Foundations and Trends in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[39] E. Hazan and S. Kale, "Beyond the regret minimization barrier: Optimal algorithms for stochastic strongly-convex optimization," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2489–2512, 2014.

[40] E. Hazan *et al.*, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2, pp. 169–192, 2007.

[41] E. Hazan *et al.*, "Beyond convexity: Stochastic quasi-convex optimization," in *Advances in Neural Information Processing Systems 28 (NIPS 28)*, 2015, pp. 1594–1602.

[42] M. Izbicki, "Algebraic classifiers: A generic approach to fast cross- validation, online training, and parallel training," in *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, May 2013, pp. 648–656.

[43] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*, 2013, pp. 315–323.

[44] P. Joulani *et al.*, "Online learning under delayed feedback," in *Proceedings of The 30th International Conference on Machine Learning (ICML)*, (extended version at arXiv:1306.0686), 2013.

[45] ——, “Fast cross-validation for incremental learning,” in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*, 2015.

[46] ——, “Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms,” in *Proceedings of the 30th Conference on Artificial Intelligence (AAAI-16)*, 2016.

[47] ——, “A modular analysis of adaptive (non-) convex optimization: Optimism, composite objectives, and variational bounds,” in *Proceedings of Machine Learning Research (Algorithmic Learning Theory 2017)*, 2017, pp. 681–720.

[48] A. Juditsky *et al.*, “Solving variational inequalities with stochastic mirror-prox algorithm,” *Stochastic Systems*, vol. 1, no. 1, pp. 17–58, 2011.

[49] S. M. Kakade and A. Tewari, “On the generalization ability of online strongly convex programming algorithms,” in *Advances in Neural Information Processing Systems*, 2009, pp. 801–808.

[50] P. Kamalaruban, “Improved optimistic mirror descent for sparsity and curvature,” *arXiv preprint arXiv:1609.02383*, 2016.

[51] H. Karimi *et al.*, “Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition,” in *Proceedings of ECML/PKDD*, 2016, pp. 795–811.

[52] J. Kivinen and M. K. Warmuth, “Exponentiated gradient versus gradient descent for linear predictors,” *Information and Computation*, vol. 132, no. 1, pp. 1–63, 1997.

[53] K. C. Kiwiel, “Proximal minimization methods with generalized bregman functions,” *SIAM Journal on Control and Optimization*, vol. 35, no. 4, pp. 1142–1168, 1997.

[54] B. Kulis and P. L. Bartlett, “Implicit online learning,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 575–582.

[55] G. Lan, “An optimal method for stochastic composite optimization,” *Mathematical Programming*, vol. 133, no. 1, pp. 365–397, 2012.

[56] R. Leblond *et al.*, “ASAGA: Asynchronous Parallel SAGA,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, A. Singh and J. Zhu, Eds., ser. Proceedings of Machine Learning Research, vol. 54, Fort Lauderdale, FL, USA: PMLR, Apr. 2017, pp. 46–54. [Online]. Available: `http://proceedings.mlr.press/v54/leblond17a.html`.

[57] R. Leblond *et al.*, “Improved asynchronous parallel optimization analysis for stochastic incremental methods,” *arXiv preprint arXiv:1801.03749*, 2018.

[58] J. C. H. Lee and P. Valiant, “Optimizing star-convex functions,” in *IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 2016, pp. 603–614.

[59] K. Y. Levy, “Online to offline conversions and adaptive minibatch sizes,” *arXiv preprint arXiv:1705.10499*, 2017.

[60] M. Lichman, *UCI machine learning repository*, 2013.

[61] J. Liu and S. J. Wright, "Asynchronous stochastic coordinate descent: Parallelism and convergence properties," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 351–376, 2015.

[62] J. Liu *et al.*, "An asynchronous parallel stochastic coordinate descent algorithm," *arXiv preprint arXiv:1311.1873*, 2013.

[63] Y. Liu *et al.*, "Efficient Approximation of Cross-Validation for Kernel Methods using Bouligand Influence Function," in *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, ser. JMLR W&CP, vol. 32, 2014, pp. 324–332.

[64] S. Łojasiewicz, "A topological property of real analytic subsets," *Coll. du CNRS, Les équations aux dérivées partielles*, vol. 117, pp. 87–89, 1963.

[65] H. Mania *et al.*, "Perturbed Iterate Analysis for Asynchronous Stochastic Optimization," *ArXiv e-prints*, Jul. 2015. arXiv: `1507.06970 [stat.ML]`.

[66] B. McMahan and M. Streeter, "Delay-tolerant algorithms for asynchronous distributed online learning," in *Advances in Neural Information Processing Systems*, 2014, pp. 2915–2923.

[67] H. B. McMahan, "A survey of algorithms and analysis for adaptive online learning," *arXiv preprint arXiv:1403.3465*, 2014.

[68] H. B. McMahan, "Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, 2011, pp. 525–533.

[69] ——, "A survey of algorithms and analysis for adaptive online learning," *Journal of Machine Learning Research*, vol. 18, no. 90, pp. 1–50, 2017.

[70] H. B. McMahan and M. Streeter, "Adaptive bound optimization for online convex optimization," in *Proceedings of the 23rd Conference on Learning Theory*, 2010.

[71] C. J. Mesterharm, "Improving on-line learning," PhD thesis, Department of Computer Science, Rutgers University, New Brunswick, NJ, 2007.

[72] M. Mohri and S. Yang, "Accelerating online convex optimization via adaptive prediction," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016, pp. 848–856.

[73] M. D. Mullin and R. Sukthankar, "Complete Cross-Validation for Nearest Neighbor Classifiers.," in *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, 2000, pp. 639–646.

[74] A. Nemirovski *et al.*, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[75] A. Nemirovsky and D. Yudin, *Problem complexity and method efficiency in optimization.* Chichester, New York: Wiley, 1983.

[76] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[77] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical programming*, vol. 120, no. 1, pp. 221–259, 2009.

[78] ——, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013, vol. 87.

[79] Y. Nesterov and B. T. Polyak, "Cubic regularization of newton method and its global performance," *Mathematical Programming*, vol. 108, no. 1, pp. 177–205, 2006.

[80] L. M. Nguyen *et al.*, "Sgd and hogwild! convergence without the bounded gradients assumption," *arXiv preprint arXiv:1802.03801*, 2018.

[81] N. Nguyen *et al.*, "Efficient generalized cross-validation with applications to parametric image restoration and resolution enhancement," *IEEE Transactions on Image Processing*, vol. 10, no. 9, pp. 1299–1308, Sep. 2001.

[82] F. Orabona *et al.*, "A generalized online mirror descent with applications to classification and regression," English, *Machine Learning*, vol. 99, no. 3, pp. 411–435, 2015.

[83] T. Pahikkala *et al.*, "Fast n-fold cross-validation for regularized least-squares," in *In: Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI)*, 2006.

[84] X. Pan *et al.*, "Cyclades: Conflict-free asynchronous machine learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2568–2576.

[85] F. Pedregosa *et al.*, "Breaking the nonsmooth barrier: A scalable parallel method for composite optimization," in *Advances in Neural Information Processing Systems*, 2017, pp. 55–64.

[86] Z. Peng *et al.*, "Arock: An algorithmic framework for asynchronous parallel coordinate updates," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, A2851–A2879, 2016.

[87] B. T. Polyak, "Gradient methods for minimizing functionals," *Zh. Vychisl. Mat. Mat. Fiz.*, vol. 3, pp. 643–653, 1963.

[88] A. Rakhlin and K. Sridharan, "Online learning with predictable sequences.," in *Conference on Learning Theory*, 2013, pp. 993–1019.

[89] A. Rakhlin *et al.*, "Making gradient descent optimal for strongly convex stochastic optimization," in *Proceedings of the 29th International Coference on International Conference on Machine Learning (ICML'12)*, Edinburgh, Scotland, 2012, pp. 1571–1578.

[90] S. Rakhlin and K. Sridharan, "Optimization, learning, and games with predictable sequences," in *Advances in Neural Information Processing Systems*, 2013, pp. 3066–3074.

[91] M. Razaviyayn *et al.*, "Parallel successive convex approximation for nonsmooth nonconvex optimization," in *Advances in Neural Information Processing Systems*, 2014, pp. 1440–1448.

[92] B. Recht *et al.*, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor *et al.*, Eds., Curran Associates, Inc., 2011, pp. 693–701.

[93] A. Saha *et al.*, "The interplay between stability and regret in online learning," *arXiv preprint arXiv:1211.6158*, 2012.

[94] G. Scutari and Y. Sun, "Parallel and distributed successive convex approximation methods for big-data optimization," in *Multi-agent Optimization*, Springer, 2018, pp. 141–308.

[95] G. Scutari *et al.*, "Parallel and distributed methods for constrained nonconvex optimization-part ii: Applications in communications and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1945–1960, 2016.

[96] G. Scutari *et al.*, "Parallel and distributed methods for constrained nonconvex optimization—part i: Theory," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1929–1944, 2016.

[97] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.

[98] S. Shalev-Shwartz and S. M. Kakade, "Mind the duality gap: Logarithmic regret algorithms for online optimization," in *Advances in Neural Information Processing Systems*, 2009, pp. 1457–1464.

[99] S. Shalev-Shwartz *et al.*, "Pegasos : Primal estimated sub-gradient solver for svm," English, *Mathematical Programming*, vol. 127, no. 1, pp. 3–30, 2011.

[100] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes," in *Proceedings of the 30th International Coference on International Conference on Machine Learning (ICML'13)*, ser. JMLR Workshop and Conference Proceedings, vol. 28, 2013, pp. 71–79.

[101] S. Sra *et al.*, "Adadelay: Delay adaptive distributed stochastic optimization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016, pp. 957–965.

[102] T. Sun *et al.*, "Asynchronous coordinate descent under more realistic assumptions," in *Advances in Neural Information Processing Systems*, 2017, pp. 6182–6190.

[103] M. Telgarsky and S. Dasgupta, "Agglomerative bregman clustering," in *Proceedings of the 29th International Conference on Machine Learning - ICML*, 2012.

[104] G. Wahba, *Spline models for observational data.* SIAM, 1990, vol. 59.

[105] J. Wang and J. D. Abernethy, "Acceleration through optimistic no-regret dynamics," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 3824–3834.

[106] Y.-X. Wang *et al.*, "Parallel and distributed block-coordinate frank-wolfe algorithms," in *International Conference on Machine Learning*, 2016, pp. 1548–1557.

[107] M. K. Warmuth and A. Jagota, "Continuous and discrete-time nonlinear gradient descent: Relative loss bounds and convergence," in *Fifth International Symposium on Artificial Intelligence and Mathematics*, Florida, Jan. 1998.

[108] M. J. Weinberger and E. Ordentlich, "On delayed prediction of individual sequences," *IEEE Transactions on Information Theory*, vol. 48, no. 7, pp. 1959–1976, Sep. 2002.

[109] L. Xiao, "Dual averaging method for regularized stochastic learning and online optimization," in *Advances in Neural Information Processing Systems*, 2009, pp. 2116–2124.

[110] Z. Zhou *et al.*, "Stochastic mirror descent in variationally coherent optimization problems," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 7040–7049.

[111] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 928–936.