Università degli Studi di Milano

Doctoral Thesis

Adaptive and Implicit Online Learning

Author:
Nicolò Campolongo

Supervisor:
Nicolò Cesa-Bianchi

PhD Coordinator:
Paolo Boldi

PhD Program in Computer Science
(XXXIII cycle)

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

Academic Year 2019-2020

UNIVERSITÀ DEGLI STUDI DI MILANO

Abstract

Faculty Name
Dipartimento di Informatica

Doctor of Philosophy

Adaptive and Implicit Online Learning

by Nicolò Campolongo

This thesis is dedicated to the study of online learning algorithms. In particular, after reviewing fundamental concepts in the theory of convex optimization, we provide a refined analysis of Implicit updates in the framework of Online Mirror Descent. We design a new adaptive algorithm based on it and carefully study its regret bound in the static case, linking it to the variability of the sequence of loss functions. Furthermore, we extend its application to the dynamic setting, studying its dynamic regret. In particular, we show that it achieves the optimal dynamic regret bound, when the quantities of interest are observable or known beforehand. On the other hand, in order to have a fully adaptive algorithm we show how to combine a recently proposed algorithm in the literature with a simple greedy strategy. Finally, we focus on the well known problem of learning with experts advice. We review existing algorithm and describe an existing open problem. We provide some recent results and partial progress on how this problem could be addressed.

Contents

Chapter 1

Introduction

```
Online Convex Optimization (OCO)

Input A convex set V
For t = 1, 2, ..., T:
  Predict a model x_t ∈ V
  Receive a convex loss function ℓ_t : V → ℝ
  Suffer loss ℓ_t(x_t)
```

The *Online Convex Optimization* (OCO) paradigm is a powerful tool to model common real world scenarios where data comes in a streaming fashion (for example, in the case of time series). In the last two decades there has been a tremendous amount of progress in this field (see, e.g., Shalev-Shwartz [2012]; Hazan [2016]; Orabona [2019], for an introduction). Indeed, in the era of *Big Data* online algorithms proved to be a computationally efficient way to handle large datasets in the practical scenarios that the big companies face on a daily basis [McMahan et al., 2013]. Moreover, the ideas employed in the OCO setting led to advances in seemingly unrelated areas of machine learning and computer science. For example, using algorithms from OCO Christiano et al. [2011] reported the first advance in almost 40 years in the Approximate Maximum Flow problem on graphs; other applications involve for example graph-sparsification [Allen-Zhu et al., 2015], semidefinite programs [Arora and Kale, 2007], computational complexity [Barak et al., 2009], robust statistics [Hopkins et al., 2020] and convex geometry [Mirrokni et al., 2017].

In the OCO setting, a learning agent faces the environment in a game played sequentially. The protocol is the following: given a time horizon T, in every round $t = 1, \dots, T$ the agent chooses a model $\boldsymbol{x}_t$ from a convex set V. Then, a convex loss function ℓ_t is revealed by the environment and the agent pays a loss $\ell_t(\boldsymbol{x}_t)$.

As usually done in the literature, we do not make assumptions on the environment, but allow it to be adversarial. There are different models of adversary which can be taken into consideration. However, in this work we will only consider oblivious adversaries, meaning that the sequence of loss functions is decided prior to the game starting and is not influenced by the decision maker's choices. The agent's goal is to minimize her regret against any fixed competitor, i.e., the cumulative sum of her losses compared to the losses of an agent which always commits to the same choice $\boldsymbol{u}$. Formally, the regret against any $\boldsymbol{u} \in V$ is defined as

$$R_T(\boldsymbol{u}) \triangleq \sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} \ell_t(\boldsymbol{u}) \ . \tag{1.1}$$

Note that by competing against any $\boldsymbol{u} \in V$, we automatically compete against the model $\boldsymbol{u}$ which minimizes $\sum_{t=1}^{T} \ell_t(\boldsymbol{u})$. When designing algorithms in the OCO setting, the main goal is to provide an upper bound to the regret. A reasonable goal is to achieve a vanishing regret when the time horizon T goes to infinity, i.e.,

$$\lim_{T \to \infty} \frac{R_T}{T} \to 0 \ .$$

An algorithm satisfying this property is dubbed as *no-regret*. While an asymptotic analysis might be useful, a more ambitious goal is to study this setting assuming a given finite time horizon T. This might seem more complicated at first, but the advances in the last two decades made the finite time analysis the standard de facto in this field. The first question that comes to mind is then: what is achievable in this setting? It can be proven that no matter what the learning strategy does, there is an instance in OCO such that the worst case regret is lower bounded by a quantity which is $\Omega(\sqrt{T})$ (see, e.g., [Orabona, 2019, Theorem 5.1]). Given this result, the main goal is therefore to develop algorithms whose regret is upper bounded by $\mathcal{O}(\sqrt{T})$. Depending on the domain V, there are currently many algorithms achieving this goal, whose analysis is clear and well understood. On the other hand, in recent years the trend has shifted towards developing *data-adaptive* algorithms which can take advantage of well behaved sequences of loss functions, and achieve better regret bounds. At the same time, these algorithms should be able to retain the worst case regret bound if the environment is truly adversarial. For example, rather than focusing on $\mathcal{O}(\sqrt{T})$ bounds, one can try to derive a bound in terms of the cumulative loss of the minimizer $\boldsymbol{u}$, which could potentially be zero. In this thesis, we follow this line of work and provide some new data-adaptive algorithms, which improve existing regret bounds when the sequence of loss functions does not vary much over time.

Next, we provide a brief description of the building blocks of this thesis.

1.1 Outline

This manuscript is organized in 5 chapters, beyond this first introductory chapter. Each chapter explores a different but related topic.

Chapter 2. In this chapter, we provide some mathematical concepts necessary for the understanding of the rest of the material in this thesis. In particular, these include several tools from the theory of convex optimization, which are now standard in the online setting.

Chapter 3. Much of the progress in the OCO field is driven by the strictly related model of Online Linear Optimization (OLO): exploiting the assumption that the loss functions are convex, we can linearize them using a first-order approximation through its (sub)gradient and subsequently minimize the linearized regret. We try to formalize this, without diving into technicalities (we refer to Section 2.2 for a broader treatment of convex functions): assuming the convexity of the loss functions, we have that $\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u}) \leq \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u} \rangle$, where $\boldsymbol{g}_t$ is a subgradient of ℓ_t in $\boldsymbol{x}_t$. This observation immediately gives an upper bound on the regret defined in Equation (1.1)

$$R_T(\boldsymbol{u}) \leq \sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u} \rangle \ ,$$

and providing an upper bound to it might be easier compared to directly analyze Equation (1.1).

In Chapter 3 we review the main algorithms and regret bounds derived using linearized loss functions using subgradients. For example, the well-known Online Gradient Descent (OGD) [Zinkevich, 2003] to update its model simply uses the direction of the negative (sub)gradient of the loss function, multiplied by a given learning rate. Usually, a properly tuned learning rate gives a regret bound of $\mathcal{O}(\sqrt{T})$.

Chapter 4. Instead of updating our model using an approximation of the loss function through its subgradient, we can choose to use directly the loss function. In the online setting, this type of update is known as *Implicit*, and algorithms designed in this way are known to have practical advantages [Kulis and Bartlett, 2010]. Unfortunately, their theoretical understanding is still limited at this point. Chapter 4 will be devoted to the investigation of Implicit Updates, in order to explain why and when they can be preferred over linearized updates. This investigation will lead us to the design of a new adaptive algorithm. In addition to proving a regret bound for the proposed algorithm, we also conduct an extensive experimental analysis.

Chapter 5. Most strategies in OCO target the static regret as defined in Equation (1.1). However, in many scenarios the regret against the best fixed model $\boldsymbol{u}$ could be a poor benchmark. In particular, when the environment is dynamic, a better option might be to compete against a sequence of different models $(\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_T)$. In this case the notion of regret has to be modified and we get the so called *dynamic regret*:

$$R_T(\boldsymbol{u}_{1:T}) \triangleq \sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} \ell_t(\boldsymbol{u}_t) \ .$$

Clearly, this is a more difficult objective. However, if one makes reasonable assumptions about the sequence of comparators $(\boldsymbol{u}_1, \dots, \boldsymbol{u}_T)$ or the sequence of loss functions $\ell_1, \dots, \ell_T$ a regret bound of $\mathcal{O}(\sqrt{T})$ is still achievable. In Chapter 5 we show how our proposed algorithms using Implicit Updates adapt to dynamic environments. In particular, we provide improved regret bounds compared to the existing literature in terms of the temporal variability of the sequence of loss functions.

Chapter 6. In the online setting, special attention is given to the problem of *Learning with Expert Advice* [Cesa-Bianchi et al., 1997], which is defined as follows: a learning agent is given a set of d experts. In every round she has to pick one expert (or a convex combination of them) and follow her advice. At the end of each round, the loss of every expert is revealed and the agent pays the loss associated to the one she chose. The agent's goal is to minimize her against the best expert. In this case the well-known Exponential Weights Update algorithm is worst-case optimal, achieving a regret bound of $\mathcal{O}(\sqrt{T \ln d})$. However, this regret bound can be improved in two ways: first, one can replace the worst case term T with a data-adaptive quantity, such as the loss of the best expert; second, the $\ln d$ term in the the bound can be made smaller if there are multiple good experts. In an unresolved open problem, Freund [2016] asks whether there exists an algorithm which can combine these advantages and achieve an improved regret bound compared to $\mathcal{O}(\sqrt{T \ln d})$.

In Chapter 6 of this thesis, we will review the existing literature related to this problem, and further describe some techniques which could be used to address the problem.

Declaration of authorship. We point out that the material in Chapter 4 is based on Campolongo and Orabona [2020], while part of the material in Chapter 5 is in a work submitted and currently under review. Finally, the last section of Chapter 6 contains new preliminary results and is part of ongoing work. All remaining chapters are based on existing literature and references are always included.

Chapter 2

Mathematical Background

In this chapter we give some mathematical definitions that will be useful in the rest of this manuscript. We first list some concepts and notation used in this thesis, then briefly describe necessary notions from convex optimization.

2.1 Notation and Preliminaries

First of all, we will only work on the *Euclidean* space $\mathbb{R}^d$ or subsets of it. In particular, $\mathbb{R}_+ \triangleq \{x \in \mathbb{R} : x \geq 0\}$. Throughout the text we use the **bold** notation to denote vectors in $\mathbb{R}^d$. We use the symbol $[n]$ to denote the set of natural number from 1 to n, i.e., $[n] \triangleq \{1, \ldots, n\}$ for $n \in \mathbb{N}$.

We recall that a *Euclidean* space $\mathbb{E}$ is a finite-dimensional real vector space endowed with an *inner product* $\langle \cdot, \cdot \rangle$, where the latter is an assignment $\langle \cdot, \cdot \rangle : \mathbb{E} \times \mathbb{E} \to \mathbb{R}$ satisfying the following three properties for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{E}$ and scalars $a, b \in \mathbb{R}$:

- **Simmetry.** $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$.
- **Bilinearity.** $\langle a\boldsymbol{x} + b\boldsymbol{y}, \boldsymbol{z} \rangle = a\langle \boldsymbol{x}, \boldsymbol{z} \rangle + b\langle \boldsymbol{y}, \boldsymbol{z} \rangle$.
- **Positive definiteness.** $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$ with equality holding iff $\boldsymbol{x} = \boldsymbol{0}$.

Therefore, in the case of space d-dimensional column vectors in $\mathbb{R}^d$, we adopt the standard *dot-product*

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle \triangleq \sum_{i=1}^{d} x_i y_i \ .$$

We will denote the coordinate vectors of $\mathbb{R}^d$ by $\boldsymbol{e}_i$.

Next, we present a very useful and well-known inequality.

Theorem 2.1 (Cauchy-Schwartz inequality)**.** *For any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, the following holds*

$$|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 \ ,$$

where $\|\boldsymbol{z}\|_2 \triangleq \sqrt{\langle \boldsymbol{z}, \boldsymbol{z} \rangle}$.

Proof. First, assume that $\|\boldsymbol{x}\|_2 = \|\boldsymbol{y}\|_2 = 1$. Then, we have that

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle + \langle \boldsymbol{y}, \boldsymbol{y} \rangle - 2\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 2(1 - \langle \boldsymbol{x}, \boldsymbol{y} \rangle) \geq 0 \ ,$$

from which we get $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \leq 1$.

Next, consider any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. If one (or both) of the vectors are $\boldsymbol{0}$, then the inequality is trivially verified. On the other hand, if they are both non-zero we have that $\boldsymbol{x}/\|\boldsymbol{x}\|_2$ and $\boldsymbol{y}/\|\boldsymbol{y}\|_2$ are unit vectors. Therefore,

$$\langle \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}, \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_2} \rangle \leq 1 \ .$$

which implies $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \leq \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2$.

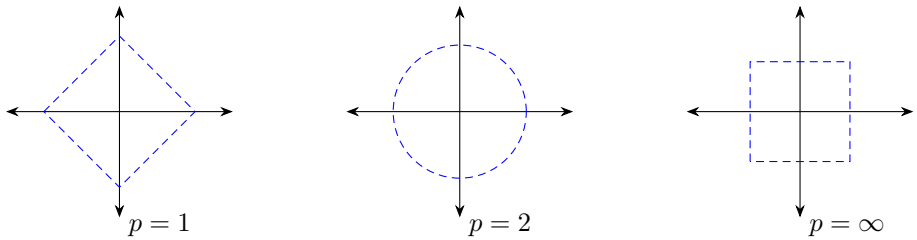


FIGURE 2.1: ℓ_p unit balls displayed according to different values of p.

Since the above holds for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, we can substitute $-\boldsymbol{y}$ to $\boldsymbol{y}$ and get the following

$$\langle \boldsymbol{x}, -\boldsymbol{y} \rangle \leq \|\boldsymbol{x}\|_2 \| - \boldsymbol{y}\|_2 = \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 \ ,$$
$$\langle \boldsymbol{x}, -\boldsymbol{y} \rangle = -\langle \boldsymbol{x}, \boldsymbol{y} \rangle \leq \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 \ ,$$

from which the stated result follows. □

Note that the definition of the function $\|\cdot\|_2$ reflects a notion of distance, which we generalize in Section 2.1.1.

Topological notions. Let $V \subset \mathbb{R}^d$. A point $\boldsymbol{x} \in V$ is an *interior point* if there exists an $\varepsilon > 0$ such that

$$B_\varepsilon(\boldsymbol{x}) = \{\boldsymbol{y} : \|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \varepsilon\} \subset V \ .$$

The *interior* of V is denoted by $\operatorname{int} V$ and defined as

$$\operatorname{int} V = \{\boldsymbol{x} \in V : \boldsymbol{x} \text{ is an interior point}\} \ .$$

The set V is *open* if $\operatorname{int} V = V$ and *closed* if its complement $V^c = \mathbb{R}^d \ V$ is open.

The *boundary* of V is denoted by ∂V and is the set of all points $\boldsymbol{x} \in \mathbb{R}^d$ such that for all $\varepsilon > 0$ the set $B_\varepsilon(\boldsymbol{x})$ contains points from V and V^c.

2.1.1 Norms

A norm on a vector space V is a function $\|\cdot\| : V \to \mathbb{R}$ for which the following three properties hold for all points $\boldsymbol{x}, \boldsymbol{y} \in V$ and scalars $a \in \mathbb{R}$:

- **Absolute homogeneity.** $\|a\boldsymbol{x}\| = |a| \cdot \|\boldsymbol{x}\|$.
- **Triangle inequality.** $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$.
- **Positivity.** $\|\boldsymbol{x}\| \geq 0$, with equality iff $\boldsymbol{x} = \boldsymbol{0}$.

For example, the $\|\cdot\|_2$ introduced in Theorem 2.1 is the well-known L_2 norm. We can generalize it as follows.

Definition 2.2 (p-norm)**.** *The p-norm of a vector $\boldsymbol{x} \in \mathbb{R}^d$ is defined as*

$$\|\boldsymbol{x}\|_p \triangleq \left(\sum_{i=1}^{d} |x_i|^p \right)^{1/p}$$

The most notable examples are the L_1, L_2 and L_∞ norms, which are displayed in Fig. 2.1.

We next define the important concept of dual norm.

Definition 2.3 (Dual Norm)**.** *The dual norm $\|\cdot\|_\star$ of a norm $\|\cdot\|$ is defined as*

$$\|\boldsymbol{\theta}\|_\star \triangleq \max_{\boldsymbol{x}:\, \|\boldsymbol{x}\| \leq 1} \langle \boldsymbol{\theta}, \boldsymbol{x} \rangle \ .$$

In words, the dual norm $\|\boldsymbol{\theta}\|_\star$ is the maximum value that the linear function $\boldsymbol{x} \to \langle \boldsymbol{\theta}, \boldsymbol{x} \rangle$ takes over the unit ball of the norm $\|\cdot\|$. For example, the dual norm of the L_2 norm is the L_2 norm itself. On the other hand, the dual norm of the L_1 norm is the L_∞ norm. In general, it can be proven (see Appendix A.4) that the dual norm of $\|\cdot\|_p$ is $\|\cdot\|_q$, where $\frac{1}{p} + \frac{1}{q} = 1$ and $p, q \geq 1$.

We note that in a sense any two norms on $\mathbb{R}^d$ are "equivalent". Indeed, for any two norms ϕ_1 and ϕ_2, there exists constant α, β such that

$$\alpha\phi_1(\boldsymbol{x}) \leq \phi_2(\boldsymbol{x}) \leq \beta\phi(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbb{R}^d \ .$$

For example, for any $\boldsymbol{x} \in \mathbb{R}^d$ we have

$$\|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1 \leq \sqrt{d}\|\boldsymbol{x}\|_2$$
$$\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \sqrt{d}\|\boldsymbol{x}\|_\infty \ .$$

2.1.2 Differentiability

Let $V \subseteq \mathbb{R}^d$. A function $f : V \to \mathbb{R}$ is *differentiable* in $\boldsymbol{x} \in V$ if there exists a vector, denoted by $\nabla f(\boldsymbol{x}) \in \mathbb{R}^d$ satisfying

$$\lim_{\boldsymbol{h} \to \boldsymbol{0}} \frac{f(\boldsymbol{x} + \boldsymbol{h}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{h} \rangle}{\|\boldsymbol{h}\|} = 0 \ .$$

The above can be reformulated with the following expression,

$$f(\boldsymbol{x} + \boldsymbol{h}) = f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{h} \rangle + o(\|\boldsymbol{h}\|) \ ,$$

where $o(\|\boldsymbol{h}\|)$ is a term satisfying $\lim_{\|\boldsymbol{h}\| \to 0} o(\|\boldsymbol{h}\|)/\|\boldsymbol{h}\| = 0$ and can be considered a *first-order* error since it decays to 0 faster than any linear function as $\boldsymbol{h}$ tends to zero. The vector $\nabla f(\boldsymbol{x})$ is called the *gradient* of f in $\boldsymbol{x}$. If the mapping $\boldsymbol{x} \to \nabla f(\boldsymbol{x})$ is well-defined and continuous on V, we say that f is C^1*-smooth.*

Definition 2.4 (Lipschitz function)**.** *Let $f : \mathbb{R}^d \to (-\infty, +\infty]$. Then, f is L-Lipschitz over a set V w.r.t. a norm $\|\cdot\|$ if $|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$, $\forall \boldsymbol{x}, \boldsymbol{y} \in V$.*

If the gradients satisfy a stronger Lipschitz property then we say f is β*-smooth.*

Definition 2.5 (β-smoothness)**.** *Let $f : V \to \mathbb{R}$ differentiable. Then f is β-smooth w.r.t. $\|\cdot\|$ if*

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq \beta\|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in V \ .$$

Also, using the notion of gradient we can derive a necessary condition for *local minimizers.* A point $\boldsymbol{x}$ is called a *local minimizer* of a function $f : \mathbb{R}^d \to \mathbb{R}$ if there exists a neighborhood Q_ε of $\boldsymbol{x}$ such that $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ for all $\boldsymbol{y} \in Q_\varepsilon$.

Definition 2.6 (Local minimizer)**.** *Let $V \subseteq \mathbb{R}^d$. A local minimizer of $f : V \to \mathbb{R}$ is a point $\boldsymbol{x}$ such that there exists $\varepsilon > 0$ with*

$$f(\boldsymbol{x}) \leq f(\boldsymbol{y}), \quad \forall \boldsymbol{y} \in V \text{ such that } \|\boldsymbol{y} - \boldsymbol{x}\|_2 \leq \varepsilon \ .$$

Theorem 2.7 ([Drusvyatskiy, 2020, Theorem 1.18])**.** *Let $V \subseteq \mathbb{R}^d$. Suppose that $\boldsymbol{x}$ is a local minimizer of a function $f : V \to \mathbb{R}$. If f is differentiable at $\boldsymbol{x}$, then equality $\nabla f(\boldsymbol{x}) = 0$ holds.*

Proof. Set $\boldsymbol{g} \triangleq -\nabla f(\boldsymbol{x})$. Then for all small $t > 0$, the definitions of differentiability and local minimizer imply that

$$0 \leq \frac{f(\boldsymbol{x} + t\boldsymbol{g}) - f(\boldsymbol{x})}{t} = -\|\nabla f(\boldsymbol{x})\|^2 + \frac{o(t)}{t} \ .$$

Letting t tend to zero yields $\nabla f(x) = 0$, as claimed. □

Finally, we introduce second order derivatives. A C^1-smooth function $f : V \to \mathbb{R}$ is twice differentiable at a point $\boldsymbol{x} \in V$ if the gradient map $\nabla f : V \to \mathbb{R}$ is differentiable at $\boldsymbol{x}$. In

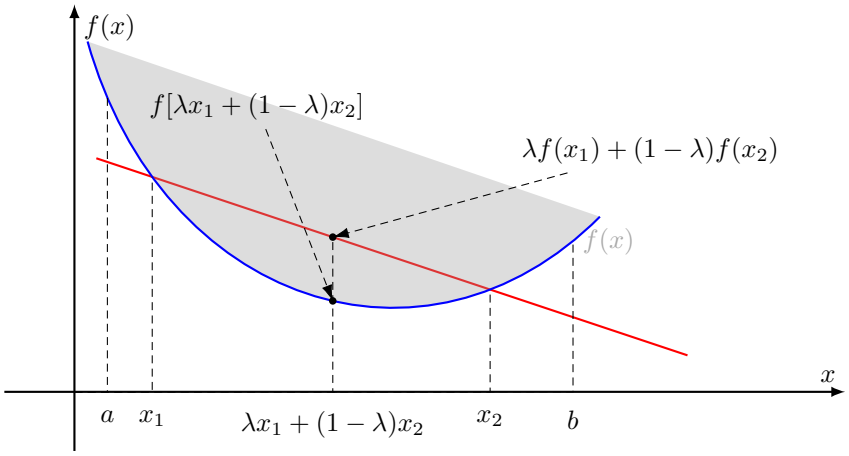


FIGURE 2.2: An illustration of convexity according to the definition in Eq. (2.1).

notation, $\nabla(\nabla f(\boldsymbol{x}))$ is denoted by $\nabla^2 f(\boldsymbol{x})$ and is called the *Hessian* of f in $\boldsymbol{x}$. In $\mathbb{R}^d$, we have that the Hessian is simply the matrix of second order partial derivatives, i.e.

$$\left[\nabla^2 f(\boldsymbol{x})\right]_{i,j} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j}, \quad \forall i,j \in [d] \ .$$

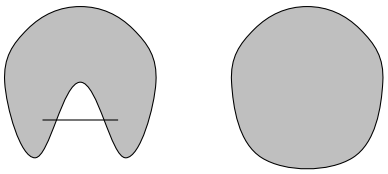
2.2 A short tour of convexity

In this section, we will provide a short description of necessary concepts from the theory of convex optimization. For a broad treatment of the subject we refer to standard textbooks in the literature [Boyd and Vandenberghe, 2004; Rockafellar, 1970].

We will consider functions f mapping $\mathbb{R}^d$ to the extended real line, i.e., $\mathbb{R} \cup \{\pm\infty\}$. A function $f : \mathbb{R}^d$ is called *proper* if it never takes the value $-\infty$ and is finite at some point. Given a function f, we define its *effective domain* and *epigraph* as follows

$$\begin{aligned} \mathrm{dom} f &\triangleq \{\boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) < +\infty\} \\ \mathrm{epi} f &\triangleq \{(\boldsymbol{x}, r) \in \mathbb{R}^d \times \mathbb{R} : f(\boldsymbol{x}) \leq r)\} \ . \end{aligned}$$

Definition 2.8 (Convex set)**.** *$V \subset \mathbb{R}^d$ is convex if for any $\boldsymbol{x}, \boldsymbol{y} \in V$ and any $\lambda \in (0,1)$, we have $\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in V$.*



In words, the above definition says that a set is convex if for any two points $\boldsymbol{x}, \boldsymbol{y}$ in the set V, the entire line segment between them is contained in the set V. This definition allows us to next define convex functions.

Definition 2.9 (Convex function)**.** *A function $f : \mathbb{R}^d \to \mathbb{R} \cup \{\pm\infty\}$ is* convex *if* epi$f$ *is a convex set in $\mathbb{R}^d \times \mathbb{R}$.*

An equivalent definition is given below.

Definition 2.10 ([Rockafellar, 1970, Theorem 4.1])**.** *Let $f : \mathbb{R}^d \to (-\infty, +\infty]$ and* dom$f$ *is a convex set. Then f is convex iff, for any $0 < \lambda < 1$, we have*

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}), \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \text{dom} f \ . \tag{2.1}$$

In other words, the function always lies below the line segment joining $f(\boldsymbol{x})$ and $f(\boldsymbol{y})$.

Next, we show how one can characterize convexity in terms of derivatives. The following proposition is very useful, since it gives a way to verify if a smooth function is convex, by showing that its Hessian is positive semi-definite everywhere.

Proposition 2.11 ([Drusvyatskiy, 2020, Theorem 3.8])**.** *The following are equivalent for a C^1-smooth function $f : V \to \mathbb{R}$ defined on a convex set $V \subset \mathbb{R}^d$*

1. ***Convexity.*** *f is convex.*
2. ***Gradient inequality.*** *$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle$, for all $\boldsymbol{x}, \boldsymbol{y} \in V$.*
3. ***Monotonicity.*** *$\langle \nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \geq 0$, for all $\boldsymbol{x}, \boldsymbol{y} \in V$.*

Furthermore, if f is C^2-smooth, then the following property can be added to the list

4. *the relation $\nabla^2 f(\boldsymbol{x}) \succeq 0$ holds for all $\boldsymbol{x} \in V$.*

Note that property 2 above allows us to construct linear lower bounds to the function f. Below we provide some common examples of convex functions.

Example 2.1. *The indicator function of the set V, $i_V : \mathbb{R}^d \to (-\infty, +\infty]$, is defined as*

$$i_V(\boldsymbol{x}) = \begin{cases} 0, & \boldsymbol{x} \in V, \\ +\infty, & \text{otherwise.} \end{cases}$$

Example 2.2. *Affine functions: $f(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{z} \rangle + b$.*

Example 2.3. *All norms defined in Section 2.1.1 are convex.*

An important property of convex functions is the first-order optimality condition.

Theorem 2.12. *Let V a non-empty convex set, $\boldsymbol{x}^* \in V$, and f a convex function, differentiable over an open set that contains V. Then $\boldsymbol{x}^* = \text{argmin}_{\boldsymbol{x} \in V} f(\boldsymbol{x})$ if and only if*

$$\langle \nabla f(\boldsymbol{x}^*), \boldsymbol{y} - \boldsymbol{x}^* \rangle \geq 0, \quad \forall \boldsymbol{y} \in V \ . \tag{2.2}$$

Finally, when dealing with probability distributions, it is often useful to use the Jensen's inequality.

Theorem 2.13. *Let $f : \mathbb{R}^d \to (-\infty, +\infty]$ be a measurable convex function and $\boldsymbol{x} \in \mathbb{R}^d$ be a random vector on some probability space such that $\mathbb{E}[\boldsymbol{x}]$ exists and $\boldsymbol{x} \in$ dom f with probability 1. Then,*

$$E[f(\boldsymbol{x})] \geq f(E[\boldsymbol{x}]) \ .$$

2.2.1 Subgradients

Often, the functions encountered in machine learning are not differentiable. For example, this is the case for the hinge loss used for classification, $\ell_t(\boldsymbol{x}) = \max(1 - y_t \langle \boldsymbol{z}_t, \boldsymbol{x} \rangle, 0)$. However, we can still define a quantity similar to the gradients defined in the previous section.

Definition 2.14 (Subdifferential)**.** *Consider a function $f : \mathbb{R}^d \to R$ and a point $\boldsymbol{x}$, with f(x) finite. Then a vector $\boldsymbol{g} \in \mathbb{R}^d$ is called a subgradient of f in $\boldsymbol{x}$ if the inequality holds:*

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x} \rangle + o(\|\boldsymbol{y} - \boldsymbol{x}\|), \quad \text{as } \boldsymbol{y} \to \boldsymbol{x} \ . \tag{2.3}$$

The set of all such vectors $\boldsymbol{g}$ is called the *subdifferential* of f at x, and we denote it by $\partial f(\boldsymbol{x})$. Note that in this definition we simply replace the equality in the definition of the gradient by an inequality.

Example 2.4 (Hinge loss)**.** *Consider* $\ell(\boldsymbol{x}) = \max(1 - \langle \boldsymbol{z}, \boldsymbol{x} \rangle, 0)$*, with* $\boldsymbol{z} \in \mathbb{R}^d$*. We have that*

$$\partial \ell(\boldsymbol{x}) = \begin{cases} \{\boldsymbol{0}\}, & 1 - \langle \boldsymbol{z}, \boldsymbol{x} \rangle \leq 0, \\ \{-\alpha \boldsymbol{z} | \alpha \in [0,1]\}, & 1 - \langle \boldsymbol{z}, \boldsymbol{x} \rangle = 0, \\ \{-\boldsymbol{z}\}, & \text{otherwise.} \end{cases}$$

Example 2.5 (Indicator function)**.** *Consider a non-empty set* V*. By definition,* $\boldsymbol{g} \in \partial i_V(\boldsymbol{x})$ *if*

$$i_V(\boldsymbol{y}) \geq i_V(\boldsymbol{x}) + \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x} \rangle, \ \forall \boldsymbol{y} \in \mathbb{R}^d \ .$$

This condition implies that $\boldsymbol{x} \in V$ *and* $0 \geq \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x} \rangle, \forall \boldsymbol{y} \in V$*. The set of all* $\boldsymbol{g}$ *that satisfies the above inequality is called the* normal cone *of* V *at* $\boldsymbol{x}$*.*

It can be shown that any convex function admits a subgradient at every point in the relative interior of its domain. On the other hand, even a convex function may fail to admit a subgradient on the boundary of its domain. One example is the function $h(x) = -\sqrt{x}$ [Drusvyatskiy, 2020, Exercise 3.32].

Theorem 2.15 (Existence of subgradients, [Drusvyatskiy, 2020, Theorem 3.38])**.** *Consider a proper convex function* $f : \mathbb{R}^d \to \mathbb{R}$*. Then the subdifferential* $\partial f(\boldsymbol{x})$ *is nonempty at every point* $\boldsymbol{x} \in \text{int dom} f$*.*

The next theorem provides an upper bound to the norm of the subgradients for convex Lipschitz functions.

Theorem 2.16 (Orabona [2019])**.** *Let* $f : \mathbb{R}^d \to (-\infty, +\infty]$ *proper and convex. Then,* f *is* L*-Lipschitz in* int dom f *w.r.t. a norm* $\|\cdot\|$ *iff for all* $\boldsymbol{x} \in$ int dom f *and* $\boldsymbol{g} \in \partial f(\boldsymbol{x})$ *we have* $\|\boldsymbol{g}\|_\star \leq L$*.*

2.2.2 Strong convexity

In this section, we provide the notion of *strong-convexity*. Contrarily to the notion of convexity, which provides a linear lower bound to a convex function, *strong-convexity* gives a *quadratic* lower bound.

Definition 2.17 (Strong-convexity)**.** *A proper function* $f : \mathbb{R}^d \to (-\infty, +\infty]$ *is* μ-strongly convex *over a convex set* $V \subseteq \text{int dom} f$ *w.r.t.* $\|\cdot\|$ *if* $\forall \boldsymbol{x}, \boldsymbol{y} \in V$ *and* $\boldsymbol{g} \in \partial f(\boldsymbol{x})$*, we have*

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{\mu}{2} \|\boldsymbol{x} - \boldsymbol{y}\|^2 \ .$$

An equivalent characterization of strong-convexity can be given through the Hessian of the function f, as reported next.

Theorem 2.18 (Shalev-Shwartz [2007], Lemma 14)**.** *Let* $V \subseteq \mathbb{R}^d$ *convex and* $f : V \to \mathbb{R}$ *twice differentiable. Then, a sufficient condition for* μ*-strong convexity in* V *w.r.t.* $\|\cdot\|$ *is that for all* $\boldsymbol{x}, \boldsymbol{y} \in V$ *we have* $\langle \nabla^2 f(\boldsymbol{x}) \boldsymbol{y}, \boldsymbol{y} \rangle \geq \mu \|\boldsymbol{y}\|^2$*, where* $\nabla^2 f(\boldsymbol{x})$ *is the Hessian matrix of* f *at* $\boldsymbol{x}$*.*

Example 2.6. *Let* $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$*. Using Theorem 2.18 it is straightforward to show that* f *is 1-strongly convex w.r.t.* $\|\cdot\|_2$*.*

Also, it can be shown that the negative *Shannon entropy* is 1-strongly convex in the unit simplex.

Theorem 2.19 ([Shalev-Shwartz, 2012, Example 2.5])**.** *The function* $\psi(\boldsymbol{x}) = \sum_{i=1}^d x_i \ln x_i$ *is* $1/B$*-strongly convex with respect to the* ℓ_1 *norm over the set* $V = \{\boldsymbol{x} \in \mathbb{R}_+^d : \|\boldsymbol{x}\|_1 = D\}$*.*

Proof. Using again Theorem 2.18, we have that

$$\langle \nabla^2 \psi(\boldsymbol{y}) \boldsymbol{x}, \boldsymbol{x} \rangle = \frac{1}{\|\boldsymbol{y}\|_1} \left(\sum_{i=1}^d y_i \right) \left(\sum_{i=1}^d \frac{x_i^2}{y_i} \right) \geq \frac{1}{\|\boldsymbol{y}\|_1} \left(\sum_{i=1}^d \sqrt{y_i} \frac{|x_i|}{\sqrt{y_i}} \right)^2 = \frac{\|\boldsymbol{x}\|_1^2}{\|\boldsymbol{y}\|_1} = \frac{1}{D} \|\boldsymbol{x}\|_1^2$$

where we used Cauchy-Schwartz in second inequality. □

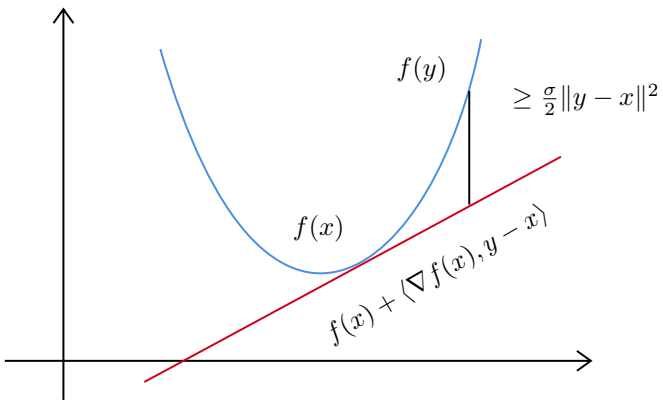


FIGURE 2.3: Strong convexity illustrated.

2.2.3 Fenchel Conjugate

The following concepts about convex conjugates will be useful.

Definition 2.20 (Fenchel conjugate)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}[-\infty, +\infty]$. The **Fenchel conjugate** of f is the function $f^* : \mathbb{R}^d \to (-\infty, +\infty]$ defined by*

$$f^*(\boldsymbol{\theta}) \triangleq \sup_{\boldsymbol{x}\in\mathbb{R}^d} \langle \boldsymbol{\theta}, \boldsymbol{x}\rangle - f(\boldsymbol{x}) \ .$$

If $f : \mathbb{R}^d \to (-\infty, +\infty]$ is a proper function, then by the definition of conjugate one can easily get the **Fenchel-Young inequality**:

$$\langle \boldsymbol{\theta}, \boldsymbol{x}\rangle \leq f(\boldsymbol{x}) + f^*(\boldsymbol{\theta}),\ \forall \boldsymbol{x}, \boldsymbol{\theta} \in \mathbb{R}^d \ .$$

Example 2.7 ([Orabona, 2019, Example 5.7])**.** *Consider the function $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|^2$, where $\|\cdot\|$ is a norm in $\mathbb{R}^d$, with dual norm $\|\cdot\|_\star$. Then, its Fenchel conjugate is $f^\star(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_\star^2$.*

Lemma 2.21 ([Orabona, 2019, Lemma 5.8])**.** *Let f be a function and let $f^\star$ be its Fenchel conjugate. For $a > 0$ and $b \in \mathbb{R}$, the Fenchel conjugate of $g(\boldsymbol{x}) = af(\boldsymbol{x}) + b$ is $g^\star(\boldsymbol{\theta}) = af^\star(\boldsymbol{\theta}/a) - b$.*

2.2.4 Bregman Divergences

We conclude this chapter by defining a fundamental ingredient which we will use in the rest of the thesis.

We first define the stronger notion of strict-convexity.

Definition 2.22 (Strictly Convex Function)**.** *Let $V \subseteq \mathbb{R}^d$ be a convex set and $f : V \subseteq \mathbb{R}^d \to \mathbb{R}$. The function f is* strictly convex *if*

$$f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) < \alpha f(\boldsymbol{x}) + (1-\alpha)f(\boldsymbol{y}),\ \forall \boldsymbol{x}, \boldsymbol{y} \in V, \boldsymbol{x} \neq \boldsymbol{y}, 0 < \alpha < 1 \ .$$

We now define the Bregman divergence between two points $\boldsymbol{x}$ and $\boldsymbol{y}$. Note that this function is similar to a notion of distance.

Definition 2.23 (Bregman Divergence)**.** *Let $\psi : V \to \mathbb{R}$ be strictly convex and continuously differentiable on* int V*. The* Bregman Divergence *w.r.t. ψ is $B_\psi : V \times \operatorname{int} V \to \mathbb{R}_+$ defined as*

$$B_\psi(\boldsymbol{x}, \boldsymbol{y}) \triangleq \psi(\boldsymbol{x}) - \psi(\boldsymbol{y}) - \langle \nabla\psi(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle \tag{2.4}$$

and ψ is called the distance generating function.

In the rest of this thesis, we assume that ψ is strongly convex w.r.t. a norm $\|\cdot\|$ in $\operatorname{int} X$. We also assume w.l.o.g. the strong convexity constant to be 1, which implies

$$B_\psi(\boldsymbol{x}, \boldsymbol{y}) \geq \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2, \quad \forall \boldsymbol{x} \in X, \boldsymbol{y} \in \operatorname{int} X \ . \tag{2.5}$$

Below we list some properties of the Bregman divergences, all of them can be easily verified from the definition given in Eq. (2.4).

Proposition 2.24. *Let B_ψ be defined as in Definition 2.23. Then, the following hold*

- ***Convexity**: $B_\psi(\boldsymbol{x}, \boldsymbol{y})$ is convex as a function of $\boldsymbol{x}$ for any $\boldsymbol{y} \in \operatorname{dom}\nabla\psi$.*
- ***Nonnegativity**: $B_\psi(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom}\psi$.*
- ***Asymmetry**: $B_\psi(\boldsymbol{x}, \boldsymbol{y}) \neq B_\psi(\boldsymbol{y}, \boldsymbol{x})$*

Below we list some examples of Bregman divergences.

Squared L_2 distance. With $\psi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$, then $\nabla\psi(\boldsymbol{x}) = \boldsymbol{x}$ and we have that

$$B_\psi(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2 + \frac{1}{2}\|\boldsymbol{y}\|_2^2 - \langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{y} \rangle = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \ ,$$

which is a squared Euclidean distance.

KL divergence. Let $V = [0, +\infty)^d$ and $\psi(\boldsymbol{x}) = \sum_{i=1}^d (x_i \ln x_i - x_i)$, with $\operatorname{dom}\psi = V$ and $0 \ln 0 = 0$. Then, for any $\boldsymbol{y} \in (0, +\infty)^d$, we have that $\nabla\psi(\boldsymbol{y}) = \ln \boldsymbol{y}$ and

$$B_\psi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^d (x_i \ln x_i - x_i) - \sum_{i=1}^d (y_i \ln y_i - y_i) - \sum_{i=1}^d (x_i - y_i) \ln y_i = \sum_{i=1}^d x_i \ln \frac{x_i}{y_i} + \sum_{i=1}^d (y_i - x_i) \ .$$

Note that if $\boldsymbol{x}, \boldsymbol{y}$ are in the d-dimensional (unit) simplex, then $B_\psi(\boldsymbol{x}, \boldsymbol{y})$ becomes the well known KL-divergence between the probability vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

Chapter 3

Online Linear Optimization

In this chapter, we describe the dominant approaches taken to design algorithm in Online Convex Optimization. We can mainly classify the existent algorithms as belonging in two classes: they are either derived using *Online Mirror Descent* (OMD) or *Follow The Regularized Leader* (FTRL). While this algorithms are in general different, it can be shown that in some particular cases their updates coincide. This is also reflected in their regret bounds. Indeed, while their analysis is different, the obtained rates of convergence are similar. Crucially, using the notions from convex optimization given in the previous chapter, we will be able to derive bounds in a simple and elegant way. In the middle, we will also describe a recent algorithm from the literature which can be seen as modified version of OMD, which solves one important issue related to unbounded domains for OMD. The description of the algorithms follows heavily Orabona [2019] and we refer to it for a more accurate and complete treatment of the subject. Neverthless, we report here some of the proofs for completeness, since the notions described in this chapter will serve as a motivation for the subsequent chapters.

3.1 A building block: OMD

In this section we explore the Online Mirror Descent algorithm, which is the online counterpart to the Mirror Descent algorithm from classical convex optimization [Beck and Teboulle, 2003]. The main idea behind OMD is to take in every iteration a step in the direction of the negative subgradient of the current loss function. Importantly, this step takes place in a dual space, where the gradients "live", contrarily to the primal space where the iterates "reside". This idea generalizes the Gradient Descent update rule, which is just a particular case of Mirror Descent where the primal and dual space coincide.

Formally, consider a set $V \subset X \subseteq \mathbb{R}^d$. The Online Mirror Descent [Warmuth and Jagota, 1997; Beck and Teboulle, 2003] update over V is

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \arg\min_{\boldsymbol{x}\in V} \; B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) + \eta_t(\ell_t(\boldsymbol{x}_t) + \langle \boldsymbol{g}_t, \boldsymbol{x} - \boldsymbol{x}_t\rangle) \\ &= \arg\min_{\boldsymbol{x}\in V} \; B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) + \eta_t\langle \boldsymbol{g}_t, \boldsymbol{x}\rangle, \end{aligned} \tag{3.1}$$

for $\boldsymbol{g}_t \in \partial\ell_t(\boldsymbol{x}_t)$ received as feedback. In words, OMD updates the solution minimizing a first-order approximation of the received loss, ℓ_t, around the predicted point, $\boldsymbol{x}_t$, constrained to be not too far from the predicted point measured with the Bregman divergence. The algorithm is depicted in Algorithm 1.

To make the update rule not pathological, we need an additional assumption. Indeed, from the update in Equation (3.1) one could have $\boldsymbol{x}_{t+1}$ on the boundary of V, which will require to evaluate $B_\psi(\boldsymbol{x}, \boldsymbol{x}_{t+1})$ in the next iterate. Since $V \subseteq X$, we might end up on the boundary of X, where the Bregman is not defined. Therefore, either one of the next two properties has to be satisfied:

$$\lim_{\boldsymbol{x}\to\partial X} \|\nabla\psi(\boldsymbol{x})\|_2 = +\infty \;, \tag{3.2}$$

$$V \subseteq \text{int}X \;. \tag{3.3}$$

Note that often this assumption in the literature is made through the requirement of ψ being a *Legendre* function (see for example [Cesa-Bianchi and Lugosi, 2006, Chapter 11]). Once one

Algorithm 1 Online Mirror Descent (OMD)

Require: Non-empty closed convex set $V \subset X \subset \mathbb{R}^d$, $\psi : X \to \mathbb{R}$, $\eta_t > 0$, $\boldsymbol{x}_1 \in V$
1: **for** $t = 1, \dots, T$ **do**
2: Output $\boldsymbol{x}_t \in V$
3: Receive $\ell_t : \mathbb{R}^d \to \mathbb{R}$ and pay $\ell_t(\boldsymbol{x}_t)$
4: Set $\boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)$
5: Update $\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x} \in V} \; B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) + \eta_t \langle \boldsymbol{g}_t, \boldsymbol{x} \rangle$
6: **end for**

of the two conditions above is verified, the update is well defined, meaning that $\boldsymbol{x}_{t+1}$ exists and is unique. We will present this result in Lemma 3.1. Before giving the lemma, we show an alternative way of writing the OMD update which will help state the result.

A two step process. Often in the online learning literature, the update rule given in Equation (3.1) is splitted in two-steps. The idea is to use first solve the optimization problem over the entire space and then going back to the set of interest, with a projection with respect the Bregman divergence at hand. The next lemma, stated without proof, provides a formal argument.

Lemma 3.1 ([Orabona, 2019, Theorem 6.13])**.** *Let $f : \mathbb{R}^d \to (-\infty, +\infty]$ proper, closed, strictly convex, and differentiable in* int dom$f$ *. Also, let $V \subset \mathbb{R}^d$ a non-empty, closed convex set with $V \cap \text{dom} f \neq \emptyset$ and assume that $\tilde{\boldsymbol{y}} = \text{argmin}_{\boldsymbol{z} \in \mathbb{R}^d} f(\boldsymbol{z})$ exists and $\tilde{\boldsymbol{y}} \in$* int dom$f$*. Denote by $\boldsymbol{y}' = \text{argmin}_{\boldsymbol{z} \in V} B_f(\boldsymbol{z}, \tilde{\boldsymbol{y}})$. Then the following hold:*

1. *$\boldsymbol{y} = \text{argmin}_{\boldsymbol{z} \in V} f(\boldsymbol{z})$ exists and is unique.*
2. *$\boldsymbol{y} = \boldsymbol{y}'$.*

To see how we can implement a two steps updated for OMD, let $\psi'(\boldsymbol{x}) = \psi(\boldsymbol{x}) + \langle \boldsymbol{g} - \nabla\psi(\boldsymbol{y}), \boldsymbol{x} \rangle$. We have that

$$\begin{aligned}
B_{\psi'}(\boldsymbol{x}, \boldsymbol{y}) &= \psi'(\boldsymbol{x}) - \psi'(\boldsymbol{y}) - \langle \nabla\psi'(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \\
&= \psi(\boldsymbol{x}) + \langle \boldsymbol{g} - \nabla\psi(\boldsymbol{y}), \boldsymbol{x} \rangle - \psi(\boldsymbol{y}) - \langle \boldsymbol{g} - \nabla\psi(\boldsymbol{y}), \boldsymbol{y} \rangle - \langle \cancel{\nabla\psi(\boldsymbol{y})} + \boldsymbol{g} - \cancel{\nabla\psi(\boldsymbol{y})}, \boldsymbol{x} - \boldsymbol{y} \rangle \\
&= \psi(\boldsymbol{x}) - \psi(\boldsymbol{y}) - \langle \nabla\psi(\boldsymbol{y}) + \cancel{\boldsymbol{g}} - \cancel{\boldsymbol{g}}, \boldsymbol{x} - \boldsymbol{y} \rangle \\
&= \psi(\boldsymbol{x}) - \psi(\boldsymbol{y}) - \langle \nabla\psi(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \\
&= B_\psi(\boldsymbol{x}, \boldsymbol{y}) \; .
\end{aligned}$$

Therefore, having $f = \psi'$ with $\boldsymbol{y} = \boldsymbol{x}_t$ and $\boldsymbol{g} = \eta_t \boldsymbol{g}_t$ we get

$$\begin{aligned}
\tilde{\boldsymbol{x}}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^d} \psi'(\boldsymbol{x}) \\
&= \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^d} \psi(\boldsymbol{x}) + \langle \eta_t \boldsymbol{g}_t - \nabla\psi(\boldsymbol{x}_t), \boldsymbol{x} \rangle \\
&= \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^d} \langle \eta_t \boldsymbol{g}_t, \boldsymbol{x} \rangle + \psi(\boldsymbol{x}) - \psi(\boldsymbol{x}_t) - \langle \nabla\psi(\boldsymbol{x}_t), \boldsymbol{x} - \boldsymbol{x}_t \rangle \\
&= \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^d} \langle \eta_t \boldsymbol{g}_t, \boldsymbol{x} \rangle + B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) \; .
\end{aligned}$$

And $\boldsymbol{x}'_{t+1} = \text{argmin}_{\boldsymbol{x} \in V} B_{\psi'}(\boldsymbol{x}, \tilde{\boldsymbol{x}}_{t+1}) = \text{argmin}_{\boldsymbol{x} \in V} B_\psi(\boldsymbol{x}, \tilde{\boldsymbol{x}}_{t+1})$.

To summarize, we have that

$$\tilde{\boldsymbol{x}}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^d} \eta_t \langle \boldsymbol{g}_t, \boldsymbol{x} \rangle + B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) \; , \tag{3.4}$$

$$\boldsymbol{x}'_{t+1} = \operatorname*{argmin}_{\boldsymbol{x} \in V} B_\psi(\boldsymbol{x}, \tilde{\boldsymbol{x}}_{t+1}) \; . \tag{3.5}$$

By using the result in Lemma 3.1, we have that $\boldsymbol{x}'_{t+1}$ coincides with the $\boldsymbol{x}_{t+1}$ in Equation (3.1).

Note that the Lemma 3.1 has two assumptions. First, $\tilde{\boldsymbol{y}} = \text{argmin}_{\boldsymbol{z} \in \mathbb{R}^d} f(\boldsymbol{z})$ must exist. This is necessary. Indeed, f being convex does not imply that a minimum exists. For example,

if we consider $f(x) = e^x$, then f does not have a minimum. The second assumption is needed for technical reasons in the proof of Lemma 3.1. Note that by having Equation (3.2), then the second assumption is automatically satisfied. Indeed, by the optimality condition in the definition of $\tilde{\boldsymbol{y}}$, we have that $\nabla f(\tilde{\boldsymbol{y}}) = \mathbf{0}$ but if $\|\nabla\psi(\boldsymbol{x})\|^2$ goes to infinity for $\boldsymbol{x} \to \partial X$, then $\tilde{\boldsymbol{y}} \in \operatorname{int} \operatorname{dom} f$.

Below, we give some examples of updates computed by using the OMD update rule. Probably the most known application of OMD is the online version of the Gradient Descent algorithm [Zinkevich, 2003], which can be seen as OMD with the squared Euclidean distance as Bregman divergence.

Example 3.1 (Online Gradient Descent)**.** *Let $\psi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2 + i_V(\boldsymbol{x})$, with $V = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\|_2 \leq D\}$, then according to the update in Equation (3.1) it is easy to show that*

$$\tilde{\boldsymbol{x}}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} \eta_t\langle \boldsymbol{g}_t, \boldsymbol{x}\rangle + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_t\|_2^2 = \boldsymbol{x}_t - \eta_t \boldsymbol{g}_t$$

$$\boldsymbol{x}_{t+1} = \Pi_V(\tilde{\boldsymbol{x}}_{t+1}) = \begin{cases} \tilde{\boldsymbol{x}}_{t+1}, & \|\tilde{\boldsymbol{x}}_{t+1}\|_2 \leq D \\ \frac{\tilde{\boldsymbol{x}}_{t+1}}{\|\tilde{\boldsymbol{x}}_{t+1}\|_2}, & \|\tilde{\boldsymbol{x}}_{t+1}\|_2 > D \end{cases}$$

where Π_V is the operation which denotes the Euclidean projection on the set V.

Another algorithm which is ubiquitous in the literature is the one using the so called exponential weigths update. In the following we derive its update rule through the two-step process.

Example 3.2 (Exponentiated gradient)**.** *Let $\psi(\boldsymbol{x}) = \sum_{i=1}^d x_i \ln x_i + i_{\mathbb{R}^d_+}(\boldsymbol{x})$, for each $\boldsymbol{x} \in \mathbb{R}^d$, where we assume $0 \ln 0 = 0$. Then, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d_+$, by definition of Bregman divergence we have*

$$\begin{aligned} B_\psi(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{i=1}^d (x_i \ln x_i - y_i \ln y_i - (1 + \ln y_i)(x_i - y_i)) \\ &= \sum_{i=1}^d x_i \ln \frac{x_i}{y_i} + \|\boldsymbol{y}\|_1 - \|\boldsymbol{x}\|_1 \ . \end{aligned}$$

In particular, if we consider the d-dimensional simplex, i.e., $V = \{\boldsymbol{x} \in \mathbb{R}^d_+ : \|\boldsymbol{x}\|_1 = 1\}$, then the function above becomes the well-known KL-divergence between the probability distributions $\boldsymbol{x}$ and $\boldsymbol{y}$.

In order to compute the update, note that the condition in Equation (3.4) is equivalent to have $\tilde{\boldsymbol{x}}_{t+1}$ such that $\nabla\psi(\tilde{\boldsymbol{x}}_{t+1}) = \nabla\psi(\boldsymbol{x}_t) - \eta_t \boldsymbol{g}_t$, that is

$$\ln(\tilde{x}_{t+1,i}) + 1 = \ln(x_{t,i}) + 1 - \eta_t g_{t,i} \ ,$$

for all $i = 1, \ldots, d$, which leads to

$$\tilde{x}_{t+1,i} = x_{t,i} \exp(-\eta_t g_{t,i}) \ .$$

The last step is to implement the Bregman projection on the simplex. By using the first-order optimality condition and the fact that $\partial(f + g) = \partial f + \partial g$ from Rockafellar [1970, Theorem 23.8] (assuming $\operatorname{int} X \cap V \neq \emptyset$, which is verified in this case), from Equation (3.5) we get

$$\mathbf{0} \in \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\tilde{\boldsymbol{x}}_{t+1}) + \partial i_V(\boldsymbol{x}_{t+1}) \ .$$

Now, we claim that $\boldsymbol{x}_{t+1} = \tilde{\boldsymbol{x}}_{t+1}/\|\tilde{\boldsymbol{x}}_{t+1}\|_1$. Note that $\|\boldsymbol{x}_{t+1}\|_1 = 1$ and therefore is in V. Furthermore, we have that for any $i = 1, \ldots, d$,

$$(\nabla\psi)_i(\tilde{\boldsymbol{x}}_{t+1}) - (\nabla\psi)_i(\boldsymbol{x}_{t+1}) = (\ln \tilde{x}_{t+1,i} + 1) - (\ln x_{t+1,i} + 1) = \ln \|\tilde{\boldsymbol{x}}_{t+1}\|_1 \ .$$

Therefore, for every $\boldsymbol{y} \in V$ we have

$$\langle \ln \|\tilde{x}_{t+1,i}\|_1 \vec{\mathbf{1}}, \boldsymbol{y} - \boldsymbol{x}_{t+1}\rangle = \ln \|\tilde{x}_{t+1,i}\|_1 (\|\boldsymbol{y}\|_1 - \|\boldsymbol{x}_{t+1}\|_1) = 0 \ ,$$

which implies $\nabla\psi(\tilde{\boldsymbol{x}}_{t+1}) - \nabla\psi(\boldsymbol{x}_{t+1}) \in \partial i_V(\boldsymbol{x}_{t+1})$ *by the definition of normal cone (see Example 2.5).*

3.1.1 Regret Analysis

We can next analyze the regret of OMD. First, we need the so called Prox-lemma [Beck and Teboulle, 2003], which gives an upper bound to the instantaneous regret at time t, $\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u})$.

Lemma 3.2 ([Orabona, 2019, Lemma 6.7])**.** *Let B_ψ the Bregman divergence w.r.t. $\psi : X \to R$ and assume ψ to be 1-strongly convex with respect to $\|\cdot\|$ in V. Let $V \subseteq X$ a non-empty closed convex set. Set $\boldsymbol{g}_t \in \partial\ell_t(\boldsymbol{x}_t)$. Assume Equation (3.2) or Equation (3.3) hold and $\eta_t > 0$. Then, $\forall \boldsymbol{u} \in V$ the following inequality holds*

$$\eta_t(\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u})) \leq \eta_t\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u}\rangle \leq B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) + \frac{\eta_t^2}{2}\|\boldsymbol{g}_t\|_\star^2 . \tag{3.6}$$

Proof. From the first-order optimality condition of the update rule of OMD in Equation (3.1) we have

$$\langle \eta_t \boldsymbol{g}_t + \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_t), \boldsymbol{u} - \boldsymbol{x}_{t+1}\rangle \geq 0, \ \forall \boldsymbol{u} \in V . \tag{3.7}$$

from which we derive $\langle \eta_t \boldsymbol{g}_t, \boldsymbol{x}_{t+1} - \boldsymbol{u}\rangle \leq \langle \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_t), \boldsymbol{u} - \boldsymbol{x}_{t+1}\rangle$.

Now, we can consider the instantaneous regret at time t. Note that $\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u}) \leq \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u}\rangle$. Furthermore,

$$\begin{aligned}
\langle \eta_t \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u}\rangle &= \langle \eta_t \boldsymbol{g}_t, \boldsymbol{x}_{t+1} - \boldsymbol{u}\rangle + \langle \eta_t \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1}\rangle \\
&\leq \langle \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_t), \boldsymbol{u} - \boldsymbol{x}_{t+1}\rangle + \langle \eta_t \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1}\rangle \\
&= B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) + \langle \eta_t \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1}\rangle \\
&\leq B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) + \frac{\eta_t^2}{2}\|\boldsymbol{g}_t\|_\star^2 + \frac{1}{2}\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\|^2 \\
&\leq B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) + \frac{\eta_t^2}{2}\|\boldsymbol{g}_t\|_\star^2,
\end{aligned}$$

where the second inequality derives from the fact that $\langle \boldsymbol{x}, \boldsymbol{y}\rangle \leq \frac{1}{2}\|\boldsymbol{x}\|^2 + \frac{1}{2}\|\boldsymbol{y}\|_\star^2$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ (it follows easily from the Fenchel-Young inequality), while in the last inequality we used the lower bound to the Bregman divergence given in Equation (2.5) since ψ is 1-strongly convex. □

Armed with the previous lemma, it is easy to prove a regret bound for OMD. Indeed all we need to do is to sum the instantaneous regret over all the time horizon T, as shown in the next theorem.

Theorem 3.3 ([Orabona, 2019, Theorem 3.8])**.** *Let $D^2 = \max_{1\leq t\leq T} B_\psi(\boldsymbol{u}, \boldsymbol{x}_t)$. Set $\boldsymbol{x}_1 \in V$ such that ψ is differentiable in $\boldsymbol{x}_1$. Assume $(\eta)_{t=1}^T$ is a non-increasing sequence. Then, under the assumptions of Lemma 3.2 and for any $\boldsymbol{u} \in V$, the following regret bounds hold*

$$\sum_{t=1}^T (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u})) \leq \frac{D^2}{\eta_T} + \frac{1}{2}\sum_{t=1}^T \eta_t \|\boldsymbol{g}_t\|_\star^2 . \tag{3.8}$$

Furthermore, if $\eta_t = \eta$ is constant, i.e. $\eta_t = \eta \ \forall t = 1, \dots, T$, we have

$$\sum_{t=1}^T (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u})) \leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta} + \frac{\eta}{2}\sum_{t=1}^T \|\boldsymbol{g}_t\|_\star^2 . \tag{3.9}$$

Proof. Fix $\boldsymbol{u} \in V$. Dividing the inequality in Lemma 3.2 by η_t and summing from $t = 1, \dots, T$, we get

$$\sum_{t=1}^T (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u})) \leq \sum_{t=1}^T \left(\frac{1}{\eta_t} B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - \frac{1}{\eta_t} B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1})\right) + \sum_{t=1}^T \frac{\eta_t}{2}\|\boldsymbol{g}_t\|_\star^2$$

$$
\begin{aligned}
&= \frac{1}{\eta_1} B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) - \frac{1}{\eta_T} B_\psi(\boldsymbol{u}, \boldsymbol{x}_{T+1}) + \sum_{t=1}^{T-1} \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t} \right) B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) + \sum_{t=1}^{T} \frac{\eta_t}{2} \|\boldsymbol{g}_t\|_\star^2 \\
&\leq \frac{1}{\eta_1} D^2 + D^2 \sum_{t=1}^{T-1} \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t} \right) + \sum_{t=1}^{T} \frac{\eta_t}{2} \|\boldsymbol{g}_t\|_\star^2 \\
&= \frac{1}{\eta_1} D^2 + D^2 \left(\frac{1}{\eta_T} - \frac{1}{\eta_1} \right) + \sum_{t=1}^{T} \frac{\eta_t}{2} \|\boldsymbol{g}_t\|_\star^2 \\
&= \frac{D^2}{\eta_T} + \sum_{t=1}^{T} \frac{\eta_t}{2} \|\boldsymbol{g}_t\|_\star^2 \,.
\end{aligned}
$$

The proof for the second part follows the same steps. □

As we can see, there are two fundamental components in the regret bounds from Theorem 3.3. First, the choice of the Bregman divergence and hence of the function ψ, which is used as a distance measure between the current and next model, but at the same time induces a dual norm used to "measure" the gradients. Therefore, different choices of ψ will lead to different regret bounds.

Second, the regret bound in Theorem 3.3 is not in closed from but crucially depends on the tuning of the learning rate η_t. We can inspect the case of constant learning rate first. The setting of η which minimizes Equation (3.9) is the following

$$
\eta = \sqrt{\frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\sum_{t=1}^{T} \|\boldsymbol{g}_t\|_\star^2}} \,.
$$

Unfortunately, neither $B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)$ nor $\sum_{t=1}^{T} \|\boldsymbol{g}_t\|_\star^2$ are known prior to the game starting. However we can still provide a worst-case upper bound, making the right assumptions.

Corollary 3.4. *Set $\boldsymbol{x}_1 \in V$ such that ψ is differentiable in $\boldsymbol{x}_1$ and let $B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) \leq D^2$. Assume the losses to be L-Lipschitz w.r.t. the norm $\|\cdot\|$ induced by ψ. Then, under the assumptions of Lemma 3.2, for all $\boldsymbol{u} \in V$ Algorithm 1 with $\eta_t = \eta = \frac{D}{L}\sqrt{\frac{2}{T}}$ for all t incurs regret bounded as*

$$
R_T(\boldsymbol{u}) \leq DL\sqrt{2T} \,.
$$

Proof. From Equation (3.9),

$$
R_T(\boldsymbol{u}) \leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|\boldsymbol{g}_t\|_\star^2 \leq \frac{D^2}{\eta} + \frac{\eta}{2} L^2 T = DL\sqrt{2T} \,,
$$

where in the second inequality we used Theorem 2.16 for bounding the norm of the subgradients and the fact that the Bregman divergence between $\boldsymbol{u}$ and $\boldsymbol{x}_1$ is bounded over V. □

With time-varying learning rates, the situation is more delicate. Intuitively, a possibility would be to set it is $\eta_t \propto 1/\sqrt{t}$. In this way, we can see immediately that the first term is roughly $\mathcal{O}(\sqrt{T})$. For the sum over time in the second term, we can use the following lemma.

Lemma 3.5 ([Cesa-Bianchi and Lugosi, 2006, Lemma 11.8])**.** *Let $a, x_1, \ldots, x_n$ be nonnegative real numbers. Then*

$$
\sum_{i=1}^{n} \frac{x_i}{\sqrt{a + \sum_{j=1}^{i} x_j}} \leq \left(\sqrt{a + \sum_{i=1}^{n} x_i} - \sqrt{a} \right)
$$

Using this result, we can provide a regret upper bound in the case of time varying learning rates.

Algorithm 2 Dual-Stabilized OMD

Require: Non-empty closed convex set $V \subset X \subset \mathbb{R}^d$, $\psi : X \to \mathbb{R}$, $\eta_t > 0$, $\boldsymbol{x}_1 \in V$, $\gamma_t > 0$
1: **for** $t = 1, \dots, T$ **do**
2: Output $\boldsymbol{x}_t \in V$
3: Receive $\ell_t : \mathbb{R}^d \to \mathbb{R}$ and pay $\ell_t(\boldsymbol{x}_t)$
4: Update $\boldsymbol{x}_{t+1} = \operatorname{argmin}_{\boldsymbol{x} \in V} \eta_t \langle \boldsymbol{g}_t, \boldsymbol{x} \rangle + \gamma_t B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) + (1 - \gamma_t) B_\psi(\boldsymbol{x}, \boldsymbol{x}_1)$
5: **end for**

Corollary 3.6. *Under the assumption of Theorem 3.3, for any $\boldsymbol{u} \in V$ Algorithm 1 with $\eta_t = \frac{D}{\sqrt{\sum_{s=1}^t \|\boldsymbol{g}_s\|_\star^2}}$ incurs regret bounded as*

$$R_T(\boldsymbol{u}) \leq 2D \sqrt{\sum_{t=1}^T \|\boldsymbol{g}_t\|_\star^2} \,. \tag{3.10}$$

Note that the algorithm deriving from the corollary above is not completely defined. Indeed, if $\sqrt{\sum_{s=1}^t \|\boldsymbol{g}_s\|_\star^2} = 0$, then the minimizer selected by the algorithm is not uniquely defined. In this case, we can just set $\boldsymbol{x}_t = \operatorname{argmin}_{\boldsymbol{x} \in V} \psi(\boldsymbol{x})$.

While the regret bound achieved by the version with fixed learning rate is tight (since without any further assumption a lower bound of $\Omega(T)$ holds), it is a worst-case result which might be overly pessimistic if the sequence of loss functions is not truly adversarial. On the other hand, with the online tuning of the learning rate given in the above corollary, the algorithm could potentially suffer a much smaller regret. In fact, a similar bound Equation (3.10) could be obtained by the algorithm with fixed learning rate by using the so called doubling-trick (see for example Cesa-Bianchi and Lugosi [2006, Section 2.3]), but we will not explore this technique here.

To conclude, we would like to point out that the time-varying version of OMD requires the Bregman divergence between $\boldsymbol{u}$ and $\boldsymbol{x}_t$ to be bounded at any point in time. Unfortunately this requirement forces the algorithm to be employed only on domains bounded w.r.t the Bregman divergence at hand. This is not due to a fault in the analysis, but it can be shown that there are some pathological cases where the algorithm fails to achieve a sublinear regret bound when employed on potentially unbounded domains, as done in Orabona and Pál [2018, Theorems 3,4].

3.2 Dual Stabilised OMD

In the following, we show a recent techique proved in Fang et al. [2020] for OMD to correct the issue highlighted in the last section. The main idea is to add in the OMD update from the previous section a term which depends on $\boldsymbol{x}_1$. Following the template of Theorem 3.3 we can prove a regret bound for this algorithm, similar in spirit to the argument followed by the authors of Fang et al. [2020]. The algorithm is depicted in Algorithm 2.

In order to prove a regret bound, we first provide a lemma similar to Lemma 3.2, which bound the regret of the algorithm in one step.

Lemma 3.7. *Let B_ψ the Bregman divergence w.r.t. $\psi : X \to R$ and assume ψ to be 1-strongly convex with respect to $\|\cdot\|$ in V. Let $V \subseteq X$ a non-empty closed convex set. Set $\boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)$. Assume Equation (3.2) or Equation (3.3) hold. Let γ_t a non-negative number such that $\gamma_t \leq 1$. Then, running Algorithm 2 $\forall \boldsymbol{u} \in V$ the following inequality holds*

$$\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u} \rangle \leq \frac{1}{\eta_t} \left(\gamma_t B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) + (1 - \gamma_t) B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) \right) + \frac{\eta_t}{2\gamma_t} \|\boldsymbol{g}_t\|_\star^2 \,. \tag{3.11}$$

Proof. From Algorithm 2, we have that the stabilized update is

$$\boldsymbol{x}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x} \in V} \eta_t \langle \boldsymbol{g}_t, \boldsymbol{x} \rangle + \gamma_t B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) + (1 - \gamma_t) B_\psi(\boldsymbol{x}, \boldsymbol{x}_1)$$

Therefore, from the optimality condition of the update, we get

$$\langle \eta_t \boldsymbol{g}_t + \gamma_t(\nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_t)) + (1-\gamma_t)(\nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_1)), \boldsymbol{u} - \boldsymbol{x}_{t+1}\rangle \geq 0 \ .$$

Rearranging terms, we get

$$\begin{aligned}
\langle \boldsymbol{g}_t, \boldsymbol{x}_{t+1} - \boldsymbol{u}\rangle &\leq \frac{\gamma_t}{\eta_t}\langle \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_t), \boldsymbol{u} - \boldsymbol{x}_{t+1}\rangle + \frac{1-\gamma_t}{\eta_t}\langle \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_1), \boldsymbol{u} - \boldsymbol{x}_{t+1}\rangle \\
&= \frac{\gamma_t}{\eta_t}\left(B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)\right) + \\
&\quad \frac{1-\gamma_t}{\eta_t}\left(B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_1)\right) \\
&\leq \frac{\gamma_t}{\eta_t}\left(B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)\right) + \frac{1-\gamma_t}{\eta_t}\left(B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1})\right) \ ,
\end{aligned}$$

where the last step derives from the fact that $\frac{1-\gamma_t}{\eta_t} \geq 0$ and $B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_1) \geq 0$.

Therefore, adding $\langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle$ on both sides and rearranging we have

$$\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u}\rangle \leq \frac{1}{\eta_t}\left(\gamma_t B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) + (1-\gamma_t)B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)\right) - \frac{\gamma_t}{\eta_t}B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1}\rangle \tag{3.12}$$

Now, note that

$$\begin{aligned}
\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1}\rangle - \frac{\gamma_t}{\eta_t}B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) &\leq \|\boldsymbol{g}_t\|_\star \|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\| - \frac{\gamma_t}{\eta_t}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2 \\
&\leq \max_{y\in\mathbb{R}}\left(\|\boldsymbol{g}_t\|_\star z - \frac{\gamma_t}{\eta_t}z^2\right) \\
&\leq \frac{\eta_t}{2\gamma_t}\|\boldsymbol{g}_t\|_\star^2 \ ,
\end{aligned}$$

where the first inequality derives from the strong-convexity of ψ.

Using this last result in Equation (3.12) we get the desired result. □

In the following we show an alternative proof of the regret bound incurred by dual-stabilized OMD depicted in Algorithm 2.

Theorem 3.8. *Set $\boldsymbol{x}_1 \in V$ such that ψ is differentiable in $\boldsymbol{x}_1$. Assume $(\eta)_{t=1}^T$ is a non-increasing sequence. Furthermore, let $\eta_0 = 1$ and set $\gamma_t = \frac{\eta_t}{\eta_{t-1}}$. Then, under the assumptions of Lemma 3.7 and for any $\boldsymbol{u} \in V$, the following regret bounds hold*

$$R_T(\boldsymbol{u}) \leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta_T} + \sum_{t=1}^T \frac{\eta_{t-1}}{2}\|\boldsymbol{g}_t\|_\star^2 \ . \tag{3.13}$$

Proof. From lemma Lemma 3.7, summing over time we get

$$\begin{aligned}
\sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u}\rangle &\leq \frac{1}{\eta_t}\left(\gamma_t B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) + (1-\gamma_t)B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)\right) + \sum_{t=1}^T \frac{\eta_t}{2\gamma_t}\|\boldsymbol{g}_t\|_\star^2 \\
&= \sum_{t=1}^T \left(\frac{1}{\eta_{t-1}}B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - \frac{1}{\eta_t}B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) - \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)\right) + \sum_{t=1}^T \frac{\eta_{t-1}}{2}\|\boldsymbol{g}_t\|_\star^2 \\
&= \frac{1}{\eta_0}B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) - \frac{1}{\eta_T}B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) + B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)\sum_{t=1}^T \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) + \sum_{t=1}^T \frac{\eta_{t-1}}{2}\|\boldsymbol{g}_t\|_\star^2 \\
&\leq \frac{1}{\eta_0}B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) + \left(\frac{1}{\eta_T} - \frac{1}{\eta_0}\right)B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) + \sum_{t=1}^T \frac{\eta_{t-1}}{2}\|\boldsymbol{g}_t\|_\star^2 \\
&= \frac{1}{\eta_T}B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) + \sum_{t=1}^T \frac{\eta_{t-1}}{2} \ .
\end{aligned}$$

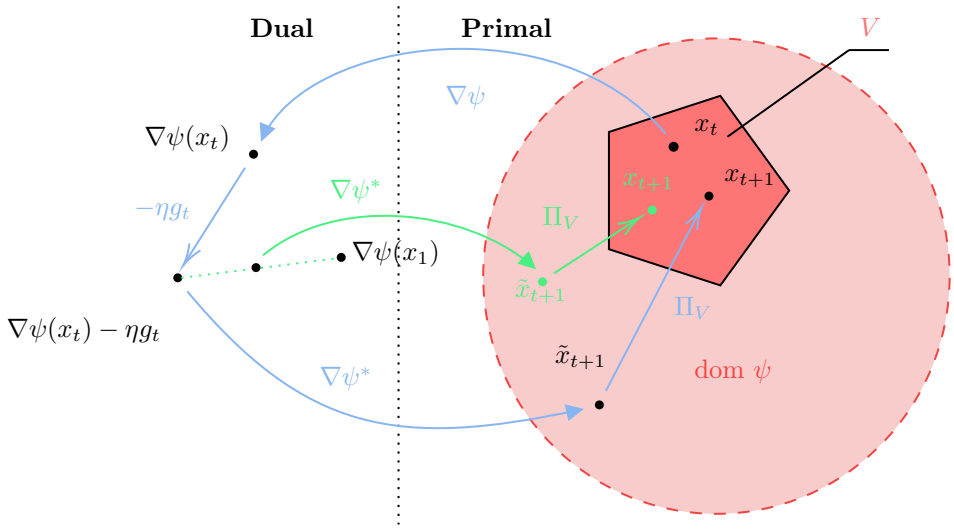


FIGURE 3.1: Comparison between OMD and Dual Stabilized OMD. In blue the steps taken by OMD. In green the corrective steps due to dual stabilization[1].

□

Compared to Theorem 3.3, we can see that this version of OMD does not depend on the maximum Bregman divergence on the domain V at any time step t, but only on the "distance" between $\boldsymbol{x}_1$ and $\boldsymbol{u}$ (which is assumed to be finite) as in the case of OMD with fixed learning rate. On the other hand, contrarily to guarantee we have for OMD, we can see that in Equation (3.13) the sum of the (squared) gradients is multiplied by η_{t-1}. Also, at this point it might not be clear why this version is supposed to have a "dual" stabilization. We will explore this next.

3.2.1 Dual stabilization

In order to talk about the dual stabilization, we should first take a step back and discuss about the "mirror" interpretation of OMD. As explained in many textbooks, the update rule of mirror descent can indeed be rewritten in a different way.

Theorem 3.9 ([Orabona, 2019, Theorem 6.11])**.** *Let B_ψ the Bregman divergence w.r.t. $\psi : X \to \mathbb{R}$, where ψ is $\lambda > 0$ strongly convex. Let $V \subseteq X$ a non-empty closed convex set such that int $X \cap V \neq \emptyset$. Then*

$$\boldsymbol{x}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x} \in V} \eta_t \langle \boldsymbol{g}_t, \boldsymbol{x} \rangle + B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) = \psi_V^\star(\nabla\psi_V(\boldsymbol{x}_t) - \eta_t \boldsymbol{g}_t) \ , \tag{3.14}$$

where ψ_V is the restriction of ψ to V, that is $\psi_V \triangleq \psi + i_V$.

From the theorem above, it can be seen that the functions $\nabla\psi_V$ and $\nabla\psi_V^\star$ play the role of *duality mappings*: the former takes $\boldsymbol{x}_t$ from the primal space to the dual, where a gradient step is performed. Then, the function $\nabla\psi_V^\star$ takes back the resulting variable to the primal space (where possibly a projection on V happens).

As shown in Fang et al. [2020], the update in Algorithm 2 can also be written as follows

$$\begin{aligned}
\hat{\boldsymbol{x}}_t &= \nabla\psi(\boldsymbol{x}_t) \\
\hat{\boldsymbol{w}}_{t+1} &= \hat{\boldsymbol{x}}_t - \eta_{t-1}\boldsymbol{g}_t \\
\hat{\boldsymbol{y}}_{t+1} &= \gamma_t \hat{\boldsymbol{w}}_{t+1} + (1-\gamma_t)\hat{\boldsymbol{x}}_1 \\
\boldsymbol{y}_{t+1} &= \nabla\psi^\star(\hat{\boldsymbol{y}}_{t+1}) \\
\boldsymbol{x}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{x} \in V} B_\psi(\boldsymbol{x}, \boldsymbol{y}_{t+1}),
\end{aligned}$$

[1]The author thanks Victor Portella for the image.

Algorithm 3 Follow-the-Regularized-Leader

Require: Closed and non-empty set $V \subseteq \mathbb{R}^d$, a sequence of regularizers $\psi_1, \dots, \psi_T : \mathbb{R}^d \to (-\infty, +\infty]$

1: **for** $t = 1$ **to** T **do**
2: Output $\boldsymbol{x}_t \in \operatorname{argmin}_{\boldsymbol{x} \in V} \ \psi_t(\boldsymbol{x}) + \sum_{i=1}^{t-1} \ell_i(\boldsymbol{x})$
3: Receive $\ell_t : \mathbb{R}^d \to (-\infty, +\infty]$ and pay $\ell_t(\boldsymbol{x}_t)$
4: **end for**

which compactly is

$$\boldsymbol{x}_{t+1} = \nabla \psi_V^\star(\gamma_t(\nabla \psi(\boldsymbol{x}_t) - \eta_{t-1}\boldsymbol{g}_t) + (1 - \gamma_t)\nabla \psi(\boldsymbol{x}_1)). \tag{3.15}$$

Writing the update in this different form, we can see that after taking the gradient step, the new variable is pushed towards $\boldsymbol{x}_1$ in the dual (see Fig. 3.1 for an illustration). The amount of "pushing" is determined by the parameter γ_t. This step can indeed be seen as inducing stabilization in the algorithm.

However, the dual stabilized version of OMD is not the only algorithm which solve the issue related to unbounded domains with time-varying learning rates. Next, we will see another algorithm tailored for this goal. On the other hand, it is worth noting that there are other versions of OMD used in the literature, which instead adopt increasing learning rates (as done for example in Streeter and McMahan [2012] or in Agarwal et al. [2017] for the bandit setting), but we will not explore them in this thesis.

3.3 FTRL

The idea behind the *Follow The Regularized Leader* (FTRL) algorithm is to play the model which minimizes the sum of the losses observed plus a regularization term. In fact, the regularization term is fundamental, since it can be proven that the simple strategy which plays the point which minimizes the current sum of the losses (also called *Follow The Leader*) is doomed to fail in the worst-case. This algorithm can be traced back to the work of Shalev-Shwartz [2007] in the online setting. On the other hand, in the context of classic convex optimization this algorithm is also known as *Dual Averaging* and first appeared in the work of Nesterov [2009].

Formally, given a convex set $V \subseteq \mathbb{R}^d$ and sequence of loss functions the update for FTRL can be written as

$$\boldsymbol{x}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x} \in V} \sum_{i=1}^{t} \ell_i(\boldsymbol{x}) + \psi_t(\boldsymbol{x}) \ . \tag{3.16}$$

The algorithm is depicted in Algorithm 3. From the update rule in *Equation* (3.16), one possible problem is immediately evident: solving the minimization problem at each step might be computationally infeasible. Therefore, by using again a linearization trick, we can instead minimize the sum of the gradients observed plus a regularization term, i.e.

$$\boldsymbol{x}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x} \in V} \sum_{i=1}^{t} \langle \boldsymbol{g}_i, \boldsymbol{x} \rangle + \psi_t(\boldsymbol{x}) \ . \tag{3.17}$$

Next, we are going to analyze the regret guarantees of this algorithm.

3.3.1 Regret Analysis

First, we can prove the following lemma.

Lemma 3.10. *Let $V \subseteq \mathbb{R}^d$ be closed and non-empty. Given a sequence of convex loss functions $\ell_1, \dots, \ell_T$, set $F_1(\boldsymbol{x}) = \psi_1(\boldsymbol{x})$ and $F_t(\boldsymbol{x}) \triangleq \psi_t(\boldsymbol{x}) + \sum_{i=1}^{t-1} \langle \boldsymbol{g}_i, \boldsymbol{x} \rangle$, where $\boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)$. Assume that $\operatorname{argmin}_{\boldsymbol{x} \in V}$ is not empty and set $\boldsymbol{x}_t = \operatorname{argmin}_{\boldsymbol{x} \in V} F_t(\boldsymbol{x})$. Then, for any $\boldsymbol{u} \in V$,*

the regret is bounded as

$$R_T(\boldsymbol{u}) \leq \sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u} \rangle \leq \psi_{T+1}(\boldsymbol{u}) - \min_{\boldsymbol{x} \in V} \psi_1(\boldsymbol{x}) + \sum_{t=1}^{T} \left[F_t(\boldsymbol{x}_t) - F_{t+1}(\boldsymbol{x}_{t+1}) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle \right] \ . \tag{3.18}$$

Proof. As usual, we have that $\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u}) \leq \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u} \rangle$. Next, note that $\sum_t \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle$ appears on both sides of Equation (3.18). On the other hand, we have that $F_{T+1}(\boldsymbol{u}) = \sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{u} \rangle + \psi_{T+1}(\boldsymbol{u})$. Therefore, rearranging terms we have

$$\begin{aligned}
-\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{u} \rangle &= \psi_{T+1}(\boldsymbol{u}) - F_{T+1}(\boldsymbol{u}) \\
&= \psi_{T+1}(\boldsymbol{u}) - F_{T+1}(\boldsymbol{u}) + \sum_{t=1}^{T} [F_t(\boldsymbol{x}_t) - F_t(\boldsymbol{x}_t)] \\
&= \psi_{T+1}(\boldsymbol{u}) - F_{T+1}(\boldsymbol{u}) - F_1(\boldsymbol{x}_1) + \sum_{t=1}^{T} [F_t(\boldsymbol{x}_t) - F_{t+1}(\boldsymbol{x}_{t+1})] + F_{T+1}(\boldsymbol{x}_{T+1}) \\
&\leq \psi_{T+1}(\boldsymbol{u}) - \min_{\boldsymbol{x} \in V} \psi_1(\boldsymbol{x}) + \sum_{t=1}^{T} [F_t(\boldsymbol{x}_t) - F_{t+1}(\boldsymbol{x}_{t+1})] \ ,
\end{aligned}$$

where the last step follows from the definition of F_t. □

It is worth noticing that Algorithm 3 is invariant to any "translation" of the regularizer ψ_t by a positive constant. Therefore, the regret guarantee above can also be stated by using $\psi_t(\boldsymbol{u})) - \min_{\boldsymbol{x} \in V} \psi_t(\boldsymbol{x})$ rather than $\psi_t(\boldsymbol{u})$.

In order to prove a regret bound which is $\mathcal{O}(\sqrt{T})$, the next step is to assume that ψ is a strongly convex function. In this way, one can easily characterize the regret bound of the algorithm, as shown in the next theorem (proof omitted).

Theorem 3.11 ([Orabona, 2019, Corollary 7.8])**.** *Let $V \subseteq \mathbb{R}^d$ be closed and non-empty. Let ℓ_t be a sequence of convex loss functions. Let $\psi : V \to \mathbb{R}$ a 1-strongly convex function w.r.t $\|\cdot\|$. Set $\psi_t(\boldsymbol{x}) = \frac{1}{\eta_{t-1}}(\psi(\boldsymbol{x}) - \min_{\boldsymbol{z} \in V} \psi(\boldsymbol{z}))$, where $(\eta_{t-1})_{t=1}^{T}$ is a non-increasing sequence. Then, for all $\boldsymbol{u} \in V$ Algorithm 3 incurs regret bounded as*

$$R_T(\boldsymbol{u}) \leq \frac{\psi(\boldsymbol{u}) - \psi(\boldsymbol{x}_1)}{\eta_{T-1}} + \frac{1}{2} \sum_{t=1}^{T} \eta_{t-1} \|\boldsymbol{g}_t\|_\star^2 \ , \tag{3.19}$$

for all $\boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)$.

From the above theorem, we can see that a proper tuning of the learning rate η_t guarantees a sublinear regret bound, as stated in the next corollary.

Corollary 3.12. *Let $V \subseteq \mathbb{R}^d$ be closed and non-empty. Let $\psi : V \to \mathbb{R}$ a 1-strongly convex function w.r.t $\|\cdot\|$, such that $\max_{\boldsymbol{x}, \boldsymbol{y} \in V} \psi(\boldsymbol{x}) - \psi_y \leq D^2$. Let $\ell_1, \dots, \ell_T$ be a sequence of convex and L-Lipschitz loss functions. Set $\eta_t = \frac{D}{L\sqrt{t}}$. Then, for all $\boldsymbol{u} \in V$ Algorithm 3 incurs regret bounded as*

$$R_T(\boldsymbol{u}) \leq 2LD\sqrt{T} \ .$$

Interestingly, we can rewrite the update rule of FTRL in a different way, as done with OMD. Let $\psi_{V,t}(\boldsymbol{x}) = \psi_t(\boldsymbol{x}) + i_V(\boldsymbol{x})$. If we assume $\psi_{V,t}(\boldsymbol{x})$ to be proper, convex and closed, then by applying Rockafellar [1970, Theorem 23.5] we have that $\boldsymbol{x}_{t+1} \in \partial \psi^\star_{V,t+1} \left(-\sum_{i=1}^{t} \boldsymbol{g}_i \right)$. Furthermore, if $\psi_{V,t+1}$ is strongly-convex, then

$$\boldsymbol{x}_{t+1} = \nabla \psi^\star_{V,t+1} \left(-\sum_{i=1}^{t} \boldsymbol{g}_i \right) \ . \tag{3.20}$$

Next, we are going to see a specific application of FTRL, similarly to what done in the case of OMD.

Example 3.3 (Exponentiated Gradient)**.** *As done in Example 3.2, we can look at the resulting algorithm when using* $\psi(\boldsymbol{x}) = \sum_{i=1}^{d} x_i \ln x_i$ *on the d-dimensional simplex, and setting* $\psi_t(\boldsymbol{x}) = \frac{1}{\eta_{t-1}}\psi(\boldsymbol{x})$*. Let* $\boldsymbol{\theta}_t = -\sum_{i=1}^{t} \boldsymbol{g}_i$*. Using the definition of Fenchel conjugate we get*

$$\psi_V(\boldsymbol{\theta}_t)^\star = \max_{\boldsymbol{x}\in V} \langle \boldsymbol{x}, \boldsymbol{\theta}_t \rangle - \psi_V(\boldsymbol{x}) = \frac{1}{\eta_{t-1}} \ln \left(\sum_{i=1}^{d} \exp(\eta_{t-1}\theta_{t,i}) \right) .$$

Also, $(\nabla \psi_V^\star)_j(\boldsymbol{\theta}_t) = \frac{\exp(\eta_{t-1}\theta_j)}{\sum_{i=1}^{d} \exp(\eta_{t-1}\theta_i)}$*. Therefore, we have*

$$x_{t+1,k} = \frac{\exp(-\eta_{t-1} \sum_{i=1}^{t} \boldsymbol{g}_{i,k})}{\sum_{j=1}^{d} \exp(-\eta_{t-1} \sum_{i=1}^{t} \boldsymbol{g}_{i,j})} , \quad \forall k = 1, \dots, d ,$$

for all $t > 1$*. For* $t = 1$*, we can just set* $\boldsymbol{x}_1 = \operatorname{argmin}_{\boldsymbol{x}\in V} \psi(\boldsymbol{x})$*, which in this case will result in* $\boldsymbol{x}_1$ *being the uniform distribution.*

Note that this update can be generalized for any $\boldsymbol{x}_1$*. In this case, we slightly modify the regularizer, using* $\psi(\boldsymbol{x}) = \mathrm{KL}(\boldsymbol{x}, \boldsymbol{x}_1) = \sum_{i=1}^{d} x_i \ln \frac{x_i}{x_{1,i}}$*. It can be shown that the update in this case is*

$$x_{t+1,k} = \frac{x_{1,i} \exp(-\eta_{t-1} \sum_{i=1}^{t} \boldsymbol{g}_{i,k})}{\sum_{j=1}^{d} x_{1,j} \exp(-\eta_{t-1} \sum_{i=1}^{t} \boldsymbol{g}_{i,j})} , \quad \forall k = 1, \dots, d .$$

Note that, compared to Example 3.2 the updates are in generally different, due to the presence of time-varying learning rates. On the other hand, if the learning rate is kept constant, then the two updates coincide.

3.4 Lower bound for OLO

We point out that many extensions to the algorithms described in this chapter exist. For example, one could consider not just the subgradients in the update rule, but any other quantity which keeps the update valid, but could make the loss smaller. This idea gave rise to optimistic-updates [Chiang et al., 2012; Rakhlin and Sridharan, 2013] in both OMD and FTRL settings.

We conclude this chapter giving a lower bound for constrained Online Linear Optimization.

Theorem 3.13 ([Orabona, 2019, Theorem 5.1])**.** *Let* $V \subset \mathbb{R}^d$ *be any non-empty bounded closed convex subset. Let* $D = \sup_{\boldsymbol{v},\boldsymbol{w}\in V} \|\boldsymbol{v} - \boldsymbol{w}\|_2$ *be the diameter of* V*. Let* $\mathcal{A}$ *be any (possibly randomized) algorithm for OLO on* V*. Let* T *be any non-negative integer. There exists a sequence of vectors* $\boldsymbol{g}_1, \dots, \boldsymbol{g}_T$ *with* $\|\boldsymbol{g}_t\|_2 \leq L$ *and* $\boldsymbol{u} \in V$ *such that the regret of algorithm* $\mathcal{A}$ *satisfies*

$$\mathrm{Regret}_T(\boldsymbol{u}) = \sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle - \sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{u} \rangle \geq \frac{\sqrt{2} L D \sqrt{T}}{4} .$$

The lower bound settle the worst-case regret to $\Omega(D\sqrt{T})$. However, this does not close the option of having data-dependent regret bounds, which can be less pessimistic that $\mathcal{O}(D\sqrt{T})$ if the sequence of loss functions is not truly adversarial, but recover the worst-case guarantee if on the other hand this happens. Furthermore, more assumptions can be imposed on the sequence of loss functions, such as strong-convexity. This gives to improved analysis and tighter regret bounds.

Chapter 4

Implicit Updates

As already mentioned in the introduction, when updating its model the learning agent can choose to use directly the loss function, rather than any approximation of it, for example through the use of (sub)gradients. This technique gives rise to updates known as *Implicit* in the online learning literature. Intuitively, the fact that we do not approximate the loss function is a good reason to think that the update will provide a better model. On the other hand, the use of subgradients provides computationally efficient algorithms. Therefore, one might think the price to pay for a better model is in terms of computational overhead. While this is true in general, we will see that there are important cases where it is still possible to compute Implicit updates in closed form.

Implicit updates were proposed for the first time in online learning by Kivinen and Warmuth [1997]. However, in the optimization literature such an update with the Euclidean divergence is known as the Proximal update and it dates back to at least to 1965 [Moreau, 1965; Martinet, 1970; Rockafellar, 1976; Parikh and Boyd, 2014]. More recently, it has been used also in the stochastic setting [Toulis and Airoldi, 2017; Asi and Duchi, 2019]. Later, this idea was re-invented by Crammer et al. [2006] for the specific case of linear prediction with losses that have a range of values in which they are zero, e.g., hinge loss and epsilon-insensitive loss. They called the corresponding family of algorithms Passive-Aggressive, to underline the behaviour of the algorithm of passively keeping the same prediction when the loss is zero and aggressively optimizing it otherwise. Implicit updates were also used for online learning with kernels [Cheng et al., 2007] and to deal with importance weights [Karampatziakis and Langford, 2011].

4.1 Outline

We now describe the structure of this chapter, shortly highlighting existing results and new contributions.

Beyond OMD. In this chapter, we start by providing a refined analysis of Implicit algorithms used in the framework of OMD in Section 4.2. Doing this allows us to understand why Implicit algorithms might practically work better compared to algorithms which use (sub)gradients in the update. In the literature, Kulis and Bartlett [2010] provide the first regret bounds for implicit updates that match those of OMD, while McMahan [2010] makes the first attempt to quantify the advantage of the implicit updates in the regret bound. Later, Song et al. [2018] generalize the results in McMahan [2010] to Bregman divergences and strongly convex functions, and quantify the gain differently in the regret bound. Note that in [McMahan, 2010; Song et al., 2018] the gain cannot be exactly quantified, providing just a non-negative data-dependent quantity subtracted to the regret bound.

On the other hand, we describe how these algorithms can potentially incur only a constant regret if the sequence of loss functions does not vary with time. To this aim, we measure the hardness of the sequence of loss functions with its *temporal variability*, which is defined as

$$V_T \triangleq \sum_{t=2}^{T} \max_{\boldsymbol{x} \in V} \ \ell_t(\boldsymbol{x}) - \ell_{t-1}(\boldsymbol{x}) \ . \tag{4.1}$$

Algorithm 4 Implicit Online Mirror Descent (IOMD)

Require: Non-empty closed convex set $V \subset X \subset \mathbb{R}^d$, $\psi : X \to \mathbb{R}$, $\eta_t > 0$, $\boldsymbol{x}_1 \in V$
1: **for** $t = 1, \dots, T$ **do**
2: Output $\boldsymbol{x}_t \in V$
3: Receive $\ell_t : \mathbb{R}^d \to \mathbb{R}$ and pay $\ell_t(\boldsymbol{x}_t)$
4: Update $\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x} \in V} \; B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) + \eta_t \ell_t(\boldsymbol{x})$
5: **end for**

This will allow us to to quantify more precisely the gain of implicit updates for slow-varying losses.

AdaImplicit. Building on our refined analysis, we next propose a new adaptive Implicit algorithm, *AdaImplicit* (see Section 4.5), which retains the worst-case $\mathcal{O}(\sqrt{T})$ regret bound but takes advantage of a slow varying sequence of loss functions and achieve a regret of $\mathcal{O}(V_T + 1)$. Differently from previous approaches, we show that an adaptive learning rate allows to achieve this improved rate in an automatic way. Also, we prove a lower bound which shows that our algorithm is optimal. In the last few years there has been much interest in designing algorithms which retain the of $\mathcal{O}(\sqrt{T})$ in worst-case scenario, but potentially behave better if the environment is somehow "nice". Our algorithm adds one more item to this list. Finally, in order to show the benefits of using Implicit algorithms in practice, in Section 4.6 we conduct an empirical analysis on real-world datasets in both classification and regression tasks.

4.2 Beyond Linearized Updates

In this section, we introduce the Implicit Online Mirror Descent (IOMD) algorithm, its relationship with OMD, and some of its properties. The results in this section are based on Campolongo and Orabona [2020].

A natural variation of the classic OMD update is to use the actual loss function ℓ_t, rather than its first-order approximation as done in Equation (3.1). This is called *implicit update* [Kivinen and Warmuth, 1997] and is defined as

$$\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x} \in V} \; B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) + \eta_t \ell_t(\boldsymbol{x}) \; . \tag{4.2}$$

Note that, in general, this update does not have a closed form, but for many interesting cases it is still possible to efficiently compute it. Notably, for $\psi = \frac{1}{2}\|\cdot\|_2^2$ and linear prediction with the square, absolute, and hinge loss, these updates can all be computed in closed form when $V = \mathbb{R}^d$ [see, e.g., Crammer et al., 2006; Kulis and Bartlett, 2010]. This update leads to the Implicit Online Mirror Descent (IOMD) algorithm in Algorithm 4.

We next show how the update in Equation (4.2) yields new interesting properties which are not shared with its non-implicit counterpart.

Proposition 4.1. *Let $\boldsymbol{x}_{t+1}$ be defined as in Equation (4.2). Then, there exists $\boldsymbol{g}'_t \in \partial \ell_t(\boldsymbol{x}_{t+1})$ such that*

$$\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)/\eta_t \geq 0, \tag{4.3}$$
$$\langle \eta_t \boldsymbol{g}'_t + \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_t), \boldsymbol{u} - \boldsymbol{x}_{t+1} \rangle \geq 0 \quad \forall \boldsymbol{u} \in V, \tag{4.4}$$
$$\langle \boldsymbol{g}'_t, \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle \geq \langle \boldsymbol{g}_t, \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle \; . \tag{4.5}$$

The first property in Equation (4.3) implies that, in contrast to OMD, the value of the loss function in $\boldsymbol{x}_{t+1}$ is always smaller than or equal to its value in $\boldsymbol{x}_t$. This means that, if $\ell_t = \ell$, the value $\ell(\boldsymbol{x}_t)$ will be monotonically decreasing over time. The second property in Equation (4.4) gives an alternative way to write the update rule expressed in Equation (4.2). In particular, using $\psi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$ and $V = \mathbb{R}^d$ the update becomes $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \boldsymbol{g}'_t$, motivating the name "implicit". Using this fact in the last property (Equation (4.5)),[1] we

[1] Equation (4.5) is nothing else than the fact that subgradients are monotone operators.

have that with L_2 regularization, the dual norm of $\boldsymbol{g}'_t$ is smaller than the dual norm of $\boldsymbol{g}_t$, i.e. $\|\boldsymbol{g}'_t\|_2 \leq \|\boldsymbol{g}_t\|_2$.

Proof of Proposition 4.1. From the update in Equation (4.2), we immediately get the following inequality

$$\eta_t \ell_t(\boldsymbol{x}_{t+1}) \leq \eta_t \ell_t(\boldsymbol{x}_{t+1}) + B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) \leq \eta_t \ell_t(\boldsymbol{x}_t) + B_\psi(\boldsymbol{x}_t, \boldsymbol{x}_t) = \eta_t \ell_t(\boldsymbol{x}_t),$$

which verifies Equation (4.3) and implies that value of the loss ℓ_t in $\boldsymbol{x}_{t+1}$ is not bigger than the one in $\boldsymbol{x}_t$.

Equation (4.4) is simply the first-order optimality condition for $\boldsymbol{x}_{t+1}$.

For Equation (4.5), from the convexity of ℓ_t we have

$$\begin{aligned} \ell_t(\boldsymbol{x}_{t+1}) &\geq \ell_t(\boldsymbol{x}_t) + \langle \boldsymbol{g}_t, \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle, \\ \ell_t(\boldsymbol{x}_t) &\geq \ell_t(\boldsymbol{x}_{t+1}) + \langle \boldsymbol{g}'_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1} \rangle \ . \end{aligned}$$

Summing both inequalities, we get the desired result. □

Let's gain some additional intuition on the implicit updates. Consider the case of $V = \mathbb{R}^d$ and $\psi(\boldsymbol{x}) = \frac{1}{2}\|\cdot\|_2^2$. We have that $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \boldsymbol{g}'_t$, where $\boldsymbol{g}'_t \in \partial \ell_t(\boldsymbol{x}_{t+1})$. Now, if $\ell_{t+1} \approx \ell_t$, we would be updating the algorithm approximately with the *next subgradient*. On the other hand, knowing future gradients is a safe way to have constant regret. Hence, we can expect IOMD to have low regret if the functions are slowly varying over time. In the next sections, we will see that this is indeed the case.

4.3 Two Regret Bounds for IOMD

In the following, we will present a new regret guarantee for IOMD. Using the properties given in the previous section we are now going to prove a useful lemma which we will later use to show a regret bound for Algorithm 4. Intuitively, the following lemma says that if we were able to play $\boldsymbol{x}_{t+1}$ on round t, then the regret would be constant or even negative. Indeed, the bound consists of two pieces: the first is the usual one related to the diameter of the set V w.r.t. the Bregman divergence, while the second one is the (negative) distance between $\boldsymbol{x}_{t+1}$ and the previous model $\boldsymbol{x}_t$.

Lemma 4.2. *Let $V \subset X \subseteq \mathbb{R}^d$ be a non-empty closed convex set. Let B_ψ be the Bregman divergence w.r.t. $\psi : X \to \mathbb{R}$. Then, Algorithm 4 guarantees*

$$\sum_{t=1}^T \ell_t(\boldsymbol{x}_{t+1}) - \sum_{t=1}^T \ell_t(\boldsymbol{u}) \leq \sum_{t=1}^T \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1})}{\eta_t} - \sum_{t=1}^T \frac{1}{\eta_t} B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) \ . \tag{4.6}$$

Furthermore, let $D^2 \triangleq \max_{\boldsymbol{x}, \boldsymbol{u} \in V} B_\psi(\boldsymbol{u}, \boldsymbol{x})$ and assume that $(\eta_t)_{t=1}^T$ is a non-increasing sequence. Then the bound can further be expressed as

$$\sum_{t=1}^T \ell_t(\boldsymbol{x}_{t+1}) - \sum_{t=1}^T \ell_t(\boldsymbol{u}) \leq \frac{D^2}{\eta_T} - \sum_{t=1}^T \frac{1}{\eta_t} B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) \ . \tag{4.7}$$

Proof. Let $\boldsymbol{g}'_t \in \partial \ell_t(\boldsymbol{x}_{t+1})$. From Equation (4.4), we have that

$$\begin{aligned} \eta_t(\ell_t(\boldsymbol{x}_{t+1}) - \ell_t(\boldsymbol{u})) &\leq \langle \eta_t \boldsymbol{g}'_t, \boldsymbol{x}_{t+1} - \boldsymbol{u} \rangle \\ &\leq \langle \nabla\psi(\boldsymbol{x}_t) - \nabla\psi(\boldsymbol{x}_{t+1}), \boldsymbol{x}_{t+1} - \boldsymbol{u} \rangle \\ &= B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) \ . \end{aligned}$$

Dividing by η_t and summing over $t = 1, \dots, T$ yields the result in Equation (4.6).

For the second part observe that similar to what we have done in the proof of Theorem 3.3 we have

$$\sum_{t=1}^T \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1})}{\eta_t} \leq \frac{D^2}{\eta_1} + D^2 \sum_{t=2}^T \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) = \frac{D^2}{\eta_T} \ . \qquad \square$$

Adding $\sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t)$ on both sides of Equation (4.6), we immediately get our new regret bound.

Theorem 4.3. *Under the assumptions of Lemma 4.2, the regret incurred by Algorithm 4 is bounded as*

$$R_T(\boldsymbol{u}) \leq \sum_{t=1}^{T} \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1})}{\eta_t} + \sum_{t=1}^{T} \left[\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta_t} \right] . \quad (4.8)$$

We note that this result could also be extrapolated from [Song et al., 2018], by carefully going through the proof of their Lemma 1. However, as in the other previous work, they did not identify that the key quantity to be used in order to quantify an actual gain is the temporal variability V_T, as we will show later.

First Regret: Recovering OMD's Guarantee. To this point, the advantages of an implicit update are still not clear. Therefore, we now show how, from Theorem 4.3, one can get a possibly tighter bound than the usual $\mathcal{O}(\sqrt{T})$. The key point in this new analysis is to introduce $\boldsymbol{g}'_t$ as defined in Proposition 4.1 and relate it to the Bregman divergence between $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$.

Theorem 4.4. *Let $\boldsymbol{g}'_t \in \partial \ell_t(\boldsymbol{x}_{t+1})$ satisfy Equation (4.4). Assume ψ to be 1-strongly convex w.r.t. $\|\cdot\|$. Then, under the assumptions of Lemma 4.2, we have that Algorithm 4 satisfies*

$$\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta_t} \leq \eta_t \|\boldsymbol{g}_t\|_\star \min\left(2\|\boldsymbol{g}'_t\|_\star, \frac{\|\boldsymbol{g}_t\|_\star}{2} \right), \ \forall t, \boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t) . \quad (4.9)$$

Proof. Using the convexity of the losses, we can bound the difference between $\ell_t(\boldsymbol{x}_t)$ and $\ell_t(\boldsymbol{x}_{t+1})$:

$$\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) \leq \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1} \rangle \leq \|\boldsymbol{g}_t\|_* \|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\|,$$

where $\boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)$. Given that ψ is 1-strongly convex, we can use Equation (2.5) to obtain

$$\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) \leq \|\boldsymbol{g}_t\|_\star \sqrt{2 B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)} . \quad (4.10)$$

Note that $\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, x_t)/\eta_t \leq \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1})$. Hence, to get the first term in the min of Equation (4.9), we can simply look for an upper bound on the term $\sqrt{2B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}$ in Equation (4.10) above. Using the fact that the Bregman divergence is convex in its first argument, we get

$$\begin{aligned} B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) &\leq \langle \nabla \psi(\boldsymbol{x}_{t+1}) - \nabla \psi(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle \\ &\leq \langle \eta_t \boldsymbol{g}'_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1} \rangle \\ &\leq \eta_t \|\boldsymbol{g}'_t\|_\star \, \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| \\ &\leq \eta_t \|\boldsymbol{g}'_t\|_\star \sqrt{2 B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}, \end{aligned}$$

where we used Equation (4.4) in the second inequality and Equation (2.5) in the last one. Solving this inequality with respect to $B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)$, we get $\sqrt{2B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)} \leq 2\eta_t \|\boldsymbol{g}'_t\|_\star$.

For the second term, it suffices to subtract $B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)/\eta_t$ on both sides of Equation (4.10) and use the fact that $bx - \frac{a}{2}x^2 \leq \frac{b^2}{2a}, \forall x \in \mathbb{R}$ with $x = B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)$. □

The above theorem immediately gives us that Algorithm 4 has a regret upper-bounded by

$$R_T(\boldsymbol{u}) \leq \sum_{t=1}^{T} \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1})}{\eta_t} + \sum_{t=1}^{T} \eta_t \|\boldsymbol{g}_t\|_\star \min\left(2\|\boldsymbol{g}'_t\|_\star, \frac{\|\boldsymbol{g}_t\|_\star}{2} \right), \quad (4.11)$$

where $\boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)$. The presence of the minimum makes this bound equivalent in a worst-case sense to the one of OMD in Equation (3.8). Moreover, at least in the Euclidean case, from Equation (4.5) we have that $\|\boldsymbol{g}'_t\|_2 \leq \|\boldsymbol{g}_t\|_2$. However, it is difficult to quantify the gain over OMD because in general $\|\boldsymbol{g}_t\|_\star$ and $\|\boldsymbol{g}'_t\|_\star$ are data-dependent. Hence, as in the other

Algorithm 5 IOMD with doubling trick

Require: Non-empty closed convex set $V \subset X \subset \mathbb{R}^d$, distance generating function $\psi : \mathbb{R}^d \to \mathbb{R}$, Lipschitz constant $L > 0$, $\beta > 0$, $\boldsymbol{x}_1 \in V$

Initialize: $i = 0$, $S_0 = 0$, $\eta_0 = \frac{\beta}{L}$
for $t = 1, \dots, T$ **do**
  Output $\boldsymbol{x}_t$
  Receive $\ell_t : \mathbb{R}^d \to (-\infty, +\infty]$ and pay $\ell_t(\boldsymbol{x}_t)$
  $\boldsymbol{x}'_{t+1} = \arg\min_{\boldsymbol{x} \in V} \; B_\psi(\boldsymbol{x}, \boldsymbol{x}_t) + \eta_i \ell_t(\boldsymbol{x})$
  $\delta_t = \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}'_{t+1}) - \frac{B_\psi(\boldsymbol{x}'_{t+1} - \boldsymbol{x}_t)}{\eta_i}$
  $S_i \leftarrow S_i + \delta_t$
  if $S_i \geq \eta_i L^2 2^i$ **then**
    $i = i + 1$
    Update $\eta_i = \frac{\beta}{L\sqrt{2^i}}$
    $S_i = 0$
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_1$
  else
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}'_{t+1}$
  end if
end for

previous analyses, the gain over OMD would be only marginal and not quantifiable. This is not a limit of our analysis: it is easy to realize that in the worst case the OMD update and the IOMD update can coincide. To show instead that a real gain is possible, we are now going to take a different path.

Second Regret: Temporal Variability in IOMD. Here we formalize our key intuition that IOMD is using an approximation of the future subgradient when the losses do not vary much over time. We use the notion of *temporal variability of the losses*, V_T, as given in Equation (4.1). Considering again our regret bound in Theorem 4.3 and using $\eta_t = \eta$ for all t, we immediately have

$$\begin{aligned} R_T(\boldsymbol{u}) &\leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta} + \sum_{t=1}^{T} \left(\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta} \right) \\ &\leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta} + \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + \sum_{t=2}^{T} \left(\max_{\boldsymbol{x} \in V} \; \ell_t(\boldsymbol{x}) - \ell_{t-1}(\boldsymbol{x}) - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta} \right) \\ &\leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta} + \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T \; . \end{aligned}$$

This means that using a *constant learning rate yields a regret bound of* $\mathcal{O}(V_T+1)$*, which might be better than* $\mathcal{O}(\sqrt{T})$ *if the temporal variability is low.* In particular, we can even get constant regret if $V_T = \mathcal{O}(1)$. On the contrary, OMD cannot achieve a constant regret for any convex loss even if $V_T = 0$, since it would imply an impossible $\mathcal{O}(1/T)$ rate for non-smooth batch black-box optimization [Nesterov, 2004, Theorem 3.2.1]. Instead, IOMD does not violate the lower bound since it is not a black-box method. As far as we know, the connection between IOMD and temporal variability has never been observed before. On the other hand, even when the temporal variability is high, we can still use a $\mathcal{O}(1/\sqrt{T})$ learning rate to achieve a worst case regret of the order $\mathcal{O}(\sqrt{T})$.

Finally, a natural question arises: can we get a bound which interpolates between $\mathcal{O}(V_T+1)$ and $\mathcal{O}(\sqrt{T})$, without any prior knowledge on the quantity V_T? We first give an encouraging result by presenting a strategy based on a doubling trick and then a positive answer to this question through a fully adaptive strategy.

4.4 Doubling Trick

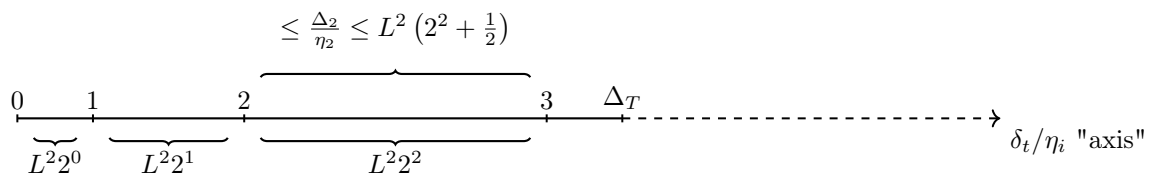


FIGURE 4.1: Doubling trick illustrated.

In this section, we present a doubling trick strategy to tune the learning rate in IOMD. In order to provide an intuition to this trick, we briefly explain it in the case it is applied to the time horizon. Assume that a parameter λ for a certain algorithm has to be tuned as a function of the time horizon T, but the latter quantity is unknown in advance. The idea is then to partition time into periods of exponentially increasing lengths. Then in each period the parameter λ is chosen optimally as a function of the length of the interval and the algorithm is run. When the periods ends, the algorithm is restarted for the next period with a new tuning of the parameter λ. In fact, this trick is more general and can be applied to any observable quantity which is *monotonically* increasing/decreasing.

If we look at our problem, it is possible to apply this construction if we consider the terms δ_t, whose sum over time is an increasing sequence thanks to Equation (4.3). In the following, we slightly modify the definition of δ_t given in Equation (4.19) (see line 6 of Algorithm 5),

$$\delta_t = \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}'_{t+1}) - \frac{B_\psi(\boldsymbol{x}_t, \boldsymbol{x}'_{t+1})}{\eta_t} \ .$$

The algorithm works as follows: at the beginning of epoch i, we set the learning rate $\eta_i = \beta/L\sqrt{2^i}$, run IOMD and monitor the sum $\sum_t \delta_t/\eta_i$ until it reaches $L^2 2^i$. Once this happens, we restart the algorithm doubling the threshold and halving the learning rate—see Algorithm 5. Note that during an epoch the learning rate stays fixed. Let t_i be the index of the first round of epoch i. We then have $\eta_t = \eta_i$ for $t \in [t_i, t_{i+1} - 1]$.

For the sake of the analysis, we assume that by using a doubling trick the time horizon is divided in $N+1$ different epochs (where N is obviously not known a priori). Next, we show that the number N is logarithmic in a quantity Δ_i, which we define in the next lemma.

Lemma 4.5. *Let t_i be the first time-step of epoch i, with $t_0 = 1$. Suppose Algorithm 5 is run for a total of $N+1$ epochs. Let $\Delta_i \triangleq \sum_{t=t_i}^{t_{i+1}-1} \delta_t$. Then, we have that*

$$N \leq \log_2\left(\frac{1}{L^2}\sum_{i=0}^{N}\frac{\Delta_i}{\eta_i} + 1\right) \ . \tag{4.12}$$

Proof. First, we have that

$$\sum_{i=0}^{N-1} a^i = \frac{a^N - 1}{a - 1} \ . \tag{4.13}$$

Now, note that the sum of the $L^2 2^i$ terms in the first $N-1$ epochs is less than or equal to the final sum of the terms monitored. Therefore, using Equation (4.13) we have

$$\sum_{i=0}^{N-1} L^2 2^i = L^2(2^N - 1) \leq \sum_{i=0}^{N}\sum_{t=t_i}^{t_{i+1}-1}\frac{\delta_t}{\eta_i} = \sum_{i=0}^{N}\frac{\Delta_i}{\eta_i},$$

and solving for N yields the desired result.

□

We can now analyze the regret incurred by the Algorithm 5. Each epoch is bounded individually. The final regret bound is given by the sum of the individual contributions over all the epochs.

Theorem 4.6. *Assume the losses $\ell_t(\boldsymbol{x})$ to be L-Lipschitz, for all $t = 1, \dots, T$ and ψ to be 1-strongly convex w.r.t. $\|\cdot\|$. Let Δ_i be defined as in Lemma 4.5. Then, for any $\boldsymbol{u} \in \mathbb{R}^d$ the*

regret of Algorithm 5 after T rounds is bounded as

$$R_T(\boldsymbol{u}) \leq c\left(\frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\beta} + \beta\right) L\sqrt{T+1} + c\frac{\beta L}{2}, \tag{4.14}$$

where $c = \frac{\sqrt{2}}{\sqrt{2}-1}$.

Proof. Using the notation introduced, we have that $\delta_{t_{i+1}-1}$ is the term which causes the restart in epoch i. Therefore, the following holds

$$\frac{\Delta_i}{\eta_i} \leq L^2\left(2^i + \frac{1}{2}\right), \tag{4.15}$$

since from Lemma 4.4 the last term in epoch i is such that $\frac{\delta_{t_{i+1}-1}}{\eta_i} \leq \frac{L^2}{2}$.

For any $\boldsymbol{u} \in \mathbb{R}^d$, we next define $R_i(\boldsymbol{u})$ as the regret during epoch i,

$$R_i(\boldsymbol{u}) = \sum_{t=t_i}^{t_{i+1}-1} \left(\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u})\right) .$$

Using Equation (4.8) with $\eta_t = \eta_i$ for $t \in [t_i, \ldots, t_{i+1}-1]$, we have that $R_i(\boldsymbol{u})$ is bounded as follows

$$\begin{aligned} R_i(\boldsymbol{u}) &\leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t_i})}{\eta_i} + \sum_{t=t_i}^{t_{i+1}-1} \left[\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}'_{t+1}) - \frac{1}{\eta_i} B_\psi(\boldsymbol{x}'_{t+1}, \boldsymbol{x}_t)\right] \\ &\leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t_i})}{\eta_i} + \sum_{t=t_i}^{t_{i+1}-1} \delta_t \\ &= \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta_i} + \Delta_i \ . \end{aligned} \tag{4.16}$$

We can now write the final regret bound by summing $R_i(\boldsymbol{u})$ over all the epochs $i = 1, \ldots, N$.

$$\begin{aligned} R_T(\boldsymbol{u}) &= \sum_{i=0}^{N} R_i(\boldsymbol{u}) \\ &\leq \sum_{i=0}^{N} \left[\frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta_i} + \Delta_i\right] \\ &\leq \sum_{i=0}^{N} \left[\frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta_i} + \eta_i L^2\left(2^i + \frac{1}{2}\right)\right] && \text{see Equation (4.15)} \\ &= L\left(\frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\beta} + \beta\right) \sum_{i=0}^{N} \sqrt{2^i} + \frac{\beta L}{2} \sum_{i=0}^{N} \frac{1}{\sqrt{2^i}} && \left(\eta_i = \frac{\beta}{L\sqrt{2^i}}\right) \\ &\leq \left(\frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\beta} + \beta\right) L\frac{2^{\frac{N+1}{2}} - 1}{\sqrt{2}-1} + \frac{\beta L}{2}\frac{\sqrt{2}}{\sqrt{2}-1} \\ &\leq c\left(\frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\beta} + \beta\right) L 2^{\frac{N}{2}} + c\frac{\beta L}{2}, \end{aligned} \tag{4.17}$$

where in the last step we used Equation (4.13) and the definition of c.

Now, using Lemma 4.5 we have that

$$2^{\frac{N}{2}} = 2^{\frac{1}{2}\log_2\left(\frac{1}{L^2}\sum_{i=0}^{N}\frac{\Delta_i}{\eta_i}+1\right)} = \sqrt{\frac{1}{L^2}\sum_{i=0}^{N}\frac{\Delta_i}{\eta_i} + 1} \ .$$

Therefore, from Equation (4.17) the regret can be bounded as

$$R_T(\boldsymbol{u}) \leq c\left(\frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\beta} + \beta\right)\sqrt{\sum_{i=0}^{N}\frac{\Delta_i}{\eta_i} + L^2} + c\frac{\beta L}{2} \ .$$

Furthermore, using Theorem 4.4 and the assumption on the losses to be L-Lipschitz w.r.t. $\|\cdot\|$, we get

$$\sum_{i=0}^{N}\frac{\Delta_i}{\eta_i} = \sum_{i=0}^{N}\sum_{t=t_i}^{t_{i+1}-1}\frac{\delta_t}{\eta_i} \leq \sum_{i=0}^{N}\sum_{t=t_i}^{t_{i+1}-1}\frac{L^2}{2} = \frac{L^2}{2}T \ .$$

Substituting back the above result we get

$$R_T(\boldsymbol{u}) \leq cL\left(\frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\beta} + \beta\right)\sqrt{T+1} + c\frac{\beta L}{2} \ . \qquad \square$$

From this last result, it is still not evident how an implicit algorithm can benefit from slowly varying losses. However, we next show how in the case of fixed losses, we can still recover a constant regret bound, even if the domain is unbounded.

4.4.1 Fixed Losses

If the losses are all equal, i.e. $\ell_t(\boldsymbol{x}) = \ell(\boldsymbol{x})$ for all $t = 1, \dots, T$, Theorem 4.6 would imply that Algorithm 5 incurs a regret which scales as $\mathcal{O}(\sqrt{T})$. However, as we are going to show in the next lemma, with a proper setting of β this is actually not the case and Algorithm 5 always stays in the first epoch, achieving a constant regret even if the domain is unbounded!

Lemma 4.7. *Assume the losses over time are fixed, i.e. $\ell_t(\boldsymbol{x}) = \ell(\boldsymbol{x})$ for all $t = 1, \dots, T$ and $\psi(\boldsymbol{x})$ be 1-strongly convex w.r.t $\|\cdot\|$. Then, Algorithm 5 with $\beta \geq 1$ will always stay in the first epoch, i.e., $N = 0$.*

Proof. In order to not have a restart, we need $\delta_t \leq \eta_i L^2 2^i$, which translates to

$$\ell(\boldsymbol{x}_t) - \ell(\boldsymbol{x}'_{t+1}) \leq \beta L\sqrt{2^i}\left(B_\psi(\boldsymbol{x}'_{t+1}, \boldsymbol{x}_t) + 1\right) \ .$$

From the inequality above, we have a reset iff

$$\begin{aligned} i &\leq 2\log_2 \frac{\ell(\boldsymbol{x}_t) - \ell(\boldsymbol{x}'_{t+1})}{\beta L\left(B_\psi(\boldsymbol{x}'_{t+1}, \boldsymbol{x}_t) + 1\right)} \leq 2\log_2 \frac{L\|\boldsymbol{x}_t - \boldsymbol{x}'_{t+1}\|}{\beta L\left(1 + \frac{1}{2}\|\boldsymbol{x}'_{t+1} - \boldsymbol{x}_t\|^2\right)} \\ &= 2\log_2 \frac{2\|\boldsymbol{x}'_{t+1} - \boldsymbol{x}_t\|}{\beta(2 + \|\boldsymbol{x}'_{t+1} - \boldsymbol{x}_t\|^2)}, \end{aligned}$$

where the second inequality derives from the Lipschitzness of the losses and by lower bounding the Bregman divergence with Equation (2.5). Now, let $f(y) = \frac{2y}{2+y^2}$, with $y \geq 0$. We have that $\lim_{y\to+\infty} f(y) = 0$ and $f(0) = 0$. Furthermore, if we take the derivative and set it to 0, we see that $f(y)$ has a maximum in $y = \sqrt{2}$. Hence, we have

$$i \leq 2\log_2 \frac{2\|\boldsymbol{x}'_{t+1} - \boldsymbol{x}_t\|}{\beta(2 + \|\boldsymbol{x}'_{t+1} - \boldsymbol{x}_t\|^2)} \leq 2\log_2 \frac{1}{\beta\sqrt{2}} = -1 - 2\log_2\beta \leq -1,$$

where the last step derives from the assumption on β. Therefore, if $i > -1$ then we will not double the learning rate. Note that this is always verified and we can conclude that $N = 0$. $\square$

We can now prove a regret bound in the case of fixed losses.

Theorem 4.8. *Under the assumptions of Lemma 4.7 the regret incurred by Algorithm 5 is bounded as*

$$R_T(\boldsymbol{u}) \leq \frac{L}{\beta}B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) + \ell(\boldsymbol{x}_1) - \ell(\boldsymbol{x}_T) \ . \tag{4.18}$$

Algorithm 6 AdaImplicit

Require: Non-empty closed convex set $V \subset X \subset \mathbb{R}^d$, $\psi : X \to \mathbb{R}$, $\lambda_1 = 0$, $\beta^2 > 0$, $\boldsymbol{x}_1 \in V$
1: **for** $t = 1, \dots, T$ **do**
2: Output $\boldsymbol{x}_t \in V$
3: Receive $\ell_t : \mathbb{R}^d \to \mathbb{R}$ and pay $\ell_t(\boldsymbol{x}_t)$
4: Update $\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x} \in V} \ \ell_t(\boldsymbol{x}) + \lambda_t B_\psi(\boldsymbol{x}, \boldsymbol{x}_t)$
5: Set $\delta_t = \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \lambda_t B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)$
6: Update $\lambda_{t+1} = \lambda_t + \frac{1}{\beta^2}\delta_t$
7: **end for**

Proof. From Equation (4.16) and the fact that we only have one epoch thanks to Lemma 4.7, we have that

$$\begin{aligned} R_T(\boldsymbol{u}) = R_0(\boldsymbol{u}) &\leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta_0} + \sum_{t=1}^T \delta_t \\ &= \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta_0} + \sum_{t=1}^T \left[\ell(\boldsymbol{x}_t) - \ell(\boldsymbol{x}_{t+1}) - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta_0} \right] \\ &\leq \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_1)}{\eta_0} + \ell(\boldsymbol{x}_1) - \ell(\boldsymbol{x}_T) \\ &= \frac{L}{\beta} B_\psi(\boldsymbol{u}, \boldsymbol{x}_1) + \ell(\boldsymbol{x}_1) - \ell(\boldsymbol{x}_T) \ . \end{aligned}$$

□

In principle, it should be possible to get a bound which interpolates between $\mathcal{O}(\sqrt{T})$ and $\mathcal{O}(V_T)$. However, for the latter possibility a bounded domain seems required, in order to bound the difference between the first and last losses of each epoch. Moreover, we doubt that a strategy based on doubling trick gives any empirical advantage compared to an adaptive algorithm, due to the periodic restarts the algorithm incurs. For these reasons, we will not provide details about how to use Algorithm 5 in order to get a regret bound order of $\mathcal{O}(V_T)$. The only advantage coming from using a doubling trick derives from the fact that it is not required to have a bounded domain (as shown in Theorem 4.6), opposed to the case of an adaptive learning rate.

Next, we show how a fully adaptive strategy can be employed instead.

4.5 Adapting to the Temporal variability with AdaImplicit

In this section, we present an adaptive strategy to set the learning rates, in order to give a regret guarantee that depends optimally on the temporal variability.

From the previous section, we saw that the key quantity in the IOMD regret bound is

$$\delta_t \triangleq \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta_t} \ . \tag{4.19}$$

The main question is now: how to set the learning rate η_t in order to have a bound which is $\mathcal{O}(\sqrt{T})$ in the worst case, but constant in the case of fixed losses? We will show it next.

AdaImplicit. As mentioned in the previous section and shown in Theorem 3.3, an adaptive algorithm based on OMD needs a bounded domain. Therefore, define $D^2 \triangleq \max_{\boldsymbol{x}, \boldsymbol{u} \in V} \ B_\psi(\boldsymbol{u}, \boldsymbol{x})$ and assume $D < \infty$. For ease of notation, we let $\eta_t = 1/\lambda_t$ where λ_t will be decided in the following. Assuming $(\lambda_t)_{t=1}^T$ to be a non-decreasing sequence, from Theorem 4.3 we get

$$R_T(\boldsymbol{u}) \leq D^2 \lambda_T + \sum_{t=1}^T \left[\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \lambda_t B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) \right] \ . \tag{4.20}$$

Ideally, to minimize the regret we would like to have λ_T to be as close as possible to the sum over time in the r.h.s. of this expression. However, setting $\lambda_t \propto \sum_{s=1}^{t} \delta_i$ would introduce an annoying recurrence in the computation of λ_t. To solve this issue, we explore the same strategy adopted in AdaFTRL [Orabona and Pál, 2015], adapting it to the OMD case: we set $\lambda_{t+1} = \frac{1}{\beta^2} \sum_{i=1}^{t} \delta_i$ for $t \geq 2$, for a parameter β to be defined later, and $\lambda_1 = 0$. We call the resulting algorithm AdaImplicit and describe it in Algorithm 6. Before proving a regret bound for it, we first provide a technical lemma for the analysis. This lemma can be found in Orabona and Pál [2018] and for completeness we give a proof.

Lemma 4.9. *Let $\{a_t\}_{t=1}^{\infty}$ be any sequence of non-negative real numbers. Suppose that $\{\Delta_t\}_{t=1}^{\infty}$ is a sequence of non-negative real numbers satisfying $\Delta_1 = 0$ and[2] $\Delta_{t+1} \leq \Delta_t + \min\{ba_t, ca_t^2/(2\Delta_t)\}$, for any $t \geq 1$. Then, for any $T \geq 0, \Delta_{T+1} \leq \sqrt{(b^2 + c)\sum_{t=1}^{T} a_t^2}$.*

Proof. From the assumptions, we have that $\Delta_{t+1} \leq \Delta_t + \min\left\{ba_t,\ ca_t^2/(2\Delta_t)\right\}$, for any $t \geq 1$. Also, observe that

$$\Delta_{T+1}^2 = \sum_{t=1}^{T} \Delta_{t+1}^2 - \Delta_t^2 = \sum_{t=1}^{T} \left[\underbrace{(\Delta_{t+1} - \Delta_t)^2}_{\text{(a)}} + \underbrace{2(\Delta_{t+1} - \Delta_t)\Delta_t}_{\text{(b)}} \right] .$$

We bound the sequences (a) and (b) separately. For (a), from the assumption on the recurrence and using the first term in the minimum we have that $(\Delta_{t+1} - \Delta_t)^2 \leq b^2 a_t^2$. On the other hand, for (b) using the second term in the minimum in the recurrence we get $2(\Delta_{t+1} - \Delta_t)\Delta_t \leq ca_t^2$. Putting together the results we have that $\Delta_{T+1}^2 \leq (b^2 + c)\sum_{t=1}^{T} a_t^2$ and the lemma follows. □

We are now ready to prove a regret bound for Algorithm 6.

Theorem 4.10. *Let $V \subset X \subseteq \mathbb{R}^d$ be a non-empty closed convex set. Let B_ψ be the Bregman divergence w.r.t. $\psi : X \to \mathbb{R}$ and let $D^2 = \max_{\boldsymbol{x},\boldsymbol{u} \in V} B_\psi(\boldsymbol{u}, \boldsymbol{x})$. Assume ψ to be 1-strongly convex with respect to $\|\cdot\|$ in V. Then, for any $\boldsymbol{u} \in V$, running Algorithm 6 with $\beta = D$ guarantees*

$$R_T(\boldsymbol{u}) \leq \min \left\{ 2(\ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T)\ ,\ 2D\sqrt{3\sum_{t=1}^{T} \|\boldsymbol{g}_t\|_\star^2} \right\}, \quad \forall \boldsymbol{g}_t \in \ell_t(\boldsymbol{x}_t)\ . \tag{4.21}$$

Proof. Using the definition of λ_t and the fact that the sequence $(\lambda_t)_{t=1}^{T+1}$ is increasing over time, the regret in Equation (4.20) can be upper bounded as $R_T(\boldsymbol{u}) \leq (D^2 + \beta^2)\lambda_{T+1}$. Therefore, we need an upper bound on λ_{T+1}. We split the proof in two parts, one for each term in the min in Equation (4.21). For the first term, using the definition of λ_t we have

$$\begin{aligned}
\beta^2 \lambda_{T+1} &= \sum_{t=1}^{T} [\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \lambda_t B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)] \\
&\leq \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + \sum_{t=2}^{T} [\ell_t(\boldsymbol{x}_t) - \ell_{t-1}(\boldsymbol{x}_t)] \leq \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T\ ,
\end{aligned}$$

from which using $\beta = D$ the result follows.

For the second term, from Lemma 4.4 for $t \geq 2$ we have $\delta_t \leq \frac{\|\boldsymbol{g}_t\|_\star^2}{2\lambda_t}$. On the other hand,

$$\begin{aligned}
\delta_t &= \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \lambda_t B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) \leq \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) \leq \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1} \rangle \\
&\leq \|\boldsymbol{g}_t\|_\star \|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\| \leq \sqrt{2} D \|\boldsymbol{g}_t\|_\star,
\end{aligned}$$

where in the last step we used Equation (2.5) and the definition of D. Therefore, putting the last two results together we get

$$\delta_t \leq \min\left(\sqrt{2} D \|\boldsymbol{g}_t\|_\star, \|\boldsymbol{g}_t\|_\star^2/(2\lambda_t)\right), \quad \forall \boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)\ .$$

[2] With a small abuse of notation, let $\min(x, y/0) = x$.

Note that $\lambda_{t+1} = \lambda_t + \frac{1}{\beta^2}\delta_t$. Hence, $\lambda_1 = 0$, $\lambda_2 = (\ell_1(\boldsymbol{x}_1) - \ell_1(\boldsymbol{x}_2))/\beta^2 \leq \sqrt{2}D\|\boldsymbol{g}_1\|_\star/\beta^2$, and

$$\lambda_{t+1} = \lambda_t + \frac{1}{\beta^2}\delta_t \leq \lambda_t + \frac{1}{\beta^2}\min\left(\sqrt{2}D\|\boldsymbol{g}_t\|_\star, \frac{\|\boldsymbol{g}_t\|_\star^2}{2\lambda_t}\right), \quad \forall t \geq 3\ .$$

Therefore, using Lemma 4.9 with $\Delta_t = \lambda_t$, $b = \frac{\sqrt{2}D}{\beta^2}$ and $c = \frac{1}{\beta^2}$, $a_t = \|\boldsymbol{g}_t\|_\star$, we get

$$\lambda_{T+1} \leq \sqrt{(2D^2/\beta^4 + 1/\beta^2)\sum_{t=1}^{T}\|\boldsymbol{g}_t\|_\star^2},$$

from which setting $\beta = D$ we obtain the second term in the min in Equation (4.21). □

This last theorem shows that Algorithm 6 can have a low regret if the temporal variability of the losses V_T is low. Moreover, differently from Optimistic Algorithms, Algorithm 6 does not need additional assumptions on the losses (for example smoothness), as done for example in Jadbabaie et al. [2015].

Lower Bound. Next, we are going to prove a lower bound in terms of the temporal variability V_T, which shows that the regret bound in Theorem 4.10 cannot be improved further. The proof is a simple modification of the standard arguments used to prove lower bounds for constrained OLO.

Theorem 4.11. *Let $d \geq 2$, $\|\cdot\|$ an arbitrary norm on $\mathbb{R}^d$, and $V = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\| \leq D/2\}$. Let $\mathcal{A}$ be a deterministic algorithm on V. Let T be any non-negative integer. Then, for any $\tau \geq 0$, there exists a sequence of convex loss functions $\ell_1(\boldsymbol{x}), \dots, \ell_T(\boldsymbol{x})$ with temporal variability equal to τ and $\boldsymbol{u} \in V$ such that the regret of algorithm $\mathcal{A}$ satisfies $R_T(\boldsymbol{u}) \geq \tau$.*

Proof. The first loss of the algorithm is $\ell_1(\boldsymbol{x}) = L\langle \boldsymbol{g}, \boldsymbol{x}\rangle$, where $\|\boldsymbol{g}\|_\star = 1$ and $\boldsymbol{g}$ is orthogonal to $\boldsymbol{x}_1$, while L will be set in the following. Note that $d \geq 2$ assures that $\boldsymbol{g}$ always exists. For $t \geq 2$, set $\ell_t(\boldsymbol{x}) = 0$. First, observe that $V_T = \max_{\boldsymbol{x}\in V} -\ell_1(\boldsymbol{x}) = L\max_{\boldsymbol{u}\in V} -\langle \boldsymbol{g}, \boldsymbol{u}\rangle = LD/2$. Hence, setting $L = 2\tau/D$, we have $V_T = \tau$. Also, we have $R_T(\boldsymbol{u}) = L\max_{\boldsymbol{u}\in V} -\langle \boldsymbol{g}, \boldsymbol{u}\rangle = V_T$. □

It is worth emphasizing that the lower bound does not contradict the upper bound of $\mathcal{O}(DL\sqrt{T})$ because in the above theorem L is chosen arbitrarily large.

4.5.1 Composite Losses

Next, we are going to show an application of our algorithm which might be relevant in practical scenarios. Assume that the losses received are composed by two parts: the first part is a convex function changing over time, while the second part is again a convex function but fixed and known to the algorithm. This losses are called *composite* [Duchi et al., 2010]. For example, we might have $\ell_t(\boldsymbol{x}) = \tilde{\ell}_t(\boldsymbol{x}) + \beta\|\boldsymbol{x}\|_1$. In this case, considering a bounded domain V the update rule for Algorithm 6 will be

$$\boldsymbol{x}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x}\in V}\ \tilde{\ell}_t(\boldsymbol{x}) + \beta\|\boldsymbol{x}\|_1 + B_\psi(\boldsymbol{x}, \boldsymbol{x}_t)/\eta_t, \tag{4.22}$$

which will promote sparsity in our model. Note that even if in practice often there are no closed form solutions to the minimization problem in Equation (4.22), approximate solutions can be found reasonably fast.

We next prove that Algorithm 6 satisfies a regret bound order of $\mathcal{O}(\min(V_T, \sqrt{T}))$.

Theorem 4.12. *Let $V \subset X \subseteq \mathbb{R}^d$ be a non-empty closed convex set. Let $\ell_t(\boldsymbol{x}) = \tilde{\ell}_t(\boldsymbol{x}) + \beta r(\boldsymbol{x})$, where $r : X \to \mathbb{R}$ is a convex function, and let $\boldsymbol{g}_t \in \partial\tilde{\ell}_t(\boldsymbol{x}_t)$ for all $t \in [T]$. Let $\lambda_t = 1/\eta_t$. Then, under the assumptions of Theorem 4.10, Algorithm 6 with $\lambda_1 = 0$ and $\lambda_t = \frac{1}{D^2}\sum_{i=1}^{t-1}(\tilde{\ell}_i(\boldsymbol{x}_i) - \tilde{\ell}_i(\boldsymbol{x}_{i+1}) - B_\psi(\boldsymbol{x}_{i+1}, \boldsymbol{x}_i))$ for $t = 2, \dots, T$ incurs the following regret*

bound for any $\boldsymbol{u} \in V$

$$R_T(\boldsymbol{u}) \leq \min\left\{2(\ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T),\ 2D\sqrt{3\sum_{t=1}^T \|\boldsymbol{g}_t\|_\star^2} + \beta[r(\boldsymbol{x}_1) - r(\boldsymbol{x}_{T+1})]\right\}, \tag{4.23}$$

Proof. First, let $\boldsymbol{g}'_t \in \partial\tilde{\ell}_t(\boldsymbol{x}_{t+1})$. Note that for any $\boldsymbol{u} \in V$ we have the following

$$\begin{aligned}
\eta_t(\ell_t(\boldsymbol{x}_{t+1}) - \ell_t(\boldsymbol{u})) &\leq \eta_t\langle \boldsymbol{g}'_t + \beta\nabla r(\boldsymbol{x}_{t+1}), \boldsymbol{x}_{t+1} - \boldsymbol{u}\rangle \\
&= \langle \eta_t\boldsymbol{g}'_t + \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_t) \\
&\quad + \eta_t\beta\nabla r(\boldsymbol{x}_{t+1}), \boldsymbol{x}_{t+1} - \boldsymbol{u}\rangle \\
&\quad - \langle \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} - \boldsymbol{u}\rangle \\
&\leq \langle \nabla\psi(\boldsymbol{x}_{t+1}) - \nabla\psi(\boldsymbol{x}_t), \boldsymbol{u} - \boldsymbol{x}_{t+1}\rangle \\
&= B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t),
\end{aligned}$$

where the second inequality derives from the optimality condition of the update rule.

Now, let $\delta_t = \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta_t}$ and $\tilde{\delta}_t = \tilde{\ell}_t(\boldsymbol{x}_t) - \tilde{\ell}_t(\boldsymbol{x}_{t+1}) - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta_t}$. Therefore, after adding $\ell_t(\boldsymbol{x}_t)$ on both sides, taking $\ell_t(\boldsymbol{x}_{t+1})$, dividing both sides by η_t and summing over time we get

$$\begin{aligned}
\sum_{t=1}^T (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u})) &\leq \sum_{t=1}^T \frac{B_\psi(\boldsymbol{u}, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}, \boldsymbol{x}_{t+1})}{\eta_t} + \sum_{t=1}^T \delta_t \\
&\leq \frac{D^2}{\eta_1} + D^2\sum_{t=2}^T\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) + \sum_{t=1}^T[\tilde{\delta}_t + \beta(r(\boldsymbol{x}_t) - r(\boldsymbol{x}_{t+1}))] \\
&= \frac{D^2}{\eta_T} + \sum_{t=1}^T \tilde{\delta}_t + \beta[r(\boldsymbol{x}_1) - r(\boldsymbol{x}_{T+1})]\ .
\end{aligned}$$

Using $\lambda_t = 1/\eta_t$, we can upper bound the regret as $R_T(\boldsymbol{u}) \leq (D^2 + \beta^2)\lambda_{T+1} + \beta[r(\boldsymbol{x}_1) - r(\boldsymbol{x}_{T+1})]$. The rest of the proof follows easily from Theorem 4.10. □

Compared to the bound given in Theorem 4.10, we can see that the result above contains a subtle difference: the term under the square root only contains the subgradients of the variable part of the loss functions. This setting was also studied in the implicit case in Song et al. [2018]. Compared to the results in Song et al. [2018], we have a regret bound which is adaptive to the sum of the subgradients, as just explained. Furthermore, their bound does not contain the temporal variability V_T, which could potentially lead to constant regret.

4.6 Empirical results

In this section, we compare the empirical performance of our algorithm AdaImplicit with standard baselines in online learning: OGD [Zinkevich, 2003], OGD with adaptive learning rate $\eta_t = \frac{\beta}{\sqrt{\sum_{i=1}^t \|\boldsymbol{g}_t\|_\star^2}}$ (AdaOGD) [McMahan and Streeter, 2010], and IOMD with $\eta_t = \beta/\sqrt{t}$ (Implicit) [Kulis and Bartlett, 2010].

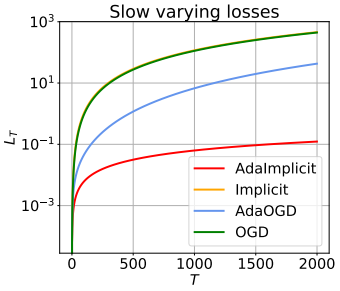


FIGURE 4.2: Synthetic experiment.

Synthetic Experiment. We first show the benefits of AdaImplicit on a synthetic dataset. The loss functions are chosen to have a small temporal variability V_T. In particular, we consider a 1-d case using $\ell_t(x) = \frac{1}{4}(x - y_t)^2$ with $y_t = 100\sin(\pi\frac{t}{10T})$, a time horizon $T = 2000$ and the L_2 ball of diameter $D = 150$. We set $\beta = 1$ in all algorithms. The update of the implicit algorithms can be computed in closed form: $x_{t+1} = x_t - \frac{\eta_t}{2+\eta_t}(x_t - y_t)$. In Figure 4.2 we show the cumulative loss $L_T = \sum_{t=1}^T \ell_t(x_t)$ of the algorithms (note that the y-axis is plotted in

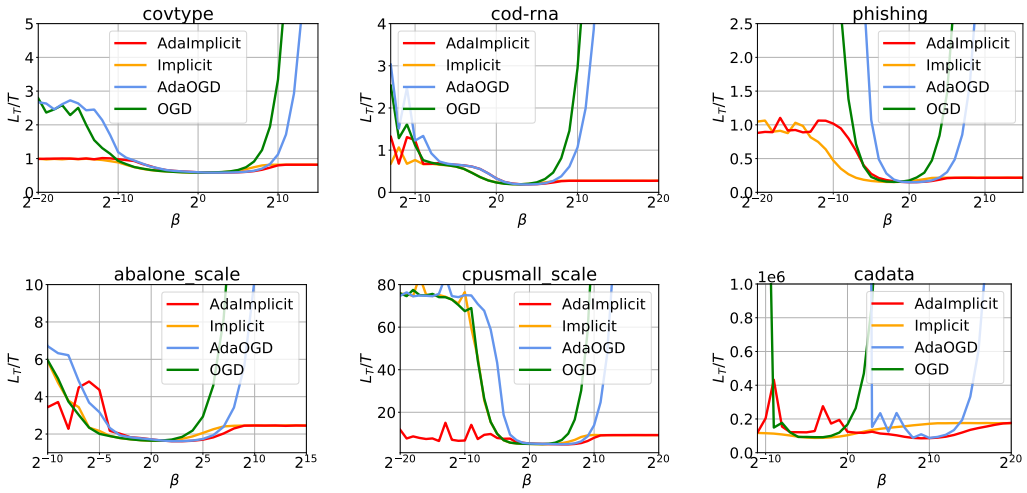


FIGURE 4.3: Plots on classification tasks using the hinge loss (top) and regression tasks using the absolute loss (bottom).

logarithmic scale). From the figure we can see that, contrarily to the other algorithms, the cumulative loss of AdaImplicit grows slowly over time, reflecting experimentally the bound given in Theorem 4.10. Also, even if not directly observable, OGD and IOMD basically incur the same total cumulative loss.

Real world datasets. We are now going to show some experiments conducted on real data. Here, there is no reason to believe that the temporal variability is small. However, we still want to verify if AdaImplicit can achieve a good worst-case performance. We consider both classification and regression tasks. Additional plots can be found in Appendix B.1.

We used datasets from the `LIBSVM` library [Chang and Lin, 2001]. Before running the algorithms, we preprocess the data by dividing each feature by its maximum absolute value so that all the values are in the range $[-1, 1]$, then we add a bias term. On the other hand, for regression tasks we do not take any preprocessing step but only add a bias term, since all the values are already in the range $[-1, 1]$. Details about the datasets can be found in Appendix B.1.

Given that in the online setting we cannot tune the hyperparameter β using hold-out data, we plot the average cumulative loss of each algorithm, i.e., $L_t/t = \frac{1}{t}\sum_{i=1}^{t} \ell_i(\boldsymbol{x}_i)$, as a function of the hyperparameter β. This allows us to evaluate at the same time the sensitivity of the algorithms to β and their best performance with oracle tuning. Note that in all the algorithms we consider the optimal worst-case setting of β to be proportional to the diameter of the feasible set. Hence, it is fair to plot their performance as a function of β. We consider values of β in $[2^{-20}, 2^{20}]$ with a grid containing 41 points. Then, each algorithm is run 10 times and results are averaged. For classification tasks we use the hinge loss, while for regression tasks we use the absolute loss. In both cases, we adopt the squared L_2 function for ψ.

Results. Experiments are illustrated in Figure 4.3. From the plots, we can see that when fine-tuned, all the algorithms achieve similar results, i.e., the minimum value of average cumulative loss is very close for all the algorithms considered and there is not a clear winner. However, note that the range of values which allows an algorithm to reach the minimum is considerably wider for Implicit algorithms and confirms their robustness regarding learning rate misspecification, as already investigated in other works [see, e.g., Toulis and Airoldi, 2017; Toulis et al., 2014]. This is a great advantage when considering online settings since, contrarily to the batch setting, algorithms cannot be fine-tuned in advance relying on training/validation sets.

4.7 Implicit Updates for FTRL

In this section, we show how to get a bound of $\mathcal{O}(V_T + 1)$ for FTRL employed with full losses. Unfortunately, contrarily to the OMD case, employing FTRL would entail solving a constrained convex optimization problem whose size (in terms of number of functions) grows each step. This would have a high running time even when the implicit updates have closed form expressions, e.g., linear classification with hinge loss. Furthermore, we show that it is not possible to adopt the same learning rate tuning strategy of *AdaImplicit* in order to get a similar regret bound.

We first remember the FTRL regret bound, stated in Chapter 3.

Lemma 3.10. *Let $V \subseteq \mathbb{R}^d$ be closed and non-empty. Given a sequence of convex loss functions $\ell_1, \dots, \ell_T$, set $F_1(\boldsymbol{x}) = \psi_1(\boldsymbol{x})$ and $F_t(\boldsymbol{x}) \triangleq \psi_t(\boldsymbol{x}) + \sum_{i=1}^{t-1} \langle \boldsymbol{g}_i, \boldsymbol{x} \rangle$, where $\boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)$. Assume that $\operatorname{argmin}_{\boldsymbol{x} \in V}$ is not empty and set $\boldsymbol{x}_t = \operatorname{argmin}_{\boldsymbol{x} \in V} F_t(\boldsymbol{x})$. Then, for any $\boldsymbol{u} \in V$, the regret is bounded as*

$$R_T(\boldsymbol{u}) \leq \sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u} \rangle \leq \psi_{T+1}(\boldsymbol{u}) - \min_{\boldsymbol{x} \in V} \psi_1(\boldsymbol{x}) + \sum_{t=1}^{T} \left[F_t(\boldsymbol{x}_t) - F_{t+1}(\boldsymbol{x}_{t+1}) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle \right] \ . \tag{3.18}$$

Note that the above lemma can also be stated with full losses rather than subgradients (the proof is the same and therefore omitted). In that case, $F_t(\boldsymbol{x}) = \psi_t(\boldsymbol{x}) + \sum_{i=1}^{t-1} \ell_i(\boldsymbol{x})$.

Now, assume that $\psi_t(\boldsymbol{x}) = \lambda_t \psi(\boldsymbol{x})$, with $(\lambda_t)_{t=1}^T$ being a non-decreasing sequence. We can rewrite the sum over time on the right-hand side of Equation (3.18) as follows

$$\begin{aligned}
&\sum_{t=1}^{T} \left[F_t(\boldsymbol{x}_t) - F_{t+1}(\boldsymbol{x}_{t+1}) + \ell_t(\boldsymbol{x}_t) \right] \\
&\quad = \sum_{t=1}^{T} \left[F_t(\boldsymbol{x}_t) + \ell_t(\boldsymbol{x}_t) - F_t(\boldsymbol{x}_{t+1}) - \ell_t(\boldsymbol{x}_{t+1}) + \psi_t(\boldsymbol{x}_{t+1}) - \psi_{t+1}(\boldsymbol{x}_{t+1}) \right] \\
&\quad \leq \sum_{t=1}^{T} \left[\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) + \psi_t(\boldsymbol{x}_{t+1}) - \psi_{t+1}(\boldsymbol{x}_{t+1}) \right] \\
&\quad = \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + \sum_{t=2}^{T} \left[\ell_t(\boldsymbol{x}_t) - \ell_{t-1}(\boldsymbol{x}_t) \right] + \sum_{t=1}^{T} (\lambda_t - \lambda_{t+1}) \psi(\boldsymbol{x}_{t+1}) \\
&\quad \leq \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T + \sum_{t=1}^{T} (\lambda_t - \lambda_{t+1}) \psi(\boldsymbol{x}_{t+1}) \ ,
\end{aligned}$$

where the first inequality derives from the fact that $F_t(\boldsymbol{x}_t) - F_t(\boldsymbol{x}_{t+1}) \leq 0$ while the last one from the definition of V_T. Therefore, the regret bound can be rewritten as follows

$$R_T(\boldsymbol{u}) \leq \lambda_T(\psi(\boldsymbol{u}) - \psi(\boldsymbol{x}_1)) + \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T + \sum_{t=1}^{T} (\lambda_t - \lambda_{t+1}) \psi(\boldsymbol{x}_{t+1})$$

We can see that with a constant learning rate the above expression would give a regret bound $\mathcal{O}(V_T + 1)$. On the other hand, it is known from the literature that a parameter $\lambda_t \propto \sqrt{t}$ would give a regret bound of $\mathcal{O}(\sqrt{T})$. Ideally, we would like to have a certain λ_t which allows to interpolate between a regret bound of $\mathcal{O}(V_T + 1)$ and $\mathcal{O}(\sqrt{T})$, as done for *AdaImplicit*. However, the techniques adopted in Section 4.5 do not seem to work in this case and one should hence resort to a different approach. In addition to the technical difficulties, as already stated earlier in this section, the computational burden of implicit updates with FTRL could be prohibitive in practice and makes this approach not worth of pursuing.

Chapter 5

Online Learning in Dynamic environments

In this chapter, we focus on online learning in the dynamic setting with *full-information* feedback, i.e, in every round the loss function is revealed. In the previous chapter, we have seen algorithms which have a sublinear regret upper bound against a fixed comparator. However, sometimes competing with the best fixed comparator is not meaningful. Indeed, there are situations where the environment is not stationary. In this case, rather than comparing the performance of an algorithm against a single benchmark, it is preferable to compete against a "moving" target, i.e., a sequence of different comparators. To model these scenarios, stronger notions of regret and various measures of the dynamic environment are used. Surprisingly, the quantity employed in the previous chapter used to quantify the improvements of Implicit updates has already been used in the context of dynamic online learning. Therefore, it seems natural to ask how Implicit algorithms behave in dynamic environments.

The notion of *dynamic regret* was first introduced in the seminal work of Zinkevich [2003]. Given a sequence $\boldsymbol{u}_{1:T} \triangleq (\boldsymbol{u}_1, \dots, \boldsymbol{u}_T)$, we define the dynamic regret against $\boldsymbol{u}_{1:T}$ as

$$R_T(\boldsymbol{u}_{1:T}) \triangleq \sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} \ell_t(\boldsymbol{u}_t) \ . \tag{5.1}$$

It can be shown that it is impossible to achieve sublinear dynamic regret in the worst-case. However, if one puts some restrictions on the sequence $\boldsymbol{u}_{1:T}$ and makes some assumptions on the regularity of the environment, then Equation (5.1) can be sublinear in T.

There are various measures which can be used to model the regularity of the environment. A natural measure of non-stationarity introduced in Zinkevich [2003] is the *path-length*[1] of the sequence $\boldsymbol{u}_{1:T}$,

$$C_T(\boldsymbol{u}_{1:T}) \triangleq \sum_{t=2}^{T} \|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\| \ . \tag{5.2}$$

Importantly, Zinkevich [2003] also proved that *Online Gradient Descent* incurs a regret bound of $\mathcal{O}(\sqrt{T}(1+C_T))$. This result was later extended by Hall and Willett [2013], who considered a modified (and possibly richer) definition of path-length.

Another measure of non-stationarity is given by the *temporal variability* of the loss functions [Besbes et al., 2015], as showed in the previous chapter. However, in the context of dynamic online learning the notion given in Equation (4.1) is slightly modified. Formally, let $\ell_{1:T}$ be the shorthand for $(\ell_1, \dots, \ell_T)$. The temporal variability of a sequence $\ell_{1:T}$ is defined as follows

$$V_T(\ell_{1:T}) \triangleq \sum_{t=2}^{T} \max_{\boldsymbol{x} \in V} |\ell_t(\boldsymbol{x}) - \ell_{t-1}(\boldsymbol{x})| \ . \tag{5.3}$$

In the remaining we will use the shorthands C_T for $C_T(\boldsymbol{u}_{1:T})$ and V_T for $V_T(\ell_{1:T})$ when the context is clear.

Another important definition is the one of *restricted dynamic regret* [Besbes et al., 2015; Jadbabaie et al., 2015; Yang et al., 2016]. In the restricted setting, the sequence of comparators

[1] One could also consider other versions of path-length, such as its squared version [Yang et al., 2016].

is given by the local minimizers of the loss functions, i.e., $\boldsymbol{u}_{1:T} = (\boldsymbol{u}_1^*, \dots, \boldsymbol{u}_T^*)$, where $\boldsymbol{u}_t^* = \operatorname{argmin}_{\boldsymbol{x} \in V} \ell_t(\boldsymbol{x})$.

5.1 Outline

We now describe the structure of this chapter, shortly highlighting existing results and new contributions.

A new lower bound. A lower bound involving the path-length of $\Omega(\sqrt{T(1+C_T)})$ is shown in Zhang et al. [2018a], where an algorithm which matches this bound is also given. Moreover, Yang et al. [2016] showed that the restricted case is actually easier compared to the general setting. Indeed, they proved that a simple strategy incurs regret bounded by $\mathcal{O}(C_T)$, with a matching lower bound. In Section 5.2 we begin to investigate what is achievable in dynamic settings in terms of V_T. While the simple strategy described in Yang et al. [2016] achieves an upper bound of $\mathcal{O}(V_T)$ on the restricted dynamic regret, to the best of our knowledge a lower bound for this setting is still not available. As our first contribution we provide a lower bound of $\Omega(V_T)$ for the setting of restricted dynamic regret. This automatically translates to a lower bound of $\Omega(V_T)$ for the harder general setting.

Implicit Updates in dynamic environments. In Section 5.3, we show how the Implicit algorithms based on OMD that we proposed in the previous chapter adapt to the dynamic setting. In particular, when an upper bound C_T' on the path-length of the sequence of comparators is fixed in advance, we show that Implicit algorithms optimally adapt to both V_T and C_T', achieving a regret bound of $\mathcal{O}(\min\{V_T, \sqrt{T(1+C_T')}\})$. On the other hand, existing algorithms cannot achieve the same goal. For example, Besbes et al. [2015] provided an analysis of restarted gradient descent in the setting of stochastic optimization with noisy gradients which incurs $\mathcal{O}(T^{2/3}(V_T'+1)^{1/3})$, where V_T' is an upper bound on V_T known in advance. Jadbabaie et al. [2015] gave an algorithm achieving a *restricted* dynamic regret of[2] $\tilde{\mathcal{O}}(\sqrt{G_T} + \min(\sqrt{(G_T+1)C_T}, ((G_T+1)T)^{1/3}(V_T+1)^{2/3}))$, where $G_T = \sum_{t=1}^T \|\nabla f_t(\boldsymbol{x}_t) - \boldsymbol{p}_t\|_\star^2$ and $\boldsymbol{p}_1, \dots, \boldsymbol{p}_T$ is a predictable sequence computable at the start of round t. Importantly, this bound is obtained without prior knowledge of D_T, C_T and V_T but under the assumption that all of them can be *observable*. If one limits the algorithm to not use predictable sequences, then the bound reduces to $\tilde{\mathcal{O}}(\sqrt{T} + \min\{\sqrt{T(1+C_T)}, T^{1/3}(V_T+1)^{2/3})\}$. In the same spirit of Jadbabaie et al. [2015], when the quantity of interests are observable we adapt our algorithm using a doubling trick, and prove an improved regret bound of $\min\{\sqrt{T(1+C_T)}, V_T\}$.

Connections to adaptive regret. The dynamic regret is only one way of studying dynamic environments. In recent years, other notions of regret have been provided for this setting. For example, Hazan and Seshadhri [2007] introduced the concept of *adaptive* regret,

$$\text{Adaptive-Regret}(T) = \max_{[s,e] \subseteq [1,T]} \left(\sum_{t=s}^{e} \ell_t(\boldsymbol{x}_t) - \min_{\boldsymbol{u} \in V} \sum_{t=s}^{e} \ell_t(\boldsymbol{u}) \right) . \tag{5.4}$$

In particular, an algorithm is called *adaptive*, if Equation (5.4) is bounded by $\tilde{\mathcal{O}}(\sqrt{T})$. Moreover, Hazan and Seshadhri [2007] provided an efficient adaptive algorithm. Its adaptive regret is bounded by $\mathcal{O}(\sqrt{T \ln^3 T})$ for convex losses and any interval $[s,e] \subseteq [1,T]$. The overall running time of their construction is $\mathcal{O}(T \ln T)$, therefore only a logarithmic factor worse than the running time of standard online learning algorithms, such as OGD. However, one limitation of the notion in Equation (5.4) is that it is vacuous for short intervals. To see this, note that if an interval has length $\mathcal{O}(\sqrt{T})$ then a regret bound of $\mathcal{O}(\sqrt{T \ln T})$ is meaningless (since it is linear in the length of the interval). In order to overcome this limitation, Daniely et al. [2015] proposed a stronger notion of adaptivity, which takes the length of the interval as a parameter. In particular, an algorithm is called *strongly-adaptive* if its regret is sublinear in the length of the interval, for any interval of any length. In other words, strongly adaptive

[2] The $\tilde{\mathcal{O}}$ notation hides poly-logarithmic terms.

Algorithm 7 Greedy optimizer

Require: Non-empty closed convex set $V \subset X \subset \mathbb{R}^d$, $\boldsymbol{x}_1 \in V$.
1: **for** $t = 1, \dots, T$ **do**
2: Output $\boldsymbol{x}_t \in V$
3: Receive $\ell_t : \mathbb{R}^d \to \mathbb{R}$ and pay $\ell_t(\boldsymbol{x}_t)$
4: Update $\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x} \in V} \ell_t(\boldsymbol{x})$
5: **end for**

algorithms obtain

$$\sum_{t=s}^{e} \ell_t(\boldsymbol{x}_t) - \min_{\boldsymbol{u} \in V} \sum_{t=s}^{e} \ell_t(\boldsymbol{u}) = \tilde{\mathcal{O}}(\sqrt{e-s}) \ , \tag{5.5}$$

for every interval $[s, e] \subseteq [1, T]$.

While there exist many strongly-adaptive algorithms, very recently Cutkosky [2020] proposed a new one, which also achieves the optimal dynamic regret in terms of path-length C_T for *any* sequence of comparators.

In Section 5.5 we first review the connections between strongly adaptive and dynamic regret. Then, we show how to combine the existing algorithm from Cutkosky [2020] with a greedy strategy, and prove a dynamic regret bound of $\mathcal{O}(\min\{V_T, \sqrt{T(1+C_T)}\})$ for any possible value of C_T.

5.2 A closer look at Temporal Variability

In this section, we turn our attention to the temporal variability of the loss functions in the full information setting. In particular, we are going to prove a lower bound of $\Omega(V_T)$ on the regret in the restricted dynamic scenario. We might directly apply the lower bound for the static case from the previous chapter to this setting, but we point out that the result in Theorem 4.11 only applies to deterministic algorithms. On the other hand, we consider a different approach which also applies to randomized algorithms.

Restricted setting. We now consider the case when the sequence of comparators is $\boldsymbol{u}_{1:T} = (\boldsymbol{u}_1^*, \dots, \boldsymbol{u}_T^*)$, with $\boldsymbol{u}_t^* = \operatorname{argmin}_{\boldsymbol{x} \in V} \ell_t(\boldsymbol{x})$ with full information feedback. A simple strategy to achieve a bound in terms of temporal variability V_T is depicted in Algorithm 7. Basically, in each round the algorithm plays the minimizer of the observed loss function in the previous round. While this has been noted in Jadbabaie et al. [2015], to the best of our knowledge a formal proof is not available. Hence, we provide the following theorem.

Theorem 5.1. *Consider a convex set $V \subseteq \mathbb{R}^d$ and any $\boldsymbol{x}_1 \in V$. Then, for any sequence of loss functions and any sequence of comparators $\boldsymbol{u}_{1:T}$, Algorithm 7 satisfies the following upper bound on the dynamic regret*

$$R_T(\boldsymbol{u}_{1:T}) \leq \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T \ . \tag{5.6}$$

Proof. For any $\boldsymbol{u}_t \in V$, we have that

$$\begin{aligned}
\sum_{t=1}^{T} (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u}_t)) &= \sum_{t=1}^{T} (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) + \ell_t(\boldsymbol{x}_{t+1}) - \ell_t(\boldsymbol{u}_t)) \\
&\leq \sum_{t=1}^{T} (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1})) = \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + \sum_{t=2}^{T} (\ell_t(\boldsymbol{x}_t) - \ell_{t-1}(\boldsymbol{x}_t)) \\
&\leq \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T
\end{aligned}$$

□

This result might seem to be in contrast with the result given in Besbes et al. [2015], who report a lower bound of $\Omega(V_T^{1/3} T^{2/3})$ on the regret in the restricted setting. However, it should be noted that in Besbes et al. [2015, Theorem 2] the feedback received by the algorithm

is different and not directly comparable to our setting. Indeed, Besbes et al. [2015] assume access only to noisy functions or gradients. Since we do not make such assumptions, their Theorem 2 is not applicable to our setting.

When taking into account the path-length, Yang et al. [2016] show that Algorithm 7 achieves an upper bound of $\mathcal{O}(\max(C_T, 1))$. They also show a matching lower bound in terms of C_T. On the other hand, a lower bound in terms of temporal variability is still missing. We next provide one, which shows that the upper bound achieved by Algorithm 7 is tight. The proof of this result is an adaptation of Proposition 1 from Yang et al. [2016].

Theorem 5.2. *Let $V = [-1, 1]$, and C be a positive constant independent of T. Then, for any algorithm $\mathcal{A}$ on V, and any $\sigma \in (1/\sqrt{T}, 1)$, there exists a sequence of loss functions $\ell_1, \dots, \ell_T$ with temporal variability less than or equal to $2\sigma T$ such that*

$$R(\boldsymbol{u}_{1:T}) \geq C V_T^{\gamma} \ , \tag{5.7}$$

for any $\gamma \in (0, 1)$.

Proof. As done in Yang et al. [2016], we consider a simple 1-d problem and employ the following sequence of loss functions. Define $\ell_t(x_t) = \frac{1}{2}(x_t - \varepsilon_t)^2$, where $\varepsilon_1, \dots, \varepsilon_T$ is a sequence of random variables sampled uniformly at random between the two values $\{-\sigma, \sigma\}$. Note that we have $\mathbb{E}[\varepsilon_t] = 0$ and $\mathrm{Var}(\epsilon_t) = \mathbb{E}[\varepsilon_t^2] = \sigma^2$. Obviously, the optimal choice in every round is $u_t = \varepsilon_t$. Assume $T \geq 1$. Then, the restricted dynamic regret is given by

$$\mathbb{E}\left[R_T(u_{1:T})\right] = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(x_t) - \ell_t(\varepsilon_t)\right] = \sum_{t=1}^{T} \frac{1}{2}\mathbb{E}[x_t^2] + \frac{1}{2}\mathbb{E}[\varepsilon^2] \geq \frac{\sigma^2}{2} T, \tag{5.8}$$

where the expectation is taken with respect to the randomness in the sequence of loss functions and any algorithm $\mathcal{A}$, while the inequality is due to the fact that x_t is independent from ε_t and $\mathbb{E}[\varepsilon_t] = 0$. Now, note that we can upper bound the temporal variability as follows

$$\begin{aligned}
V_T &= \sum_{t=1}^{T-1} \max_{x \in V} |\ell_t(x) - \ell_{t+1}(x)| \\
&= \sum_{t=1}^{T-1} \max_{x \in V} \left| \frac{1}{2}(x - \varepsilon_t)^2 - \frac{1}{2}(x - \varepsilon_{t+1})^2 \right| \\
&= \sum_{t=1}^{T-1} \max_{x \in V} |x(\varepsilon_{t+1} - \varepsilon_t)| \\
&\leq \sum_{t=1}^{T-1} |\varepsilon_{t+1} - \varepsilon_t| \\
&\leq 2\sigma T \ .
\end{aligned} \tag{5.9}$$

Observe that if we set $\sigma = C'/2$ for a positive constant C', then we recover the result in Proposition 1 of Besbes et al. [2015] which says that it is impossible to achieve sublinear dynamic regret unless $V_T = o(T)$.

The rest of the proof follows easily from Yang et al. [2016], but we report it here for completeness. We let $\sigma = T^{-\mu}$, with $\mu = (1 - \gamma)/(2 - \gamma)$ and $\mu \in (0, 1/2)$. Then, from Equation (5.8) we have that $R(u_{1:T}) \geq T^{1-2\mu}/2$, while from Equation (5.9) we have that $T \geq (V_T/2)^{\frac{1}{1-\mu}}$. Therefore, putting these results together we have that $R(u_{1:T}) \geq \frac{1}{2}(V_T/2)^{\frac{1-2\mu}{1-\mu}} = C V_T^{\gamma}$. Note that if $\gamma = 1$ then $\mu = 0$ and the regret must be linear in T. Therefore, we let $\gamma < 1$. □

Theorem 5.2 implies that it is impossible to achieve a dynamic regret bound better that $\mathcal{O}(V_T^{\gamma})$, with $\gamma < 1$. Note that this result holds even for randomized algorithms, contrarily to Theorem 4.11 given in the previous chapter. Moreover, it automatically applies to the general dynamic regret setting, which is harder.

Based on Theorem 5.2 and the result from Yang et al. [2016], it follows that the best policy to adopt in the restricted setting with full-information is the greedy algorithm given

Algorithm 8 Dynamic IOMD

Require: Non-empty closed convex set $V \subset X \subset \mathbb{R}^d$, $\psi : X \to \mathbb{R}$, $\boldsymbol{x}_1 \in V$, γ such that $B_\psi(\boldsymbol{x}, \boldsymbol{z}) - B_\psi(\boldsymbol{y}, \boldsymbol{z}) \leq \gamma \|\boldsymbol{x} - \boldsymbol{y}\|, \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V$, non increasing sequence $(\eta_t)_{t=1}^T$.

1: **for** $t = 1, \dots, T$ **do**
2: Output $\boldsymbol{x}_t \in V$
3: Receive $\ell_t : \mathbb{R}^d \to \mathbb{R}$ and pay $\ell_t(\boldsymbol{x}_t)$
4: Update $\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x} \in V} \ \ell_t(\boldsymbol{x}) + B_\psi(\boldsymbol{x}, \boldsymbol{x}_t)/\eta_t$
5: **end for**

in Algorithm 7, which at round t plays the minimizer of the previous loss function. However, the restricted dynamic regret could be a poor benchmark, as an upper bound for it could be very loose for another sequence of comparators whose path length is smaller. Furthermore, as explained in Zhang et al. [2018a] it could be meaningless in the problem of statistical machine learning, where the loss functions ℓ_t are sampled independently from the same distribution and minimizing the restricted dynamic regret could lead to overfitting. Therefore, we will shift our attention to the general dynamic regret in the following.

To the best of our knowledge there are no algorithms which achieve a dynamic regret bound of $\mathcal{O}(\min\{V_T, \sqrt{T(1 + C_T)}\})$ in the full information setting. Indeed, algorithms designed for dynamic regret such as Jadbabaie et al. [2015] and Besbes et al. [2015] have a dependency of $V_T^{1/3}$. The same holds true for strongly adaptive algorithms as well, as shown in Zhang et al. [2018b]. This is not really surprising: a bound of $\mathcal{O}(V_T)$ would imply constant regret in case the loss functions are fixed, i.e. $\ell_t = \ell$ for all t. In this case, using an online-to-batch conversion [Cesa-Bianchi et al., 2004] would result in a convergence rate of $\mathcal{O}(1/T)$. However, this would be in contrast with the lower bound by Nesterov [2004] on non-smooth batch black-box optimization. Unfortunately, all the algorithms mentioned above make use of gradients and therefore are subject to the lower bound of $\mathcal{O}(1/\sqrt{T})$. Therefore, in the dynamic setting the lower bound given in Besbes et al. [2015] would probably continue to hold for algorithms using only gradients, even if the feedback structure is changed (i.e. the noise assumptions are removed). However, not all is lost: in the next section, we illustrate how to achieve a bound in $\mathcal{O}(V_T)$ using an algorithm which makes full use of the loss function (and not just its gradient).

5.3 Implicit updates in dynamic environments

In this section, we show how OMD with Implicit updates described in the previous chapter adapts to dynamic settings. To this aim, we require a Lipschitz continuity condition on the Bregman divergence. For ease of exposition, we report the algorithm in Algorithm 8.

We first provide a generic dynamic regret bound for Algorithm 8. Then, we show that when an upper bound C'_T on the path-length of the sequence of comparators is fixed in advance, Implicit algorithms optimally adapt to both V_T and C'_T, achieving a dynamic regret bound of $\mathcal{O}(\min\{V_T, \sqrt{T(1 + C'_T)}\})$. Finally, when an upper bound C'_T is not known in advance, but the sequence of comparators is observable, we adapt our algorithm using a doubling trick and we show an upper bound on the dynamic regret of $\mathcal{O}(\min\{V_T, \sqrt{T(1 + C_T)}\})$.

Theorem 5.3. *Let $V \subset X \subset \mathbb{R}^d$ be non-empty closed convex sets, $\psi : X \to \mathbb{R}$, and $\boldsymbol{x}_1 \in V$. Assume there exists $\gamma \in \mathbb{R}$ such that $B_\psi(\boldsymbol{x}, \boldsymbol{z}) - B_\psi(\boldsymbol{y}, \boldsymbol{z}) \leq \gamma \|\boldsymbol{x} - \boldsymbol{y}\|, \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V$. Define $D^2 \triangleq \max_{\boldsymbol{x}, \boldsymbol{y} \in V} B_\psi(\boldsymbol{x}, \boldsymbol{y})$. Let $(\eta_t)_{t=1}^T$ be a non-increasing sequence. Then, the regret of Algorithm 8 against any sequence $\boldsymbol{u}_{1:T}$ with $\boldsymbol{u}_t \in V$ for all t is bounded as follows*

$$R_T(\boldsymbol{u}_{1:T}) \leq \frac{D^2}{\eta_T} + \gamma \sum_{t=2}^{T} \frac{\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|}{\eta_t} + \sum_{t=1}^{T} \delta_t, \tag{5.10}$$

where $\delta_t = \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)/\eta_t$.

Proof. Let $\boldsymbol{g}'_t \in \partial \ell_t(\boldsymbol{x}_{t+1})$. From the update rule of Algorithm 8 we have that

$$\eta_t(\ell_t(\boldsymbol{x}_{t+1}) - \ell_t(\boldsymbol{u}_t)) \leq \langle \eta_t \boldsymbol{g}'_t, \boldsymbol{x}_{t+1} - \boldsymbol{u}_t \rangle$$

$$\begin{aligned} &\leq \langle \nabla\psi(\boldsymbol{x}_t) - \nabla\psi(\boldsymbol{x}_{t+1}), \boldsymbol{x}_{t+1} - \boldsymbol{u}_t \rangle \\ &= B_\psi(\boldsymbol{u}_t, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}_t, \boldsymbol{x}_{t+1}) - B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t), \end{aligned} \tag{5.11}$$

where the first inequality follows from the convexity of the loss functions, while the second from the first-order optimality condition.

Now, we consider the first two terms of the r.h.s. of Equation (5.11). Using the Lipschitz continuity condition on the Bregman divergence and the fact that η_t is non-increasing over time, we get

$$\begin{aligned} &\sum_{t=1}^{T} \frac{1}{\eta_t}(B_\psi(\boldsymbol{u}_t, \boldsymbol{x}_t) - B_\psi(\boldsymbol{u}_t, \boldsymbol{x}_{t+1})) \\ &\quad\leq \frac{D^2}{\eta_1} + \sum_{t=2}^{T}\left(\frac{B_\psi(\boldsymbol{u}_t, \boldsymbol{x}_t)}{\eta_t} - \frac{B_\psi(\boldsymbol{u}_{t-1}, \boldsymbol{x}_t)}{\eta_{t-1}}\right) \\ &\quad= \frac{D^2}{\eta_1} + \sum_{t=2}^{T}\left(\frac{B_\psi(\boldsymbol{u}_t, \boldsymbol{x}_t)}{\eta_t} - \frac{B_\psi(\boldsymbol{u}_{t-1}, \boldsymbol{x}_t)}{\eta_t} + \frac{B_\psi(\boldsymbol{u}_{t-1}, \boldsymbol{x}_t)}{\eta_t} - \frac{B_\psi(\boldsymbol{u}_{t-1}, \boldsymbol{x}_t)}{\eta_{t-1}}\right) \\ &\quad\leq \frac{D^2}{\eta_1} + \gamma\sum_{t=2}^{T}\frac{\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|}{\eta_t} + \sum_{t=2}^{T} B_\psi(\boldsymbol{u}_{t-1}, \boldsymbol{x}_t)\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) \\ &\quad\leq \frac{D^2}{\eta_1} + D^2\left(\frac{1}{\eta_T} - \frac{1}{\eta_1}\right) + \gamma\sum_{t=2}^{T}\frac{\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|}{\eta_t} \\ &\quad= \frac{D^2}{\eta_T} + \gamma\sum_{t=2}^{T}\frac{\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|}{\eta_t}\ . \end{aligned}$$

Adding $\ell_t(\boldsymbol{x}_t)$ on both sides of Equation (5.11) and summing over time yields the regret bound in Equation (5.10). □

Notice that the Lipschitz continuity assumption is not a strong requirement. For example in the case of learning with expert advice, we have that $\gamma = \mathcal{O}(\ln T)$ if we use a "clipped" simplex (see, e.g., Jadbabaie et al. [2015]). In general, it is often not required to modify the domain of interest. Indeed, when the function ψ is Lipschitz on V, the Lipschitz condition on the Bregman divergence is automatically satisfied.

Using a fixed learning rate and assuming to fix in advance an upper bound to the path length of the sequence of comparators, we show the following regret bound on the dynamic regret.

Corollary 5.4. *Let $\boldsymbol{g}_t \in \partial\ell_t(\boldsymbol{x}_t)$ and assume $\|\boldsymbol{g}_t\|_\star \leq L$. Let $C'_T \geq 0$ be a fixed positive constant, then using $\eta_t = \eta = \frac{1}{L}\sqrt{\frac{2(D^2+\gamma C'_T)}{T}}$, Algorithm 8 gives the following regret bound against all sequence $\boldsymbol{u}_{1:T}$ whose path length C_T is upper bounded by C'_T*

$$R_T(\boldsymbol{u}_{1:T}) \leq L\sqrt{(D^2 + \gamma C'_T)T}\ . \tag{5.12}$$

Proof. Using a constant learning rate $\eta_t = \eta$ for all t, from Equation (5.10) we get

$$\begin{aligned} R_T(\boldsymbol{u}_{1:T}) &\leq \frac{D^2}{\eta} + \gamma\sum_{t=2}^{T}\frac{\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|}{\eta} + \sum_{t=1}^{T}\delta_t \\ &\leq \frac{D^2}{\eta} + \gamma\sum_{t=2}^{T}\frac{\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|}{\eta} + \sum_{t=1}^{T}\frac{\eta}{2}\|\boldsymbol{g}_t\|_\star^2 \\ &\leq \frac{D^2}{\eta} + \frac{\gamma}{\eta}C'_T + \eta\frac{L^2 T}{2}, \end{aligned}$$

where in the second inequality we used Theorem 4.4, while the last inequality follows from the assumption that $\|\boldsymbol{g}_t\|_\star \leq L$. Therefore, using the given learning rate we get the bound given in Equation (5.12). □

From the above result, we can see that the upper bound matches the lower bound given in Zhang et al. [2018a] for all sequence whose path-length C_T is equal to C'_T.

Temporal Variability. While Corollary 5.4 shows a regret bound in terms of the path-length C_T, it is still not clear how OMD with Implicit updates shown in Algorithm 8 can adapt to the temporal variability. Fortunately, in the previous chapter we showed that a carefully chosen adaptive learning rate gives a bound in terms of V_T, in the static setting (see Theorem 4.10). With the same technique, we can achieve a similar bound in the dynamic setting. The next theorem shows how this can be done using Algorithm 8.

Theorem 5.5. *Under the assumptions of Theorem 5.3, for any $\boldsymbol{u}_{1:T}$ such that $\boldsymbol{u}_t \in V$, running Algorithm 8 with $1/\eta_t = \lambda_t = \frac{1}{\beta^2}\sum_{i=1}^{t-1}\delta_t$ guarantees*

$$R_T(\boldsymbol{u}_{1:T}) \leq \frac{D^2 + \beta^2 + \gamma C_T}{\beta^2}\min(B_1, B_2),$$

where $B_1 = \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T$ and $B_2 = \sqrt{(2D^2+\beta^2)\sum_{t=1}^T \|\boldsymbol{g}_t\|_\star^2}$

Proof. We can rewrite the bound in Equation (5.10) as follows

$$\begin{aligned}
R_T(\boldsymbol{u}_{1:T}) &\leq \lambda_T D^2 + \gamma\sum_{t=2}^T \lambda_t\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\| + \beta^2\lambda_{T+1} \\
&\leq (D^2+\beta^2)\lambda_{T+1} + \gamma\lambda_{T+1}\sum_{t=2}^T \|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\| \\
&= (D^2 + \beta^2 + \gamma C_T)\lambda_{T+1},
\end{aligned}$$

where in the second inequality we have used the fact that $(\lambda_t)_{t=1}^T$ is an increasing sequence.

From the choice of λ_t, we have that

$$\beta^2\lambda_{T+1} \leq \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T, \tag{5.13}$$

where $V_T = \sum_{t=2}^T \max_{\boldsymbol{x}\in V} \ell_t(\boldsymbol{x}) - \ell_{t-1}(\boldsymbol{x})$. On the other hand, using Lemma 4.9 we get

$$\beta^2\lambda_{T+1} \leq \sqrt{(2D^2+\beta^2)\sum_{t=1}^T \|\boldsymbol{g}_t\|_\star^2}\,. \tag{5.14}$$

Therefore, putting together Equations (5.13) and (5.14) we get the stated bound. □

We can see that the bound in Theorem 5.5 still requires the parameter β to be tuned. When competing against a sequence of comparators whose path length C_T is fixed beforehand, then the value C_T can be used in the tuning. Setting $\beta^2 = (D^2 + \gamma C_T)$, we get the following corollary. Note that the algorithm resulting from Theorem 5.5 is identical to *Adaimplicit* from last chapter, except for the choice of the parameter β.

Corollary 5.6. *Let $C'_T \geq 0$ be a positive constant. Then, running Algorithm 8 with $1/\eta_t = \lambda_t = \frac{1}{\beta^2}\sum_{i=1}^{t-1}\delta_t$ and $\beta^2 = (D^2 + \gamma C'_T)$, the dynamic regret against any sequence $\boldsymbol{u}_{1:T}$ whose path length C_T is less or equal than C'_T is upper bounded as follows*

$$R_T(\boldsymbol{u}_{1:T}) \leq \min\left\{2(\ell_1(\boldsymbol{x}_1) - \ell_t(\boldsymbol{x}_{T+1}) + V_T),\, 2\sqrt{(3D^2+\gamma C'_T)\sum_{t=1}^T \|\boldsymbol{g}_t\|_\star^2}\right\}.$$

The above result gives us an algorithm whose general dynamic regret is upper bounded by $\mathcal{O}(V_T, \sqrt{T(1+C_T)'})$. Note that C'_T is an upper bound to the path-length of the sequences of comparators given beforehand. Next, we are going to show how to extend this algorithm when the path-length of the sequence of comparators is not known beforehand, but can be observed.

Algorithm 9 Dynamic AdaImplicit

Require: Non-empty closed convex set $V \subset X \subset \mathbb{R}^d$, $\psi : X \to \mathbb{R}$, $\boldsymbol{x}_1 \in V$, γ such that $B_\psi(\boldsymbol{x}, \boldsymbol{z}) - B_\psi(\boldsymbol{y}, \boldsymbol{z}) \leq \gamma \|\boldsymbol{x} - \boldsymbol{y}\|, \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V$, $\beta_0^2 > 0$

$i \leftarrow 0$, $\lambda_1^0 \leftarrow 0$, $Q_0 \leftarrow \sqrt{2}D$, $C_0 \leftarrow 0$
for $t = 1, \dots, T$ **do**
  Output $\boldsymbol{x}_t \in V$
  Receive $\ell_t : \mathbb{R}^d \to \mathbb{R}$ and pay $\ell_t(\boldsymbol{x}_t)$
  Update $C_i \leftarrow C_i + \|\boldsymbol{u}_t^\star - \boldsymbol{u}_{t-1}^\star\|$, where $\boldsymbol{u}_t^\star := \operatorname{argmin}_{\boldsymbol{x} \in V} \ell_t(\boldsymbol{x})$
  if $C_i > Q_i$ **then**
    $i \leftarrow i + 1$
    $Q_i \leftarrow \sqrt{2}D2^i$, $\lambda_{t+1}^i \leftarrow 0$, $C_i \leftarrow 0$, $\beta_i^2 \leftarrow D^2 + \gamma Q_i$
    Update $\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}_t$
  else
    Update $\boldsymbol{x}_{t+1} \leftarrow \arg\min_{\boldsymbol{x} \in V} \ \ell_t(\boldsymbol{x}) + \lambda_t^i B_\psi(\boldsymbol{x}, \boldsymbol{x}_t)$
    Set $\delta_t \leftarrow \ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{x}_{t+1}) - \lambda_t^i B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)$
    Update $\lambda_{t+1}^i \leftarrow \lambda_t^i + \frac{1}{\beta_i^2}\delta_t$
  end if
end for

5.3.1 Adapting on the fly

The result given in the previous paragraph was limited to all sequences of comparators whose path-length is fixed beforehand. Following Jadbabaie et al. [2015], the approach given above can be generalized to any sequence of $\boldsymbol{u}_{1:T}$ whose path length C_T can be calculated on the fly. Formally, we denote by Π a set of strategies. Each $\pi = (\pi_1, \dots, \pi_T) \in \Pi$ is a sequence of mappings such that $\pi_t : \mathcal{F}^{t-1} \to V$, where $\mathcal{F}^{t-1}$ denotes the history of the game up to time $t-1$. For example Π could be the set of strategies where the sequence $\boldsymbol{u}_{1:T}$ is allowed to switch only k times, i.e. $\Pi = \{\boldsymbol{u}_{1:T}, \boldsymbol{u}_t \in V : \sum_{t=1}^T \mathbb{1}\{\boldsymbol{u}_t \neq \boldsymbol{u}_{t-1}\} \leq k-1\}$. In this case the path-length C_T might not be known a priori, but we can adapt Algorithm 8 to this setting using a doubling trick. We depict this strategy in Algorithm 9.

Doubling trick. The idea is to run Algorithm 8 in phases and tune the learning rate λ_t appropriately. At the beginning of each phase i, we start monitoring the path length C_i. Once it reaches a certain threshold, we restart the algorithm doubling the threshold. Formally, we introduce a quantity Q_i for phase i and set the learning rate λ_t of the algorithm as $\lambda_t^i = \frac{1}{\beta_i^2}\sum_{s=1}^{t-1}\delta_s$, with $\beta_i^2 = D^2 + \gamma Q_i$. The resulting algorithm is shown in Algorithm 9.

We are now going to analyze the regret bound incurred by Algorithm 9. First, we need the following lemma which bounds the number of times the algorithm is restarted.

Lemma 5.7. *Let t_i be the first time-step of epoch i, with $t_0 = 1$. Suppose Algorithm 9 is run for a total of $N+1$ epochs. Let $C_i = \sum_{t=t_i}^{t_{i+1}-1} \|\boldsymbol{u}_t^\star - \boldsymbol{u}_{t-1}^\star\|$, with $\|\boldsymbol{u}_1^\star - \boldsymbol{u}_0^\star\| \triangleq 0$. Let $C_T = \sum_{i=0}^N C_i$. Then, we have that N satisfies*

$$N \leq \log_2\left(\frac{C_T}{\sqrt{2}D} + 1\right) . \tag{5.15}$$

Proof. First, recall that $\sum_{i=0}^{N-1} a^i = \frac{a^N - 1}{a-1}$. Now, note that the sum in the first N epochs of the quantity we are monitoring is at most equal to the final sum over all $N+1$ epochs.

Therefore, we have the following

$$\sum_{i=0}^{N-1} \sqrt{2}D2^i \leq \sqrt{2}D(2^N - 1) \leq \sum_{i=0}^{N} \sum_{t=t_i}^{t_{i+1}-1} \|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\| = C_T,$$

where $\|\boldsymbol{u}_{t_0} - \boldsymbol{u}_{t_0-1}\| = \|\boldsymbol{u}_1 - \boldsymbol{u}_0\| \triangleq 0$ by definition. Solving for N yields the desired result. □

Next, we provide a theorem which gives a regret bound to Algorithm 9.

Theorem 5.8. *Let $V \subset \mathbb{R}^d$ be a non-empty closed convex set. Fix a class of strategies Π, where each strategy $\pi \in \Pi$ is such that $\pi = (\pi_1, \dots, \pi_T)$ and $\pi_t : \mathcal{F}^{t-1} \to V$. Assume Algorithm 9 is run for N epochs. Then, under the assumptions of Theorem 5.3 the regret against any strategy $\pi \in \Pi$ is bounded as*

$$R_T(\pi) \leq (2+c)\min\Bigg(\big(\ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T\big), \sqrt{\Big(3D^2\left(\log_2 \tfrac{C_T}{\sqrt{2}D} + 1\right) + \gamma C_T\Big)\textstyle\sum_{t=1}^T \|\boldsymbol{g}_t\|_\star^2}\Bigg), \tag{5.16}$$

where $c \triangleq \frac{\sqrt{2}}{D+\gamma\sqrt{2}}$ and $C_T = \sum_{t=2}^{T-1} \|\pi_t(\ell_{1:t-1}) - \pi_{t-1}(\ell_{1:t-2})\|$.

Proof. Let $V_i = \sum_{t=t_i+1}^{t_{i+1}-1} \max_{\boldsymbol{x}\in V} |\ell_t(\boldsymbol{x}) - \ell_{t-1}(\boldsymbol{x})|$. Using the result from Theorem 5.5, assuming the knowledge of C_i during each phase i we have that

$$\begin{aligned} R(\boldsymbol{u}^\star_{1:T}) &= \sum_{i=0}^{N}\sum_{t=t_i}^{t_{i+1}-1} R(\boldsymbol{u}^\star_{t_i:t_{i+1}-1}) \\ &\leq \sum_{i=0}^{N}\sum_{t=t_i}^{t_{i+1}-1} \frac{D^2+\gamma C_i+\beta_i^2}{\beta_i^2}\min(B_1^i, B_2^i), \end{aligned}$$

where in the last inequality we used Theorem 5.5 with $B_1^i = \ell_{t_i}(\boldsymbol{x}_1) - \ell_{t_{i+1}-1}(\boldsymbol{x}_{t_{i+1}}) + V_i$ and $B_2 = \sqrt{(2D^2+\beta^2)\sum_{t=t_i}^{t_{i+1}-1}\|\boldsymbol{g}_t\|_\star^2}$.

Note that

$$\frac{D^2+\gamma C_i+\beta_i^2}{\beta_i^2} = \frac{D^2+\gamma C_i + D^2 + \gamma Q_i}{D^2+\gamma Q_i} = 2 + \gamma\frac{(C_i - Q_i)}{D^2+\gamma Q_i} \leq 2 + \gamma\frac{\sqrt{2}D}{D^2+\gamma\sqrt{2}D2^i},$$

where the last inequality derives from the fact that the last term in C_i which causes the algorithm to restart is such that $\|\boldsymbol{x}-\boldsymbol{y}\| \leq \sqrt{2}D$, $\forall \boldsymbol{x}, \boldsymbol{y} \in V$.

Therefore, we have

$$\begin{aligned} R(\boldsymbol{u}_{1:T}) &\leq \sum_{i=0}^{N}\sum_{t=t_i}^{t_{i+1}-1}\left(2+\gamma\frac{\sqrt{2}}{D+\gamma 2^{i+\frac{1}{2}}}\right)\min(B_1^i, B_2^i) \\ &\leq \sum_{i=0}^{N}(2+c)\min\left\{\ell_{t_i}(\boldsymbol{x}_{t_i}) - \ell_{t_{i+1}-1}(\boldsymbol{x}_{t_{i+1}}) + V_i, \sqrt{(3D^2+\gamma\sqrt{2}D2^i)\sum_{t=t_i}^{t_{i+1}-1}\|\boldsymbol{g}_t\|_\star^2}\right\} \\ &\leq (2+c)\sum_{i=0}^{N}\min\left\{\ell_{t_i}(\boldsymbol{x}_{t_i}) - \ell_{t_{i+1}-1}(\boldsymbol{x}_{t_{i+1}}) + V_i, \sqrt{(3D^2+\gamma C_i)\sum_{t=t_i}^{t_{i+1}-1}\|\boldsymbol{g}_t\|_\star^2}\right\} \\ &\leq (2+c)\min\Bigg\{\underbrace{\sum_{i=0}^{N}(\ell_{t_i}(\boldsymbol{x}_{t_i}) - \ell_{t_{i+1}-1}(\boldsymbol{x}_{t_{i+1}}) + V_i)}_{(a)}, \underbrace{\sum_{i=0}^{N}\sqrt{(3D^2+\gamma C_i)\sum_{t=t_i}^{t_{i+1}-1}\|\boldsymbol{g}_t\|_\star^2}}_{(b)}\Bigg\}, \end{aligned}$$

where in the the second inequality we used the definition of c. We now analyze (a) and (b) separately.

For (b), using the Cauchy-Schwartz inequality we have that

$$\begin{aligned} \sum_{i=0}^{N}\sqrt{(3D^2+\gamma C_i)\sum_{t=t_i}^{t_{i+1}-1}\|\boldsymbol{g}_t\|_\star^2} &\leq \sqrt{\sum_{i=0}^{N}(3D^2+\gamma C_i)}\cdot\sqrt{\sum_{i=0}^{N}\sum_{t=t_i}^{t_{i+1}-1}\|\boldsymbol{g}_t\|_\star^2} \\ &= \sqrt{3ND^2+\gamma C_T}\cdot\sqrt{\sum_{t=1}^{T}\|\boldsymbol{g}_t\|_\star^2} \end{aligned}$$

$$\leq \sqrt{\left(3D^2\left(\log_2 \frac{C_T}{\sqrt{2}D} + 1\right) + \gamma C_T\right) \sum_{t=1}^{T} \|\boldsymbol{g}_t\|_\star^2} \ .$$

On the other hand, for (a) we have

$$\sum_{i=0}^{N} (\ell_{t_i}(\boldsymbol{x}_{t_i}) - \ell_{t_{i+1}-1}(\boldsymbol{x}_{t_{i+1}}) + V_i) \leq \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + V_T \ .$$

Therefore, putting together the results for (a) and (b), we get the stated bound. □

To summarize, for sequence of comparators whose path-length is observable, from Equation (5.16) we have a worst-case regret bound of

$$R(\boldsymbol{u}_{1:T}) = \tilde{\mathcal{O}}\left(\min\left\{V_T, \sqrt{T(1+C_T)}\right\}\right) \ . \tag{5.17}$$

In light of this last result, compared to Jadbabaie et al. [2015], our upper bound from Theorem 5.8 strictly improves their result when optimistic predictions are not helpful.

We stress that a doubling trick is necessary for Algorithm 9. Indeed, in order to have a fully adaptive learning rate, we should be able to tune it as a function of two quantities varying over time, namely the path-length observed and the temporal variability of the losses paid by the algorithm. While both quantities are increasing quantities over time, they also appear both at the numerator and denominator of the learning rate λ_t. However, this would result in a non-monotone sequence of learning rates, thus contradicting the assumptions in Theorem 5.3. Also, we would like to point out that to the best of our knowledge there are no existing methods in the literature which tune the learning rates with non-monotone sequences.

All the results up to this point are given under the assumption that the class of strategies we want to compete against is fixed before the start of the game, i.e., C_T (or an upper bound to it) is known beforehand or can be computed through observable quantities. This can be limiting in practice. On the other hand, we would like to note that online learning algorithms can be used in the offline setting as well. At least in the case of static regret, one can still guarantee convergence rates using standard arguments for online-to-batch conversions [Cesa-Bianchi et al., 2004]. Therefore, in the case of offline optimization, one should be able to select the optimal β in Corollary 5.6 by trying different values, in order to guarantee the regret bound given in Corollary 5.6

Nevertheless, in a truly realistic online setting, knowing the right C_T beforehand might be impossible. Therefore, in the next section we are going to provide an algorithm which can adapt to the values of C_T for any possible sequence of comparators but at the same time guarantees a bound in V_T.

5.4 Strongly Adaptive Regret

As already mentioned at the beginning of this chapter, a parallel line of work on non-stationary environments involves the study of the weakly and strongly-adaptive regret [Hazan and Seshadhri, 2007; Daniely et al., 2015], which aims to minimize the static regret over any possible (sub)interval over the time horizon T. In other words, for any interval $[s, e] \subset [1, T]$ a strongly-adaptive algorithm has to maintain a regret bound of $\mathcal{O}(\sqrt{(e-s)})$.

The dominant approach in the design of strongly adaptive algorithms has been the following. Consider an anytime algorithm $\mathcal{A}$ with static regret bound of $\mathcal{O}(\sqrt{t})$ for the interval $[1, t]$. At each time-step t initialize a new copy of $\mathcal{A}$. Then, to come up with a prediction at round t, use an expert algorithm $\mathcal{B}$ to combine the predictions of the t existing base learners. The resulting strategy is depicted in Algorithm 10.

For the sake of the analysis, from this point on we assume the losses to be bounded in $[0, 1]$. Using Algorithm 10, the regret over an interval $[s, e]$ can be decomposed as follows

$$\sum_{t=s}^{e} (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u})) \leq \sum_{t=s}^{e} \left(\sum_{i=1}^{t} p_{t,i} \ell_t(\boldsymbol{x}_t^i) - \ell_t(\boldsymbol{u})\right)$$

Algorithm 10 Strongly-adaptive algorithm

Require: Non-empty closed convex set $V \subset \mathbb{R}^d$, OLO algorithm $\mathcal{A}$, expert algorithm $\mathcal{B}$
1: **for** $t = 1, \dots, T$ **do**
2: Receive predictions from $\mathcal{A}_1, \dots, \mathcal{A}_t$, denoted $\boldsymbol{x}_t^1, \dots, \boldsymbol{x}_t^t$
3: Receive distribution $\boldsymbol{p}_t$ from $\mathcal{B}$
4: Output $\boldsymbol{x}_t = \sum_{i=1}^t p_{t,i} \boldsymbol{x}_t^i$
5: Observe loss ℓ_t and pay $\ell_t(\boldsymbol{x}_t)$
6: Pass ℓ_t to $\mathcal{A}_1, \dots, \mathcal{A}_t$
7: Set $g_{t,i} = \ell_t(\boldsymbol{x}_t^i)$ and pass $\boldsymbol{g}_t$ to $\mathcal{B}$
8: **end for**

$$
\begin{aligned}
&= \sum_{t=s}^{e} (\langle \boldsymbol{p}_t, \boldsymbol{g}_t \rangle - \ell_t(\boldsymbol{x}_t^s)) + \sum_{t=s}^{e} (\ell_t(\boldsymbol{x}_t^s) - \ell_t(\boldsymbol{u})) \\
&= R^{\mathcal{B}}_{[s,e]}(\boldsymbol{e}_s) + \mathcal{O}(\sqrt{e-s}) \\
&= \mathcal{O}(\sqrt{(e-s)\ln T}) + \mathcal{O}(\sqrt{e-s}) \qquad (5.18) \\
&= \mathcal{O}(\sqrt{(e-s)\ln T}, \qquad (5.19)
\end{aligned}
$$

where the first inequality derives from Jensen's inequality, the third from the regret bound of $\mathcal{A}_s$ and the last from the regret bound of the expert algorithm. Therefore, with this construction for any interval $[s, e] \subset [1, T]$ we have a regret bound of $\tilde{\mathcal{O}}(\sqrt{e-s})$.

However, by looking carefully at Algorithm 10, we can see that the number of experts (i.e., base learners) is growing in every round and it is not clear how to use a regular algorithm for the setting of Learning with Expert Advice as $\mathcal{B}$. Instead, in order to solve this problem algorithm $\mathcal{B}$ should be suited for the "sleeping experts" setting (details in Appendix A).

Running time. From the construction depicted in Algorithm 10, it seems that in every time-step t the algorithm needs to mantain t copies of the base algorithm $\mathcal{A}$. This would lead to a quadratic overall running time $\mathcal{O}(T^2)$. In fact, this can be improved. Indeed, as illustrated in Daniely et al. [2015] it is possible to get the same bound as in Equation (5.18) only running at most $\ln T$ algorithms at the same time. The technique is called the *geometric covering intervals*, and we sketch it next.

Let N be the smallest integer such that $T \leq 2^N$ (so that $N = \mathcal{O}(\log_2 T)$). For each $i = 0, 1, \dots, N$, we keep a set of disjoint intervals such that $\bigcup_{I \in S_i} I = [1, T]$. In particular, the set S_i consists of all intervals of length 2^i starting at a multiple of 2^i. contains at most $\mathcal{O}(T/2^i)$ intervals. Since at most $\mathcal{O}(\log_2(t))$ intervals contain any given time point t, the time complexity of Algorithm 10 is a factor $\mathcal{O}(\log_2(t))$ larger than that of the black-box $\mathcal{A}$. Moreover, any given index t is contained within at most one interval in each S_i. The construction is illustrated in Figure 5.1. It can be shown that the regret of the algorithm on any interval can be effectively decomposed (see for example Jun et al. [2017]) and standard techniques can be applied, as the ones sketched at the beginning of this section.

5.4.1 Dynamic regret of strongly adaptive methods

It has been shown that strongly-adaptive regret bounds imply dynamic regret bounds. In this section, we are going to sketch the arguments used to prove these results.

Path-length bounds. Very recently, Cutkosky [2020] provided a strongly-adaptive algorithm that achieves the optimal dynamic regret bound in terms of path-length, for any sequence of comparators. The argument used to prove the optimal dynamic regret bound is general and could be applied to other strongly adaptive algorithms too. We report it here for completeness, but first provide a useful lemma that we will use in the proof.

Lemma 5.9 ([Cutkosky, 2020])**.** *Consider a set $V \subset \mathbb{R}^d$ such that $\max_{\boldsymbol{x},\boldsymbol{y} \in V} \|\boldsymbol{x} - \boldsymbol{y}\| \leq D$. Define a time interval $I = [s, e]$ and let $\boldsymbol{u}_t \in V$ for any $t \in I$. Let $C_I = \sum_{t=s+1}^{e} \|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|$. Then it is possible to break the interval I into K disjoint intervals $I = J_1 \cup \dots \cup J_K$ such that*

for each i we have that

$$C_{J_i} \leq 2D \ . \tag{5.20}$$

Furthermore, we have that $K \leq \frac{C_I+D}{D}$.

Proof. We can build the set of subintervals $J_1, \ldots, J_K$ iteratively. Define $J_1 = [s, t_1]$ as the interval such that t_1 is the first time-step when $\sum_{t=s+1}^{t_1} \|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\| \geq D$. Then we have that $C_{J_1} \leq 2D$ (since $\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\| \leq D$ in any step t). We can repeat this process: given t_{i-1}, let t_i be the time-step such that $C_{[t_{i-1},t_i]} \geq D$ and define $J_i = [t_{i-1}, t_i]$. If such a t_i does not exist then set $i = K$ and $t_i = e$. Then for any subinterval we have that $C_{J_i} \leq 2D$.

On the other hand, we have $C_{J_i} \geq D$ for all i but the last one. Assume that there are K subintervals. We have that $\sum_{i=1}^{K} C_{J_i} \leq C_I$. Therefore,

$$C_I \geq \sum_{i=1}^{K} C_{J_i} \geq (K-1)D \ ,$$

from which the desired result follows. □

We can now prove that main result.

Theorem 5.10 ([Cutkosky, 2020, Theorem 7])**.** *Assume there exists an algorithm which for any interval $I = [s, e] \subseteq [1, T]$ guarantees a dynamic regret*

$$R_I(\boldsymbol{u}_{s:e}) = \tilde{\mathcal{O}}\left((D + C_I)\sqrt{1 + \sum_{t \in I} \|\boldsymbol{g}_t\|_\star}\right) \ . \tag{5.21}$$

Then, for any interval it also guarantees

$$R_I(\boldsymbol{u}_{s:e}) \leq \tilde{\mathcal{O}}\left(C_I + D + \sqrt{D(C_I + D)\sum_{t=s}^{e} \|g_t\|_\star}\right) \ , \tag{5.22}$$

where $C_I = \sum_{t=s+1}^{e} \|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|$ and furthermore we have that $C_I \leq \sqrt{DC_I|I|}$.

Proof. Let $J_1, \ldots, J_K$ be the set of disjoint intervals resulting from the construction in Lemma 5.9. We have that $I = J_1 \cup \cdots \cup J_K$, such that for each i we have $C_{J_i} \leq 2D$, and $K \leq \frac{C_I+D}{D}$. Now, on each of intervals $J_i = [s_i, e_i]$, using Equation (5.21) the regret is bounded as

$$R(\boldsymbol{u}_{s_i,e_i}) \leq \tilde{\mathcal{O}}\left((D + C_{J_i})\sqrt{1 + \sum_{t=s_i}^{e_i} \|\boldsymbol{g}_t\|_\star^2}\right) = \tilde{\mathcal{O}}\left(D\sqrt{1 + \sum_{t=s_i}^{e_i} \|\boldsymbol{g}_t\|_\star^2}\right) \ ,$$

where in the last inequality we used the assumption that $C_{J_i} \leq 2D$.

Then, using the Cauchy-Schwartz inequality we have that for interval $I = [s, e]$

$$\begin{aligned}
R(\boldsymbol{u}_{s:e}) = \sum_{i=1}^{K} R(\boldsymbol{u}_{s_i:e_i}) &\leq \tilde{\mathcal{O}}\left(D\sqrt{K^2 + K\sum_{t=s}^{e} \|\boldsymbol{g}_t\|_\star^2}\right) \\
&\leq \tilde{\mathcal{O}}\left(D\sqrt{\left(\frac{C_I + D}{D}\right)^2 + \frac{C_I + D}{D}\sum_{t=s}^{e} \|\boldsymbol{g}_t\|_\star^2}\right) \\
&= \tilde{\mathcal{O}}\left(\sqrt{(C_I + D)^2 + (C_I + D)\sum_{t=s}^{e} \|\boldsymbol{g}_t\|_\star^2}\right) \\
&\leq \tilde{\mathcal{O}}\left(C_I + D + \sqrt{(C_I + D)\sum_{t=s}^{e} \|\boldsymbol{g}_t\|_\star^2}\right) \ ,
\end{aligned}$$

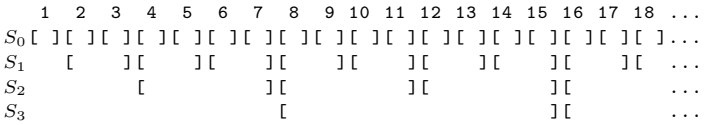


FIGURE 5.1: Geometric covering intervals. Each interval is denoted by `[ ]`. Figure taken from Jun et al. [2017].

where in the second inequality we used the assumption that $K \leq \frac{C_I+D}{D}$, while in the last step we used the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. To conlude, it suffices to notice that $C_I \leq D|I|$, in order to obtain that $C_I \leq \sqrt{DC_I|I|}$. □

From the assumption in Equation (5.21) in the theorem above, we can see that the crucial condition in order to have a bound of $\mathcal{O}(\sqrt{T(1+C_T)})$ for any possible sequence of comparators $\boldsymbol{u}_{1:T}$ is to have for any possible subinterval $I \subseteq [1,T]$ a bound of $\mathcal{O}((C_I+1)\sqrt{|I|})$. We have seen that bounds of this type are automatically satisfied by OMD-style algorithms when a Lipschitz condition is assumed on the Bregman divergence at hand. On the other hand, we are not aware of dynamic regret bounds for FTRL-style algorithms. Therefore, we speculate the same theorem above could be applied to any regular strongly-adaptive algorithm as described in the previous section, where the base algorithm $\mathcal{A}$ used as blackbox is based on OMD and satisfy a bound as the one in Equation (5.21).

Temporal variability. In Zhang et al. [2018b] it is shown that dynamic regret of the strongly adaptive algorithm given in Jun et al. [2017] is order of $\tilde{\mathcal{O}}(\max\{\sqrt{T}, T^{2/3}(V_T+1)^{1/3}\})$. We next report a simplified version of the proof of this result.

Theorem 5.11 ([Zhang et al., 2018b])**.** *Assume $\mathcal{A}$ is a strongly adaptive algorithm with $R_I = \mathcal{O}(\sqrt{|I|\ln T})$ for any interval I. Then, for any sequence of comparators $\boldsymbol{u}_{1:T}$ and loss function $\ell_1, \dots, \ell_t$, $\mathcal{A}$ incurs the following regret*

$$R(\boldsymbol{u}_1, \dots, \boldsymbol{u}_T) = \mathcal{O}\left(T^{2/3}(V_T \ln T)^{1/3}\right) .$$

Proof. Consider any partition of the time horizon into K subintervals $I_1, \dots, I_K$, where $I_i = [s_i, e_i]$. Also, for any subinterval let V_{I_k} be the temporal variability over the subinterval I_k, i.e,

$$V_{I_k} = \sum_{t=s_k+1}^{e_k} \max_{\boldsymbol{x} \in V} |\ell_t(\boldsymbol{x}) - \ell_{t-1}(\boldsymbol{x})| .$$

Let $\boldsymbol{u}_t^* = \operatorname{argmin}_{\boldsymbol{x} \in V} \ell_t(\boldsymbol{x})$. For any $k \in [K]$, we have that

$$\begin{aligned}\sum_{t=s_k}^{e_k} (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u}_t)) &\leq \sum_{t=s_k}^{e_k} (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u}_{s_k}^*)) + \sum_{t=s_k}^{e_k} (\ell_t(\boldsymbol{u}_{s_k}^*) - \ell_t(\boldsymbol{u}_t)) \\ &= \mathcal{O}(\sqrt{|I_k|\ln T}) + \sum_{t=s_k}^{e_k} (\ell_t(\boldsymbol{u}_{s_k}^*) - \ell_t(\boldsymbol{u}_t)) , \end{aligned} \tag{5.23}$$

where we have used the guarantee of the strongly adaptive algorithm.

Now, note that

$$\begin{aligned}\ell_t(\boldsymbol{u}_{s_k}^*) - \ell_t(\boldsymbol{u}_t) &\leq \ell_t(\boldsymbol{u}_{s_k}^*) - \ell_{s_k}(\boldsymbol{u}_{s_k}^*) + \ell_{s_k}(\boldsymbol{u}_t) - \ell_t(\boldsymbol{u}_t) \\ &= \sum_{\tau=s_k+1}^{e_k} (\ell_\tau(\boldsymbol{u}_{s_k}^*) - \ell_{\tau-1}(\boldsymbol{u}_{s_k}^*)) + \sum_{\tau=s_k+1}^{e_k} (\ell_{\tau-1}(\boldsymbol{u}_t) - \ell_\tau(\boldsymbol{u}_t)) \\ &\leq 2 \sum_{\tau=s_k+1}^{e_k} \max_{\boldsymbol{x} \in V} |\ell_\tau(\boldsymbol{x}) - \ell_{\tau-1}(\boldsymbol{x})| \end{aligned}$$

$$\leq 2V_{I_k} \ , \tag{5.24}$$

where the first inequality derives from the fact that $\ell_{s_k}(\boldsymbol{u}^*_{s_k}) \leq \ell_{s_k}(\boldsymbol{u}_t)$. Therefore, putting together Equation (5.23) and Equation (5.24) we have that for any sequence of comparators $\boldsymbol{u}_{1:T}$

$$\begin{aligned}\sum_{t=1}^{T}(\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u}_t)) &\leq \sum_{k=1}^{K}\mathcal{O}(\sqrt{|I_k|\ln T}) + 2\sum_{k=1}^{K}V_{I_k}|I_k| \\ &\leq \mathcal{O}(\sqrt{TK\ln T}) + 2\max_k |I_k|\sum_{k=1}^{K}V_{I_k} \\ &\leq \mathcal{O}(\sqrt{TK\ln T}) + 2\max_k |I_k| V_T \ .\end{aligned}$$

Given a fixed M, the partition which minimizes $\max_k |I_k|$ is the one dividing the time horizon in intervals of possibly equal length. Indeed, in this way we would have $\max_k |I_k| = \mathcal{O}(T/K)$ and the last bound becomes

$$\sum_{t=1}^{T}(\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{u}_t)) \leq \mathcal{O}\left(\sqrt{TK\ln T} + 2\frac{T}{K}V_T\right) \ .$$

Setting $M = \lfloor (T/\ln\ln T)^{1/3} V_T^{2/3} \rfloor$ yields the desired result. □

To summarize, using existing strongly-adaptive algorithms, from Theorem 5.10 and Theorem 5.11 the following dynamic regret bound can be achieved

$$\tilde{\mathcal{O}}\left(\min\left\{\sqrt{T(1+C_T)}, T^{2/3}(V_T+1)^{1/3}\right\}\right) \ , \tag{5.25}$$

for any sequence $\boldsymbol{u}_{1:T}$ without knowing its path-length in advance.

However, given our lower bound of $\Omega(V_T)$ in Theorem 5.2, this bound is suboptimal.

In the next section, we show instead how one can achieve an improved bound of

$$\mathcal{O}(\min\{\sqrt{T(1+C_T)}, V_T\} \ , \tag{5.26}$$

for any sequence of comparators $\boldsymbol{u}_{1:T}$ by combining existing algorithms.

Finally, we would like to point out that better bounds can be achieved making additional assumptions. For example, improved rates for the dynamic regret were shown in Mokhtari et al. [2016] in the case of strongly-convex functions and in Zhang et al. [2017] when it is possible to query the function multiple times per round.

5.5 Adapting to different path-lengths

Our approach to achieve the improved bound given Equation (5.26) for any given sequence of comparators is to combine different existing algorithms. In particular, using either the parameter-free algorithm given in Cutkosky [2020] or the one from Zhang et al. [2018a], we can achieve a bound of $\mathcal{O}(\sqrt{T(1+C_T)})$ for all possible sequences $\boldsymbol{u}_{1:T}$. In order to achieve a bound on the temporal variability of $\mathcal{O}(V_T)$, we can simply use the greedy algorithm given in Algorithm 7, which in every step plays the minimizer of the last seen loss function. Finally, to combine these two different algorithms and get the best result, we use a modification of the *ML-Prod* algorithm [Gaillard et al., 2014] proposed in Sani et al. [2014] and depicted in Algorithm 11. This algorithm takes as input two base learners (aka experts) $\mathcal{A}$ and $\mathcal{B}$ and guarantees a regret which is (almost) constant against $\mathcal{B}$ and $\mathcal{O}(\sqrt{T}\ln\ln T)$ against $\mathcal{A}$ in the worst case. The next theorem provides a regret bound to Algorithm 11.

Theorem 5.12 (Adapted from [Gaillard et al., 2014, Theorem 1])**.** *Consider a sequence of loss functions $\boldsymbol{g}_1, \dots, \boldsymbol{g}_T$, with $\boldsymbol{g}_t \in [0,1]^2$ for all t. Then, for all non-increasing sequences of*

Algorithm 11 Anytime $(\mathcal{A}, \mathcal{B})$-PROD

Require: Algorithms $\mathcal{A}, \mathcal{B}$, $\eta_1 = 1/2$, $w_{1,\mathcal{A}} = w_{1,\mathcal{B}} = 1/2$.
1: **for** $t = 1, \dots, T$ **do**
2: Let $s_t = \frac{\eta_t w_{t,\mathcal{A}}}{\eta_t w_{t,\mathcal{A}} + w_{t,\mathcal{B}}/2}$
3: Get $\boldsymbol{a}_t$ from $\mathcal{A}$ and $\boldsymbol{b}_t$ from $\mathcal{B}$ and predict $\boldsymbol{x}_t = s_t \boldsymbol{a}_t + (1 - s_t)\boldsymbol{b}_t$
4: Receive $\ell_t : \mathbb{R}^d \to [0, 1]$ and pay $\ell_t(\boldsymbol{x}_t)$
5: Feed ℓ_t to $\mathcal{A}$ and $\mathcal{B}$
6: Set $r_{t,\mathcal{B}} = 0$, $r_{t,\mathcal{A}} = \ell_t(\boldsymbol{b}_t) - \ell_t(\boldsymbol{a}_t)$
7: Set $\eta_{t+1,\mathcal{A}} = \sqrt{(1 + \sum_{i=1}^{t} (\ell_i(\boldsymbol{b}_i) - \ell_i(\boldsymbol{a}_i))^2)^{-1}}$, $\eta_{t+1,\mathcal{B}} = \eta_{t,\mathcal{B}}$
8: Set $w_{t+1,i} = w_{t,i}(1 + \eta_t r_{t,i})^{\eta_{t+1,i}/\eta_{t,i}}$, for $i \in \{\mathcal{A}, \mathcal{B}\}$
9: **end for**

learning rates $(\eta_{t,i})_{t=1}^T$*, with* $\eta_{t,i} \le 1/2$ *for* $i \in \{\mathcal{A}, \mathcal{B}\}$ *and* $t \in [T]$*, Algorithm 11 ensures*

$$R_T(\boldsymbol{e}_i) \le \frac{\ln 2}{\eta_{1,i}} + \sum_{t=1}^{T} \eta_{t,i} r_{t,i}^2 + \frac{1}{\eta_{T+1,i}} \ln \left(1 + \frac{1}{e} \sum_{t=1}^{T} \sum_{j=1}^{2} \left(\frac{\eta_{t,j}}{\eta_{t+1,j}} \right) - 1 \right) . \tag{5.27}$$

In our proposed algorithm, we set the strongly-adaptive algorithm from Cutkosky [2020] as algorithm $\mathcal{A}$, and the greedy algorithm in Algorithm 7 as $\mathcal{B}$.

In the next theorem, we provide a dynamic regret bound on the resulting algorithm.

Theorem 5.13. *Let* $V \subset \mathbb{R}^d$ *be a closed convex set. Furthermore, assume its diameter is bounded by* D*. Let* $\ell_1, \dots, \ell_t$ *be a sequence of loss functions such that* $\ell_t(\boldsymbol{x}) \in [0, 1]$*, for all* $\boldsymbol{x} \in V$*. Let* $\boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)$ *and assume that* $\|\boldsymbol{g}_t\|_\star \le L$ *for all* t*. Then, running Algorithm 11 with* $\mathcal{A}$ *as the strongly-adaptive algorithm from Cutkosky [2020], and* $\mathcal{B}$ *as Algorithm 7, guarantees*

$$R_T(\boldsymbol{u}_{1:T}) = \tilde{\mathcal{O}}(\min(V_T, \sqrt{TD(C_T + D)} + \sqrt{T})),$$

where $\tilde{O}$ *hides poly-logarithmic terms in* T*.*

Proof. In the following, let $\ell_{t,\mathcal{A}} = \ell_t(\boldsymbol{a}_t)$ and $\ell_{t,\mathcal{B}} = \ell_t(\boldsymbol{b}_t)$. From Theorem 5.12, we have that the loss of Algorithm 11 compared to algorithm $\mathcal{B}$ is bounded as follows

$$\begin{aligned} \sum_{t=1}^{T} (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{a}_t)) &\le \sum_{t=1}^{T} \sum_{i \in \{\mathcal{A}, \mathcal{B}\}} p_{t,i} \ell_{t,i} - \sum_{t=1}^{T} \ell_t(\boldsymbol{a}_t) \\ &\le 2 \ln 2 + 2 \ln \left(1 + \frac{1}{e} \sum_{t=1}^{T} \frac{\eta_t}{\eta_{t+1}} - 1 \right) , \end{aligned}$$

where the first inequality follows from Jensen's inequality, while the second choosing $i = \mathcal{B}$ in Eq. (5.27). Following Gaillard et al. [2014], it can be shown that the last term above is order of $\ln \ln T$. Therefore, given a sequence $\boldsymbol{u}_{1:T}$ adding and subtracting $\sum_{t=1}^{T} \boldsymbol{u}_t$ on both sides above, and taking $\sum_{t=1}^{T} \ell_t(\boldsymbol{a}_t)$ on the right side we get

$$\begin{aligned} R_T(\boldsymbol{u}_{1:T}) &\le R_{T,\mathcal{B}}(\boldsymbol{u}_{1:T}) + 2 \ln 2 + 2K'_T \\ &\le V_T + \ell_1(\boldsymbol{x}_1) - \ell_T(\boldsymbol{x}_{T+1}) + 2K_T \\ &= \tilde{\mathcal{O}}(V_T), \end{aligned} \tag{5.28}$$

where we have used the regret guarantee of algorithm $\mathcal{B}$ in the second inequality, and defined $K'_T = \ln(1 + \frac{1}{e} \ln(T + 1))$ and $K_T = \ln 2 + K'_T$.

On the other hand, for algorithm $\mathcal{A}$, applying again Eq. (5.27) with $i = \mathcal{A}$ we have that

$$\sum_{t=1}^{T} (\ell_t(\boldsymbol{x}_t) - \ell_t(\boldsymbol{b}_t)) \le \sum_{t=1}^{T} \sum_{i \in \{\mathcal{A}, \mathcal{B}\}} p_{t,i} \ell_{t,i} - \sum_{t=1}^{T} \ell_t(\boldsymbol{b}_t)$$

$$
\begin{aligned}
&\leq 2\ln 2 + \sum_{t=1}^{T} \eta_{t,\mathcal{A}}(\ell_t(\boldsymbol{b}_t) - \ell_t(\boldsymbol{a}_t))^2 + \frac{1}{\eta_{T+1,\mathcal{A}}} 2\ln\left(1 + \frac{1}{e}\sum_{t=1}^{T}\frac{\eta_t}{\eta_{t+1}} - 1\right) \\
&\leq (2 + K_T)\sqrt{1 + \sum_{t=1}^{T}(\ell_t(\boldsymbol{b}_t) - \ell_t(\boldsymbol{a}_t))^2} + K_T + 4 \\
&\leq (2 + K_T)\sqrt{T+1} + K_T + 4,
\end{aligned}
$$

where the second-to-last inequality derives from applying Corollary 4 in Gaillard et al. [2014], while in the last step we used the fact that the losses are bounded in $[0, 1]$. Adding and subtracting again $\sum_{t=1}^{T} \boldsymbol{u}_t$ on both sides, and taking $\sum_{t=1}^{T} \ell_t(\boldsymbol{b}_t)$ on the right hand side we get

$$
\begin{aligned}
R_T(\boldsymbol{u}_{1:T}) &\leq R_{T,\mathcal{A}}(\boldsymbol{u}_{1:T}) + c(2 + K_T)\sqrt{T} + K_T + 4 \\
&= \tilde{\mathcal{O}}(\sqrt{TD(C_T + D)} + \sqrt{T}) \ , \qquad (5.29)
\end{aligned}
$$

where we have used the regret guarantee of algorithm $\mathcal{A}$ in the last step.

Putting together Equation (5.28) and Equation (5.29) we get the result stated in Theorem 5.13. □

In the worst case, assuming a high temporal variability and that the loss of algorithm $\mathcal{A}$ is linear in T, we have that the regret of our combiner algorithm is order of $\tilde{\mathcal{O}}(\sqrt{TD(C_T + D)} + \sqrt{T})$.

5.6 Discussion

In this chapter, we have shown how existing bounds in the dynamic setting with full information feedback can be improved. We established new lower bounds on the dynamic regret in terms of temporal variability and we showed algorithms with matching upper bounds.

In particular, using implicit updates we designed an algorithm that can adapt to both the temporal variability and the path-length of the sequence of comparators. Furthermore, when the desired path-length is not fixed in advance and cannot be observed on the fly, we showed how to combine existing algorithms in order to achieve an optimal bound. Despite the appealing regret bound achieved in the latter setting, the resulting algorithm might not be practical in a realistic scenario due to the complexity of running 3 algorithms in parallel. Furthermore, as observed in previous work, all strongly-adaptive algorithms [Cutkosky, 2020; Jun et al., 2017] have a running time of $\mathcal{O}(T \ln T)$ and it is currently an open question if it can be improved. Future research directions therefore could aim at designing faster and more practical algorithms which can adapt to unknown path-lengths, or in alternative prove that this goal cannot be achieved.

Chapter 6

Learning with Expert Advice

The *Learning with Expert Advice* setting is one of the most widely known and studied problems in online learning. In this setting, given a time horizon T a learning agent has to output a prediction in every round on the problem at hand, following the advice of d different "experts". These experts can be physical entities or even algorithms themselves. After a choice is made, the losses associated to each expert are revealed and the agent pays the loss associated to its choice. Formally, let Δ_d be the d-dimensional simplex, i.e., $\Delta_d = \{\boldsymbol{x} \in \mathbb{R}^d_+ : \|\boldsymbol{x}\|_1 = 1\}$ and $\boldsymbol{x}_t$ be the distribution over the d experts of the learning agent at time t. We assume that in every round, the decision maker can play any $\boldsymbol{x} \in \Delta_d$. Note that naturally every expert is associated with a corner of the simplex, i.e., expert i is represented by $\boldsymbol{e}_i$. If a single expert has to be selected in every round by the learning strategy, then randomization is necessary, since the set $V = \{\boldsymbol{e}_i\}_{i=1}^d$ is not convex (see also Shalev-Shwartz [2012, Section 2.1]). The loss function is linear, i.e. $\ell_t(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{g}_t \rangle$, and $\langle \boldsymbol{e}_i, \boldsymbol{g}_t \rangle = g_{t,i}$ represents the loss of expert i at time t. As usual, given any $\boldsymbol{u} \in \Delta_d$, the goal is to minimize the regret against $\boldsymbol{u}$,

$$R_T(\boldsymbol{u}) = \sum_{t=1}^{T} \langle \boldsymbol{x}_t - \boldsymbol{u}, \boldsymbol{g}_t \rangle \ .$$

This problem has a long history tracking back to Cesa-Bianchi et al. [1997] and various algorithms have been proposed since then. While existing lower bounds settle the worst-case regret to $\Omega(\sqrt{T \ln d})$ [Orabona and Pál, 2015], recently there has been a surge of interest regarding data dependent bounds improving the worst-case regret bound in more favourable scenarios. In general, there are two directions that can be exploited to improve the worst-case bound. One way is to improve the dependence on T using quantities such as the loss of the best expert, denoted $L^\star$, which could potentially be 0. These bounds are also called *first-order* bounds. Another way is to improve the dependence on the number of experts. These bounds are called *quantile bounds*. We will describe these concepts next.

Quantile bounds. When there are multiple good experts, one can improve the dependence on $\ln d$ to $\ln(1/\varepsilon)$, where ε is the quantile representing the εd fraction of good experts over the total Chaudhuri et al. [2009]. Bounds of this type are called *quantile* bounds. More in general, one could aim at regret bounds involving the KL divergence between the competitor $\boldsymbol{u}$ and any prior distribution over the experts $\boldsymbol{x}_1$. Such a bound implies a quantile bound. This line of work culminated in parameter-free algorithms (which do not require the knowledge of parameters to be tuned in advance in terms of unknown quantities) such as the ones in Koolen and van Erven [2015] and Orabona and Pal [2016], whose regret bound is order of

$$R_T(\boldsymbol{u}) = \mathcal{O}(\sqrt{T \, \mathrm{KL}(\boldsymbol{u}, \boldsymbol{x}_1)}) \ . \tag{6.1}$$

To see why a bound involving the KL divergence implies a quantile bound, assume that $\boldsymbol{u}$ is concentrated on a fraction ε of the experts, i.e., $\boldsymbol{u} = (1/\varepsilon d, \dots, 1/\varepsilon d, 0, \dots, 0)$. Then, letting $\boldsymbol{x}_1$ to be the uniform distribution, from Equation (6.1) we have that

$$\mathrm{KL}\left(\boldsymbol{u}; \boldsymbol{x}_1\right) = \sum_{i=1}^{d} u_i \ln \frac{u_i}{x_i} = \sum_{i=1}^{j^*} \frac{1}{\varepsilon d} \ln \frac{1}{\varepsilon d} - \ln \frac{1}{d} \sum_{i=1}^{d} u_i = \ln \frac{1}{\varepsilon d} - \ln \frac{1}{d} = \ln(1/\varepsilon) \ .$$

In this case a bound of $\mathcal{O}(\sqrt{T \ln d})$ might be too pessimistic, especially if the number of experts is high.

Second order bounds. While first-order bounds are appealing, one can try to go even further and aim for a regret bound of the form $\mathcal{O}(\sqrt{V_T})$, where V_T is a second-order quantity referred to as the (cumulative) "variance" over the losses. In the literature, mainly two kinds of variance have been considered. The first, that we denote by $V_T^{\boldsymbol{u}}$, has also been called variance *over time* [Freund, 2016],

$$V_T^{\boldsymbol{u}} = \sum_{i=1}^{d} u_i V_{T,i} = \sum_{i=1}^{d} u_i \sum_{t=1}^{T} (\langle \boldsymbol{x}_t, \boldsymbol{g}_t \rangle - g_{t,i})^2 \ . \tag{6.2}$$

The *Squint* algorithm has a regret bound of $\mathcal{O}(\sqrt{V_T^{\boldsymbol{u}} \operatorname{KL}(\boldsymbol{u}; \boldsymbol{x}_1)})$ [Koolen and Van Erven, 2015].

In contrast, in Cesa-Bianchi et al. [2005] another notion of variance is given, which is defined as variance *over actions* in Freund [2016]: let $\boldsymbol{x}_t$ be the distribution over the d experts of the learning agent at time t, then define V_T as follows

$$V_T = \sum_{t=1}^{T} v_t = \sum_{t=1}^{T} \sum_{i=1}^{d} x_{t,i} (g_{t,i} - \langle \boldsymbol{x}_t, \boldsymbol{g}_t \rangle)^2 \tag{6.3}$$

The algorithm provided in Cesa-Bianchi et al. [2005] has a regret bound order of $\mathcal{O}(\sqrt{V_T \ln d})$.

Open problem. In a description of different types of variance, Freund [2016] argues that the variance over actions should be preferred over the one over time. Moreover, while there exist algorithms which have regret bounds in terms of variance over actions (see Cesa-Bianchi et al. [2005]; de Rooij et al. [2014]), or variance over time and quantiles (Koolen and Van Erven [2015]), no existing algorithms combine variance over actions and quantiles and reach a regret bound of the type

$$R_T(\boldsymbol{u}) = \mathcal{O}\left(\sqrt{V_T \ln(1/\varepsilon)}\right) \ .$$

The question about whether such an algorithm exists has been left open since Freund [2016].

Even if the open problem described in Freund [2016] is restricted to the setting where $g_{t,i} \in [0,1]$ for all $i \in [d]$ and all $t \in [T]$, a regret bound involving V_T as defined in Equation (6.3) would imply a *scale-free* algorithm. This means that it does not have to know in advance the range of the losses.

Outline. In this chapter, we start by describing the well known exponential weights algorithm: we first analyze how it behaves in the worst case by giving a regret bound of $\mathcal{O}(\sqrt{T})$ (zeroth-order bound). Then, we proceed to show how this version can be improved in order to obtain a bound of $\mathcal{O}(\sqrt{L^*})$ (first-order bound). Next, we describe an algorithm achieving a second-order bound in V_T, as defined in Equation (6.3). Finally, we show how one can combine different algorithms in order to tune to an unknown quantity, which in this case is the quantile ε. In particular, we show how to use a recent technique proposed in Bhaskara et al. [2020] and reach a similar bound to the one asked in [Freund, 2016]. The new results contained in the end of this chapter (i.e., Section 6.3) are part of ongoing work and have not been published yet.

6.1 Worst-case regret bounds and beyond

The first idea for designing an algorithm in the LEA setting would be to use OGD, as illustrated in Example 3.1. In this way, we have

$$\begin{aligned} \tilde{\boldsymbol{x}}_{t+1} &= \boldsymbol{x}_t - \eta_t \boldsymbol{g}_t \\ \boldsymbol{x}_{t+1} &= \Pi_{\Delta_d}(\tilde{\boldsymbol{x}}_{t+1}) \ . \end{aligned}$$

Algorithm 12 Exponential Weights Update algorithm (Hedge)

Require: A sequence of non-increasing learning rates $(\eta_t)_{t=0}^{T-1}$
$\boldsymbol{x}_1 = [1/d, \dots, 1/d] \in \Delta_d$
$\boldsymbol{\theta}_1 = \mathbf{0}$
for $t = 1, \dots, T$ **do**
  Output $\boldsymbol{x}_t$
  Receive $\boldsymbol{g}_t \in \mathbb{R}^d$ and pay $\langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle$
  Update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{g}_t$
  Update $x_{t+1,j} \propto \exp\left(\eta_{t-1}\theta_{t+1,j}\right),\ j \in [d]$
end for

However, contrarily to Example 3.1, in this case we should operate a Euclidean projection on the simplex, i.e., solve the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^d} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_{t+1}\|_2^2 \\ \text{s.t. } \|\boldsymbol{x}\|_1 = 1 \\ \boldsymbol{x} \geq 0 \ . \end{aligned}$$

This problem has no closed-form solution, however there exist efficient algorithms which can find a solution in $\mathcal{O}(d \ln d)$ running time [Shalev-Shwartz and Singer, 2006]. By setting $\boldsymbol{x}_1 = [1/d, \dots, 1/d]$ and applying Theorem 3.3 we have for any $\boldsymbol{u} \in \Delta_d$

$$R_T(\boldsymbol{u}) \leq \frac{D^2}{\eta_T} + \frac{1}{2}\sum_{t=1}^{T} \eta_t \|\boldsymbol{g}_t\|_2^2 \ ,$$

where $D^2 = \max_{\boldsymbol{x}, \boldsymbol{u} \in \Delta_d} B_\psi(\boldsymbol{u}, \boldsymbol{x})$.

Now, observe that for any $\boldsymbol{u}, \boldsymbol{x} \in \Delta_d$ we have

$$B_\psi(\boldsymbol{u}, \boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{x}\|_2^2 \leq \frac{1}{2}\|\boldsymbol{u}\|_2^2 + \frac{1}{2}\|\boldsymbol{x}\|_2^2 \leq \frac{1}{2}\|\boldsymbol{u}\|_1 + \frac{1}{2}\|\boldsymbol{x}\|_1 = 1 \ ,$$

where we have used the triangle inequality first and then the fact that $\|\boldsymbol{y}\|_2^2 \leq \|\boldsymbol{y}\|_1$ for any $\boldsymbol{y} \in \Delta_d$. Assuming the losses to be bounded, we set the learning rate η_t to get a regret bound of $\mathcal{O}(\sqrt{T})$, as shown in the next corollary.

Corollary 6.1. *Consider the setting of LEA (Learning with Expert Advice), and assume that $\|\boldsymbol{g}\|_\infty \leq L$. Let $\boldsymbol{x}_1$ be the uniform distribution over Δ_d. Then, running OGD (see Example 3.1) with $\eta_t = \sqrt{1/dL^2t}$ yields the following regret bound for any $\boldsymbol{u} \in \Delta_d$*

$$R_T(\boldsymbol{u}) \leq 2L\sqrt{Td} \ .$$

Proof. Note that $\|\boldsymbol{g}_t\|_\infty \leq L$ implies $\|\boldsymbol{g}_t\|_2^2 \leq dL^2$. Therefore, from Theorem 3.3 we have that

$$R_T(\boldsymbol{u}) \leq L\sqrt{Td} + \frac{L\sqrt{d}}{2}\sum_{t=1}^{t} \frac{1}{\sqrt{t}} \leq 2L\sqrt{Td} \ ,$$

where the last step derives from using Lemma 3.5. □

Given the lower bound of $\Omega(\sqrt{T \ln d})$, this upper bound is not tight. Next, we explore the possibility of using a different function $\psi(\boldsymbol{x})$ in order to improve the dependence on the dimension d.

6.1.1 Exponential weights

As we have seen in Chapter 3, a particularly useful function when dealing with the simplex is the negative entropy, i.e., $\psi(\boldsymbol{x}) = \sum_{i=1}^{d} x_i \ln x_i$, which gives rise to the KL-divergence and is 1-strongly convex w.r.t. $\|\cdot\|_1$, as shown in Theorem 2.19.

One crucial point for obtaining improved (first or second order) regret bounds is to have time-varying learning rates. We also have seen that OMD with time-varying learning rates requires a bounded domain with respect to the maximum Bregman divergence between any two points on the domain of interest, which in this case is the simplex. Unfortunately, the maximum KL-divergence between any two distributions on the simplex can be unbounded.

On the other hand, FTRL does not require the assumption of bounded domain with respect to ψ. Therefore, using the update obtained in Example 3.3 (which can be computed in closed form and requires $\mathcal{O}(d)$ operations per round), we run the exponential weights update algorithm with time-varying learning rates (depicted in Algorithm 12). Moreover, by using the generic analysis of the FTRL algorithm given the Chapter 3, we immediately provide a regret bound to the algorithm, as shown in the next corollary.

Corollary 6.2. *Consider the setting of LEA, and assume that $\|\boldsymbol{g}\|_\infty \leq L$. Let $\boldsymbol{x}_1$ be the uniform distribution over Δ_d. Then, for any $\boldsymbol{u} \in \Delta_t$ the Exponential Weights algorithm (Algorithm 12) with $\eta_{t-1} = L\sqrt{\ln d / t}$ incurs regret bounded as*

$$R_T(\boldsymbol{u}) \leq 2L\sqrt{T \ln d} \,. \tag{6.4}$$

Proof. Using Theorem 3.11 and the learning rate stated in the assumptions, the regret can be bounded as follows

$$\begin{aligned} R_T(\boldsymbol{u}) &\leq \frac{\psi(\boldsymbol{u}) - \psi(\boldsymbol{x}_1)}{\eta_{T-1}} + \frac{1}{2}\sum_{t=1}^{T} \eta_{t-1} \|\boldsymbol{g}_t\|_\star^2 \\ &\leq \frac{1}{\eta_{T-1}} \sum_{i=1}^{d} x_{1,i} \ln \frac{1}{x_{1,i}} + \frac{L^2}{2} \sum_{t=1}^{T} \eta_{t-1} \\ &\leq 2L\sqrt{T \ln d} \,, \end{aligned}$$

where in the last step we used the fact that the negative entropy is minimized by the uniform distribution on the simplex with value $\ln d$, and again Lemma 3.5 to bound the sum over time. □

In this case, the bound is tight. However, as explained in the beginning of this chapter, we can do better if the best expert suffers a low loss in every round. We will explore this possibility in the next section.

6.1.2 First-Order bounds

In this section, we are going to show how to improve on the worst-case $\mathcal{O}\sqrt{T}$ regret bound using a different learning rate. The final bound will depend on the cumulative loss L^* of the competitor. This result first appeared in Auer et al. [2002], which probably designed one of the very first adaptive algorithms that later inspired a whole line of work. However, rather than reporting the analysis in Auer et al. [2002], we follow a different path. The crucial step is to use the so called "local norms", which depend on the Hessian of the regularizer ψ.

Theorem 6.3. *Consider the setting of LEA, and assume $g_{t,i} \in [0,1]$ for all $i = 1, \dots, d$ and all $t \in [T]$. Then, for all $\boldsymbol{u} \in \Delta_d$ the Exponential Weights algorithm (Algorithm 12) with $\eta_{t-1} = \sqrt{\ln d / (1 + \sum_{i=1}^{t-1} \ell_i(\boldsymbol{x}_i))}$ incurs regret bounded as*

$$R_T(\boldsymbol{u}) \leq 2\sqrt{\ln d \sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{u} \rangle} + 4 \ln d + 2\sqrt{\ln d} \,. \tag{6.5}$$

Proof. Assuming that $x_t \in \operatorname{int} V$, by using [Lattimore and Szepesvári, 2020, Exercise 28.12], we rewrite the FTRL regret bound as follows

$$R_T(\boldsymbol{u}) \leq \frac{\psi(\boldsymbol{u}) - \psi(\boldsymbol{x}_1)}{\eta_{T-1}} + \sum_{t=1}^{T} \left[\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1} \rangle - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta_{t-1}} \right] .$$

Also, by the Taylor's theorem (see, e.g., Lattimore and Szepesvári [2020, Theorem 26.12], we have that

$$B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) = \psi(\boldsymbol{x}_{t+1}) - \psi(\boldsymbol{x}_t) - \langle \nabla\psi(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} - \boldsymbol{x}_t\rangle = \frac{1}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2_{\nabla^2\psi(\boldsymbol{z}_t)} \ ,$$

where $\boldsymbol{z}_t = \alpha_t\boldsymbol{x}_t + (1-\alpha_t)\boldsymbol{x}_{t+1}$ with $\alpha_t \in [0,1]$ is a point on the segment connecting $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$. For notational convenience, let $H_t = \nabla^2\psi(\boldsymbol{z}_t)$. Note that with negative entropy regularization, $H_t = \operatorname{diag}(1/z_{t,1}, \dots, 1/z_{t,d})$.

Next, by Holder's inequality, we have that

$$\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1}\rangle \le \|\boldsymbol{g}_t\|_{H_t^{-1}}\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\|_{H_t} \ ,$$

since the dual norm of $\|\cdot\|_A$ for a positive definite matrix A is $\|\cdot\|_{A^{-1}}$. Therefore,

$$\begin{aligned}
\underbrace{\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1}\rangle - \frac{B_\psi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)}{\eta_{t-1}}}_{(\star)} &\le \|\boldsymbol{g}_t\|_{H_t^{-1}}\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\|_{H_t} - \frac{1}{2\eta_{t-1}}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2_{H_t} \\
&\le \frac{\eta_{t-1}}{2}\|\boldsymbol{g}_t\|^2_{H_t^{-1}} \\
&= \frac{\eta_{t-1}}{2}\sum_{i=1}^d z_{t,i}g_{t,i}^2 \\
&\le \frac{\eta_{t-1}}{2}\sum_{i=1}^d z_{t,i}g_{t,i} \ ,
\end{aligned}$$

where the second inequality derives from the fact that $bx - ax^2/2 \le \frac{b^2}{2a}$ for all $x \in \mathbb{R}$, while in the last step we used the fact that $g_{t,i} \in [0,1]$.

Now, observe that if $\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}_{t+1}\rangle < 0$, then the term $(\star)$ above is less than 0 (since the Bregman is always non-negative). Therefore we assume $\langle \boldsymbol{g}_t, \boldsymbol{x}_{t+1}\rangle \le \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle$ and get

$$\sum_{i=1}^d z_{t,i}g_{t,i} = \alpha_t\sum_{i=1}^d x_{t,i}g_{t,i} + (1-\alpha_t)\sum_{i=1}^d x_{t+1,i}g_{t,i} \le \sum_{i=1}^d x_{t,i}g_{t,i} \ .$$

Finally, by using the learning rate $\eta_t = \sqrt{\frac{\ln d}{1+\sum_{i=1}^t \langle \boldsymbol{x}_t, \boldsymbol{g}_t\rangle}}$, we get

$$\begin{aligned}
R_T(\boldsymbol{u}) = \sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u}\rangle &\le \frac{\ln d}{\eta_{T-1}} + \sum_{t=1}^T \frac{\eta_{t-1}}{2}\langle \boldsymbol{x}_t, \boldsymbol{g}_t\rangle \\
&\le \frac{\ln d}{\eta_T} + \frac{\sqrt{\ln d}}{2}\sum_{t=1}^T \frac{\ell_t(\boldsymbol{x}_t)}{\sqrt{1+\sum_{i=1}^{t-1}\ell_i(\boldsymbol{x}_t)}} \\
&\le \sqrt{\ln d\left(1+\sum_{t=1}^T \ell_t(\boldsymbol{x}_t)\right)} + \frac{\sqrt{\ln d}}{2}\sum_{t=1}^T \frac{\ell_t(\boldsymbol{x}_t)}{\sqrt{\sum_{i=1}^{t}\ell_i(\boldsymbol{x}_t)}} \\
&\le \sqrt{\ln d\left(1+\sum_{t=1}^T \ell_t(\boldsymbol{x}_t)\right)} + \frac{1}{2}\sqrt{\ln d\sum_{t=1}^T \ell_t(\boldsymbol{x}_t)} \\
&\le 2\sqrt{\ln d\left(1+\sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle\right)} \\
&\le 2\sqrt{\ln d} + 2\sqrt{\ln d\sum_{t=1}^T \langle \boldsymbol{x}_t, \boldsymbol{g}_t\rangle} \ ,
\end{aligned}$$

where the second inequality is due to the fact that η_t is non-increasing, the third since the

Algorithm 13 Adahedge

Require: $\alpha > 0$, $\boldsymbol{x}_1 \in \Delta_d$
$\lambda_1 = 0$
$\boldsymbol{\theta}_1 = \mathbf{0}$
for $t = 1, \dots, T$ **do**
  Output $\boldsymbol{x}_t$
  Receive $\boldsymbol{g}_t \in \mathbb{R}^d$ and pay $\langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle$
  Update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{g}_t$
  Set $\delta_t = \begin{cases} \langle \boldsymbol{g}_1, \boldsymbol{x}_1 \rangle, & t = 1 \\ \lambda_t \ln \left(\sum_{j=1}^d x_{t,j} \exp(-g_{t,j}/\lambda_t) \right) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle, & t \geq 2 \end{cases}$
  Update $\lambda_{t+1} = \lambda_t + \frac{1}{\alpha^2} \delta_t$
  Update $x_{t+1,j} \propto x_{1,j} \exp\left(\frac{\theta_{t+1,j}}{\lambda_{t+1}} \right)$, $j \in [d]$
end for

losses $\ell_t(\boldsymbol{x}) \in [0, 1]$ for every $\boldsymbol{x} \in V$, the fourth by using Lemma 3.5, and last step from the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, for any $a, b \in \mathbb{R}_+$.

Note that the bound above is an implicit bound, since $\sum_{t=1}^T \ell_t(\boldsymbol{x}_t)$ is appearing on both sides of the inequality. In order to get a true regret bound, we use a result from Shalev-Shwartz [2007, Lemma 19], which says that given $x, b, c \in \mathbb{R}_+$, if $x - b\sqrt{x} - c \leq 0$ holds, then $x \leq c + b^2 + b\sqrt{c}$.

Therefore, with $b = 2\sqrt{\ln d}$ and $c = \sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{u} \rangle - 2\sqrt{\ln d}$, we get

$$\sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle \leq \sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{u} \rangle + 2\sqrt{\ln d \sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{u} \rangle} + 4 \ln d + 2\sqrt{\ln d} \,,$$

which, after rearranging terms, yields the desired result. □

First-order bounds are certainly appealing, but one can go even further. Moreover, in the last result we had the assumption that the losses of the experts are restricted in a certain interval (in particular, $[0, 1]$) known to the decision maker. Next, we will explore an algorithm which does not require this assumption and reaches a refined second-order regret bound.

6.2 Variance over actions

In this section, we describe another variation of the Exponential Weights update algorithm, which gives rise to a second-order regret bound in terms of V_T, as defined in Equation (6.3) at the beginning of this chapter. The key ingredient is again a different tuning of the learning rate.

In particular, we report a regret analysis for the *Adahedge* algorithm depicted in Algorithm 13. The following results can be can be found in de Rooij et al. [2014] and [Orabona, 2019, Section 7.3]. However, compared to their analysis, we slightly modify the regularizer ψ, by using the KL divergence instead of the negative entropy, i.e., $\psi(\boldsymbol{x}) = \sum_{i=1}^d x_i \ln \frac{x_i}{x_{1,i}}$, where $\boldsymbol{x}_1$ is the model played in the first round. In order to prove a regret bound, we need a technical lemma, whose proof can be found in Appendix A.

Lemma 6.4. *Let $\boldsymbol{x}_t$ be a probability distribution on Δ_d and $\boldsymbol{g}_t \in \mathbb{R}^d$. Define $s_t \triangleq \max_{i \in [d]} g_{t,i} - \min_{i \in [d]} g_{t,i}$ and δ_t as follows,*

$$\delta_t \triangleq \begin{cases} \langle \boldsymbol{g}_1, \boldsymbol{x}_1 \rangle, & t = 1 \\ \lambda_t \ln \left(\sum_{j=1}^d x_{t,j} \exp(-g_{t,j}/\lambda_t) \right) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle, & t \geq 2 \end{cases} \tag{6.6}$$

Then, we have that $0 \leq \delta_t \leq s_t$.

We next provide a regret bound to the algorithm.

Theorem 6.5. *Consider the LEA setting. Let* $\lambda_1 = 0$, $\lambda_t = \frac{1}{\alpha^2}\sum_{i=1}^{t-1}\delta_t$, *where* $\delta_1 = \langle \boldsymbol{g}_1, \boldsymbol{x}_1\rangle$, $\delta_t = \lambda_t \ln\left(\sum_{j=1}^d x_{t,j}\exp(-g_{t,j}/\lambda_t)\right) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle$. *Let* $s_t = \max_{i\in[d]} g_{t,i} - \min_{i\in[d]} g_{t,i}$, *and define* $S = \max_{t\in[T]} s_t$. *Let* $\psi(\boldsymbol{x}) = \mathrm{KL}(\boldsymbol{x}; \boldsymbol{x}_1)$ *and* $\psi_t(\boldsymbol{x}) = \lambda_t\psi(\boldsymbol{x})$. *Then, for any* $\boldsymbol{u} \in \Delta_d$ *the regret of* Adahedge *(Algorithm 13) is bounded as*

$$R_T(\boldsymbol{u}) \leq \frac{\mathrm{KL}(\boldsymbol{u}; \boldsymbol{x}_1) + \alpha^2}{\alpha^2}\left(\alpha\sqrt{V_T} + \left(\frac{2}{3}\alpha^2 + 1\right)S\right) . \tag{6.7}$$

Proof. Let $\psi_t(\boldsymbol{x}) = \lambda_t\psi(\boldsymbol{x})$, where $(\lambda_t)_{t=1}^T$ is a non-decreasing sequence. From Equation (3.18), we have that

$$R_T(\boldsymbol{u}) \leq \sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u}\rangle \leq \lambda_T(\psi(\boldsymbol{u}) - \min_{\boldsymbol{x}\in V}\psi(\boldsymbol{x})) + \sum_{t=1}^T \left[F_t(\boldsymbol{x}_t) - F_{t+1}(\boldsymbol{x}_{t+1}) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle\right] .$$

In particular, we upper bound the sum over time on the right hand side above as follows

$$\begin{aligned}
F_t(\boldsymbol{x}_t) - F_{t+1}(\boldsymbol{x}_{t+1}) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle &= F_t(\boldsymbol{x}_t) - \lambda_{t+1}\psi(\boldsymbol{x}_{t+1}) - \sum_{i=1}^t \langle \boldsymbol{g}_i, \boldsymbol{x}_{t+1}\rangle + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle \\
&\leq F_t(\boldsymbol{x}_t) - \lambda_t\psi(\boldsymbol{x}_{t+1}) - \sum_{i=1}^t \langle \boldsymbol{g}_i, \boldsymbol{x}_{t+1}\rangle + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle \\
&\leq F_t(\boldsymbol{x}_t) - \min_{\boldsymbol{x}\in V}\left(\lambda_t\psi(\boldsymbol{x}) + \sum_{i=1}^t \langle \boldsymbol{g}_i, \boldsymbol{x}\rangle\right) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle \\
&= \psi_t(\boldsymbol{x}_t) - \langle \boldsymbol{\theta}_{t-1}, \boldsymbol{x}_t\rangle + \max_{\boldsymbol{x}\in V}\left(\langle \boldsymbol{\theta}_t, \boldsymbol{x}\rangle - \psi_t(\boldsymbol{x})\right) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle .
\end{aligned}$$

Note that the last expression above is always greater or equal than 0.

Now, we remember the definition of Fenchel conjugate, i.e., $\psi_t^\star(\boldsymbol{\theta}) = \max_{\boldsymbol{x}\in V}\left(\langle \boldsymbol{\theta}, \boldsymbol{x}\rangle - \psi_t(\boldsymbol{x})\right)$. It can be shown that that in the case of the KL divergence, i.e., $\psi(\boldsymbol{x}) = \mathrm{KL}(\boldsymbol{x}, \boldsymbol{x}_1)$, then the Fenchel conjugate is $\psi^\star(\boldsymbol{\theta}) = \ln(\sum_{i=1}^d x_{1,i}\exp(\theta_i))$.

Therefore, define $\boldsymbol{\theta}_t = -\sum_{i=1}^t \boldsymbol{g}_i$. We have that

$$\begin{aligned}
F_t(\boldsymbol{x}_t) - F_{t+1}(\boldsymbol{x}_{t+1}) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle &\leq -\psi_t^\star(\boldsymbol{\theta}_{t-1}) + \psi_t^\star(\boldsymbol{\theta}_t) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle \\
&= \lambda_t \ln\left(\frac{\sum_{i=1}^d x_{1,i}\exp(\theta_{t,i}/\lambda_t)}{\sum_{j=1}^d x_{1,i}\exp(\theta_{t-1,i}/\lambda_t)}\right) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle \\
&= \lambda_t \ln\left(\sum_{i=1}^d x_{t,i}\exp(-g_{t,i}/\lambda_t)\right) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle ,
\end{aligned}$$

which is exactly the definition of δ_t given in Algorithm 13, for $t \geq 2$.

Now, note that $\lambda_t = \frac{1}{\alpha^2}\sum_{i=1}^{t-1}\delta_i$. Therefore, we have that

$$\begin{aligned}
R_T(\boldsymbol{u}) &\leq (\psi(\boldsymbol{u}) - \psi(\boldsymbol{x}_1))\,\lambda_T + \sum_{t=1}^T \delta_t \leq \frac{\mathrm{KL}(\boldsymbol{u}, \boldsymbol{x}_1)}{\alpha^2}\sum_{t=1}^T \delta_t + \sum_{t=1}^T \delta_t \\
&= \frac{\mathrm{KL}(\boldsymbol{u}, \boldsymbol{x}_1) + \alpha^2}{\alpha^2}\sum_{t=1}^T \delta_t .
\end{aligned}$$

It remains to bound δ_t. We introduce $\Delta_t \triangleq \sum_{s=1}^t \delta_t$. The analysis now follows from [de Rooij et al., 2014, Lemma 5.5], but we report it here for completeness. The strategy is to have a second order inequality in Δ_T, solving which will give us a regret bound. We have the following

$$\Delta_T^2 = \sum_{t=1}^T(\Delta_t^2 - \Delta_{t-1}^2) = \sum_{t=1}^T\left[(\Delta_{t-1} + \delta_t)^2 - \Delta_{t-1}^2\right] = \sum_{t=1}^T(\delta_t^2 + 2\delta_t\Delta_{t-1})$$

$$= \sum_{t=1}^{T} (\delta_t^2 + 2\alpha^2 \lambda_{t+1} \delta_t) \leq \sum_{t=1}^{T} (s_t \delta_t + 2\alpha^2 \lambda_{t+1} \delta_t) \leq S\Delta_T + 2\alpha^2 \sum_{t=1}^{T} \lambda_{t+1} \delta_t$$

where the second to last inequality derives from the fact that $\delta_t \leq s_t$ using Lemma 6.4, while the last inequality from the definition of S. We are now left with providing a bound on the quantity $\lambda_{t+1}\delta_t$. This can be done following the same analysis in de Rooij et al. [2014][1] (details omitted), through a clever use of the *Bernstein* bound (see, e.g., [Cesa-Bianchi and Lugosi, 2006, Lemma A.5]) which leads to

$$\lambda_{t+1}\delta_t \leq \frac{1}{2} v_t + \frac{1}{3} s_t \delta_t \leq \frac{1}{2} v_t + \frac{1}{3} S\delta_t \ ,$$

where $v_t = \sum_{i=1}^{d} x_{t,i}(g_{t,i} - \langle \boldsymbol{x}_t, \boldsymbol{g}_t \rangle)^2$.

We therefore have

$$\Delta_T^2 \leq 2\alpha^2 \sum_{t=1}^{T} \left(\frac{1}{2} v_t + \frac{1}{3} S\delta_t \right) + S\Delta_T = \alpha^2 V_T + \left(\frac{2}{3}\alpha^2 + 1 \right) S\Delta_T \ ,$$

which is of the form $x^2 \leq a + bx$, with $x = \Delta_T, a = V_T\alpha^2, b = \left(\frac{2}{3}\alpha^2 + 1\right) S$. Solving for x we get

$$x \leq \frac{1}{2} b + \frac{1}{2}\sqrt{b^2 + 4a} \leq \frac{1}{2} b + \frac{1}{2}\left(\sqrt{b^2} + \sqrt{4a}\right) = b + \sqrt{a} \ .$$

Substituting back a, b and x completes the proof. □

Corollary 6.6. *Using $\boldsymbol{x}_1 = [1/d, \ldots, 1/d]$ and $\alpha = \sqrt{\ln d}$, Algorithm 13 incurs the following regret bound against any $\boldsymbol{u} \in \Delta_d$*

$$R_T(\boldsymbol{u}) \leq 2\sqrt{V_T \ln d} + \left(\frac{4}{3} \ln d + 2 \right) S \ . \tag{6.8}$$

Interestingly, it is shown in de Rooij et al. [2014] that the regret bound in Equation (6.8) implies improved first-order bounds with respect to the algorithm given in Section 6.1.2. Furthermore, Algorithm 13 is invariant to translation and rescaling of the losses and does not need to know the range of the losses in advance. In fact, this holds for every LEA algorithm attaining a bound in terms of variance over actions. To see this, consider a random variable Z_t in an unknown interval $[a_t, a_t + i_t]$. Then, we have that $\mathrm{Var}(Z_t) \leq i_t^2/4$. Indeed, consider a random variable $X \in [0, 1]$. We have that

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - \big(\mathbb{E}[X]\big)^2 \leq \mathbb{E}[X] - \big(\mathbb{E}[X]\big)^2 \leq 1/4 \ ,$$

since the function $f(x) = x - x^2$ has maximum value $1/4$ in $x = 1/2$.

Now, consider the variable $Z = a + (b - a)X$. We have that

$$\mathrm{Var}(Z) = \mathrm{Var}(a + (b-a)X) = \mathrm{Var}((b-a)X) = (b-a)^2 \ ,$$

and therefore $\mathrm{Var}(Z) \leq (b-a)^2/4$. Hence, assuming that $\max_t i_t \leq I$, an algorithm with a regret bound order of $\sqrt{V_T}$ would automatically imply a regret bound of $\mathcal{O}\big(I\sqrt{T \ln d}\big)$ without the need of knowing in advance neither a_t nor I.

Despite all these advantages, *Adahedge* has a parameter to be tuned, i.e., α^2. Since we do not have any prior knowledge of the vector $\boldsymbol{u}$, we can only set this parameter in order to prevent the worst-case term of $\ln d$, as shown in Corollary 6.6. On the other hand, next we describe another algorithm, which can reduce this term if the number of experts performing well is more than 1, or in general a fraction εd of the total.

[1] It is assumed that $s_t/\lambda_t \geq 0$ and therefore $s_t \geq 0$ since λ_t is never negative.

Algorithm 14 Hierarchical Algorithm

Require: LEA algorithm $\mathcal{A}$, K base algorithms $\mathcal{B}_1, \dots, \mathcal{B}_K$.
Initialize base algorithm $\mathcal{B}_j$ by setting $C_0^j = 0$ for all j.
for $t = 1, \dots, T$ **do**
  Get the prediction of $\mathcal{B}_j$, $\boldsymbol{x}_t^j$ for all j
  Get the prediction of the master $\mathcal{A}$, $\boldsymbol{q}_t$
  Compute $\boldsymbol{x}_t = \sum_{j=1}^K q_{t,j} \boldsymbol{x}_t^j$
  Observe $\ell_t(\boldsymbol{x}_t) = \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle$
  Define $c_t^j = \langle \boldsymbol{x}_t^j, \boldsymbol{g}_t \rangle$
  Update $C_t^j = C_{t-1}^j + c_t^j$ for all j and update $\boldsymbol{q}_t$
  Pass $\boldsymbol{g}_t$ to $\mathcal{B}_1, \dots, \mathcal{B}_K$ and update $\boldsymbol{x}_t^j$ for all $j \in [K]$
end for

6.3 Quantile bounds

As already sketched in the introduction, there may be situations when many experts perform well at the same time. In such cases, we may not want to compete against the single best expert, but instead against the best fraction of experts. Formally, we assume that $\boldsymbol{u}$ is concentrated on a fraction $\varepsilon = j^*/d$ (where j^* is an integer) of the experts, i.e. $\boldsymbol{u} = (1/\varepsilon d, \dots, 1/\varepsilon d, 0, \dots, 0)$. In this case a bound of $\mathcal{O}(\sqrt{T \ln d})$ might be too pessimistic, especially if the number of experts is high. Next, we show how to achieve a dependence on $\ln(1/\varepsilon)$ in the bound, by adopting a simple technique to combine online learning algorithms.

6.3.1 Combining different algorithms

In this section, we explore an approach proposed in Luo [2017]. In order to get the desired regret bound of $\mathcal{O}(\sqrt{T \ln(1/\varepsilon)})$, their idea is to use a "hierarchical" algorithm. There is a master algorithm $\mathcal{A}$ which keeps a distribution $\boldsymbol{q}_t \in \mathbb{R}^K$ over K base algorithms $\mathcal{B}_1, \dots, \mathcal{B}_K$, which in turn keep a distribution over experts, $\boldsymbol{x}_t^j \in \mathbb{R}^d$ for $j \in [K]$. Each base algorithm $\mathcal{B}_j$ has a different tuning of the learning rate. The goal is to prove that there is a base algorithm $\mathcal{B}_{j^*}$ with optimal tuning, and that the master algorithm identifies it quickly enough.

In particular, at each round the master $\mathcal{A}$ receives the distributions of the base algorithms and outputs a prediction $\boldsymbol{x}_t$ by weighting them according to its own distribution $\boldsymbol{q}_t$. After observing the loss, both the master and the base algorithms get updated. This algorithm is depicted in Algorithm 14. In particular, it is an adaptation of Algorithm 12 with a constant learning rate both as master and base algorithm. Also, for simplicity it is again assumed that $\|\boldsymbol{g}_t\|_\infty \leq 1$ for all t. Next, we report their regret bound for this hierarchical algorithm.

Theorem 6.7. *Consider the LEA setting, and assume $g_{t,i} \in [0,1]$ for all $i = 1, \dots, d$ and $t \in [T]$. Set K such that*

$$K = \left\lceil \log_2 \sqrt{\frac{\ln d}{\ln \frac{d}{d-1}}} \right\rceil + 1 \,, \tag{6.9}$$

and for $j = 1, \dots, K$ set $\mathcal{B}_j$ as the Exponential Weights update algorithm (Algorithm 12), such that $\eta_{t,j} = \eta_j = \frac{1}{2^j}\sqrt{\ln d / T}$. Furthermore, choose $\mathcal{A}$ as the Exponential Weights update algorithm with $\eta_t = \eta = \sqrt{\ln K / T}$ for all $t = 1, \dots, T$.

Then, for any $\boldsymbol{u} \in \Delta_d$ such that $\mathrm{KL}(\boldsymbol{u}, \vec{\mathbf{1}}\frac{1}{d}) = \ln(1/\varepsilon)$ and $\varepsilon = \Omega(\ln(d \ln d)/d)$, the regret incurred by Algorithm 14 satisfies

$$R_T(\boldsymbol{u}) = \mathcal{O}(\sqrt{T \ln 1/\varepsilon}) \tag{6.10}$$

Proof. For the sake of the analysis, the regret bound can be decomposed as follows

$$R_T(\boldsymbol{u}) = \sum_{t=1}^T \ell_t(\boldsymbol{x}_t) - \sum_{t=1}^T \ell_t(\boldsymbol{u})$$

$$= \underbrace{\sum_{t=1}^{T} \langle \boldsymbol{x}_t - \boldsymbol{x}_t^j, \boldsymbol{g}_t \rangle}_{(a)} + \underbrace{\sum_{t=1}^{T} \langle \boldsymbol{x}_t^j - \boldsymbol{u}, \boldsymbol{g}_t \rangle}_{(b)} ,$$

where the first term (a) is the regret of the master compared to any base algorithm, while the second term (b) is the regret of any base algorithm against the comparator $\boldsymbol{u}$. We analyze the two terms separately.

For (a), note that this is a regular LEA problem, where the experts are the base algorithms. Therefore, from Theorem 3.11 for every $j \in [K]$ with we have that

$$\sum_{t=1}^{T} \langle \boldsymbol{x}_t - \boldsymbol{x}_t^j, \boldsymbol{g}_t \rangle = \sum_{t=1}^{T} \sum_{j=1}^{K} q_{t,j} \langle \boldsymbol{x}_t^j, \boldsymbol{g}_t \rangle - \sum_{t=1}^{T} \langle \boldsymbol{x}_t^j, \boldsymbol{g}_t \rangle \le \frac{\ln K}{\eta} + \frac{\eta}{2} T , \tag{6.11}$$

where K is the number of base algorithms.

On the other hand, each base algorithm is facing the regular LEA problem in Δ_d. Therefore, for (b) we have that for any j-th base algorithm from Theorem 3.11 the regret is bounded as

$$\sum_{t=1}^{T} \langle \boldsymbol{x}_t^j - \boldsymbol{u}, \boldsymbol{g}_t \rangle \le \frac{\ln(1/\varepsilon)}{\eta_j} + \frac{\eta_j}{2} T . \tag{6.12}$$

Summing up the two contributions from Equations (6.11) and (6.12), and using $\eta = \sqrt{\ln K / T}$, we get

$$R_T(\boldsymbol{u}) \le 2\sqrt{T \ln K} + \frac{\ln(d/j^*)}{\eta_j} + \frac{\eta_j}{2} T .$$

We now analyze the overall regret bound. We first analyze the contribution of the optimal base algorithm and verify later that the master satisfies $K \sim \ln d$. Notice that with this choice of learning rates, the whole range $[\ln \frac{d}{d-1}, \dots, \ln d]$ is covered, so that there is a j^+ such that $\eta_{j^+} \le \sqrt{\frac{\ln(d/j^*)}{T}} \le \eta_{j^+-1} = 2\eta_{j^+}$. Hence, we bound the contribution of the j^+ base as follows

$$\begin{aligned} R_T(\boldsymbol{u}) &\le \overbrace{\frac{\ln(1/\varepsilon)}{\eta_{j^+}} + \eta_{j^+} T}^{\text{optimal base}} + \overbrace{2\sqrt{T \ln K}}^{\text{master}} \\ &\le \frac{\ln(1/\varepsilon)}{\frac{1}{2}\sqrt{\frac{\ln(1/\varepsilon)}{T}}} + \sqrt{T \ln(1/\varepsilon)} + 2\sqrt{T \ln K} \\ &= \mathcal{O}\big(\sqrt{T \ln(1/\varepsilon)} + \sqrt{T \ln K}\big) . \end{aligned}$$

Now, remember that

$$K = \left\lceil \log_2 \sqrt{\frac{\ln d}{\ln \frac{d}{d-1}}} \right\rceil + 1 .$$

Using that $\ln(1+x) \ge x/2$ for all $x \le 1$, we have that $\ln(\frac{d}{d-1}) = \ln(1 + \frac{1}{d-1}) \ge \frac{1}{2(d-1)}$.

Therefore,

$$K \le \left\lceil \log_2 \sqrt{2(d-1) \ln d} \right\rceil + 1 = \mathcal{O}\big(\ln(d \ln d)\big) .$$

So, as long as εd is bigger than $\mathcal{O}\big(\ln(d \ln d)\big)$ we have the desired bound, which is

$$R_T(\boldsymbol{u}) = \mathcal{O}\big(\sqrt{T \ln(1/\varepsilon)}\big) + \mathcal{O}\big(\sqrt{T \ln \ln(d \ln d)}\big) = \mathcal{O}\big(\sqrt{T \ln(1/\varepsilon)}\big) .$$

□

In the following we explain some design choices for Algorithm 14.

First, note that the optimal tuning would be

$$\eta_j^* = \sqrt{\frac{\ln(d/j^*)}{T}} ,$$

for a certain $j^* \in [1, d]$. An intuitive choice would be to have an algorithm for every possible value of j from 1 to d. In this way, there would certainly be a base with the optimal η_j^*. However, the number of base algorithms K would be equal to d and thus the master's contribution to the regret bound (the term (a)) would be $\mathcal{O}\sqrt{T \ln d}$, preventing us to get the desired bound of $\mathcal{O}(\sqrt{T \ln \varepsilon})$.

In order to have a reduced number of base algorithms, a logarithmically spaced grid for η_j is used. As also explained in Luo [2017], we want the learning rate to span the interval $[\ln \frac{d}{d-1}, \dots, \ln d]$, i.e.

$$\eta_j \in \left[\sqrt{\ln \frac{d}{d-1} \Big/ T}, \dots, \sqrt{\frac{\ln d}{T}} \right] . \tag{6.13}$$

Therefore, as stated in the assumption of the theorem above, the following choice is adopted:

$$\eta_j = \frac{1}{2^j} \sqrt{\frac{\ln d}{T}} \quad , \quad j = 0, \dots, K-1 \ .$$

In particular, we have that $\eta_0 = \sqrt{\ln d / T}$, while $\eta_K = \frac{1}{2^{K-1}} \sqrt{\frac{\ln d}{T}}$. In order to span the whole range in Equation (6.13), we want the latter value to be approximately equal to $\sqrt{\ln \frac{d}{d-1} / T}$. Hence,

$$\frac{1}{2^{K-1}} \sqrt{\frac{\ln d}{T}} = \sqrt{\frac{\ln \frac{d}{d-1}}{T}} \ ,$$

and solving for K yields

$$K = \log_2 \sqrt{\frac{\ln d}{\ln \frac{d}{d-1}}} + 1 \ .$$

Since K has to be an integer we round it to the next integer, therefore the choice adopted in Equation (6.9).

This technique is indeed more general, and can be used anytime there is an unknown quantity needed in the tuning of the learning rate, in order to get a tighter bound. Unfortunately, it cannot be extended to get a data-dependent bound, since the master algorithm in the worst case incurs a regret bound order of $\mathcal{O}(\sqrt{T})$ in order to learn which one is the best base algorithm.

We point out that there exist more elegant and general approaches to this problem, which can guarantee regret bounds dependent on $\mathrm{KL}(\boldsymbol{u}, \boldsymbol{x}_1)$, for any $\boldsymbol{x}_1 \in \Delta_d$, such as the ones in Orabona and Pal [2016]; Koolen and van Erven [2015]. However, the approach of combining algorithms is a quick way to verify whether a regret bound is achievable. To this aim, in the next section we provide a similar technique, which under particular assumptions achieves a better regret bound compared to Equation (6.10).

6.3.2 Preliminary results

Up to this point, all the results described regarding the LEA setting are known in the literature. In this section, we slightly modify the combiner technique given in the previous section and provide a different regret bound.

In the previous section, we have seen how the regret bound of $\mathcal{O}(\sqrt{T \ln(1/\varepsilon)})$ of Algorithm 14 cannot be improved, since the standard Exponential Weights algorithm used as master in the worst case incurs a regret bound of $\mathcal{O}(\sqrt{T})$. On the other hand, we described in Section 6.1.2 how this algorithm can be adapted to get a first-order bound. What we propose is to use the Exponential Weights algorithm as the master algorithm and as tuning the one suggested in Theorem 6.3. Also, we set the base learners as *Adahedge* and we modify them to use a different tuning of the parameter α^2. We have the following result.

Theorem 6.8. *Define $\mathcal{B}_1, \dots, \mathcal{B}_K$ as K different copies of* Adahedge *(Algorithm 13), each with $\alpha = \alpha_j$ for $j \in [K]$. Denote $R_{T,\mathcal{B}_j}(\boldsymbol{u})$ as the regret bound of $\mathcal{B}_j$ against any $\boldsymbol{u} \in \Delta_d$ such that $\mathrm{KL}(\boldsymbol{u}, \mathbf{1}\frac{1}{d}) = \ln(1/\varepsilon)$. Let $\alpha_j^2 = \frac{1}{2^j} \ln d$ for $j \in [0, K-1]$, with $K = \left\lceil \log_2 \frac{\ln d}{\ln \frac{d}{d-1}} \right\rceil + 1$.*

Then there is a $j^* \in [K]$ such that the following holds

$$R_{T,\mathcal{B}_{j^*}}(\boldsymbol{u}) \leq 4\sqrt{V_T^{j^*} \ln(1/\varepsilon)} + 4S\left(\frac{2}{3}\ln(1/\varepsilon) + 1\right) , \tag{6.14}$$

where $V_T^{j^*}$ is the variance over actions of $\mathcal{B}_{j^*}$ according to the definition in Eq. (6.3).

Proof. Assuming $\varepsilon \in [1/d, (d-1)/d]$, with our grid of α_j we have a $j^\star$ such that $\alpha_{j^\star}^2 \leq \ln(1/\varepsilon) \leq \alpha_{j^\star+1}^2$, i.e.

$$\alpha_{j^*}^2 = \frac{1}{2^{j^\star}} \ln d \leq \ln \frac{1}{\varepsilon} \leq \frac{1}{2^{j^\star - 1}} \ln d = 2\alpha_{j^\star}^2 .$$

Therefore, considering the regret bound of Equation (6.7), we have that there is a j^* such that

$$\begin{aligned} R_{T,\mathcal{B}_{j^*}}(\boldsymbol{u}) &\leq \frac{\ln(1/\varepsilon) + \alpha_{j^*}^2}{\alpha_{j^*}^2}\left(\alpha_{j^*}\sqrt{V_T^{j^*}} + \left(\frac{2}{3}\alpha_{j^*}^2 + 1\right) S\right) \\ &\leq \frac{2\ln(1/\varepsilon)}{\frac{1}{2}\ln(1/\varepsilon)}\left(\sqrt{V_T^{j^*}\ln(1/\varepsilon)} + \left(\frac{2}{3}\ln(1/\varepsilon) + 1\right) S\right) \\ &= 4\sqrt{V_T^{j^*}\ln(1/\varepsilon)} + 4S\left(\frac{2}{3}\ln(1/\varepsilon) + 1\right) , \end{aligned}$$

where $V_T^{j^*} = \sum_{t=1}^T \sum_{i=1}^d x_t^{j^*}(g_{t,i} - \langle \boldsymbol{x}_t^{j^*}, \boldsymbol{g}_t\rangle)^2$. □

Hence, by adopting the exponentially spaced grid in the same spirit of the previous section, a direct application of this modified hierarchical algorithm with K base learners gives a regret bound of,

$$R_T(\boldsymbol{u}) = \tilde{\mathcal{O}}\left(\sqrt{L_T^{j^*}\ln K} + \sqrt{V_T^{j^*}\ln(1/\varepsilon)}\right) , \tag{6.15}$$

where $L_T^{j^*}$ is the cumulative loss of the best base learner, i.e., $L_T^{j^*} = \sum_{t=1}^T \langle \boldsymbol{x}_t^{j^*}, \boldsymbol{g}_t\rangle$, and $\boldsymbol{x}_t^{j^*}$ is the output of the j^*-th learner at time t.

Ideally, in order to have a regret bound of $\mathcal{O}(\sqrt{V_T \ln(1/\varepsilon)})$ as asked in Freund [2016], one should have $L_T^{j^*} \simeq V_T^{j^*}$. However, the term $L_T^{j^*}$ can be T in the worst-case (for example when losses are in $[0,1]$), therefore ruining the bound in Equation (6.15).

Modified losses. We further modify the losses fed to the master algorithm in Algorithm 14 and we let $c_t^j = \langle \boldsymbol{x}_t^{j^*}, \boldsymbol{g}_t\rangle - \langle \boldsymbol{u}_t, \boldsymbol{g}_t\rangle$. If we adopt $\boldsymbol{u}_t = \operatorname{argmin}_{\boldsymbol{x}\in\Delta_d}\langle \boldsymbol{g}_t, \boldsymbol{x}\rangle$, then the losses c_t^j fed to the master algorithm are still in the range $[0,1]$ and Theorem 6.3 can be applied. We then prove the following theorem. This result is not only restricted to the symplex, but any convex set V.

Theorem 6.9. *Given a convex set V, suppose $\mathcal{B}_1, \ldots, \mathcal{B}_K$ are OLO algorithms with regret upper bounds $R_{T,\mathcal{B}_j}(\boldsymbol{u})$, for any $\boldsymbol{u} \in V$. Assume $g_{t,i} \in [0,1]$ for all i and t and define $c_t^j = \langle \boldsymbol{x}_t^j - \boldsymbol{u}_t^*, \boldsymbol{g}_t\rangle$, where $\boldsymbol{u}_t^* = \operatorname{argmin}_{\boldsymbol{x}\in V}\langle \boldsymbol{g}_t, \boldsymbol{x}\rangle$. Suppose Algorithm 14 is run using EWU with the tuning suggested in Theorem 6.3 as $\mathcal{A}$, and $C_t^j = \sum_{i=1}^t c_i^j$. Then, for any $\boldsymbol{u} \in V$ the regret is bounded as*

$$R_T(\boldsymbol{u}) = \tilde{\mathcal{O}}\left(R_{T,\mathcal{B}_{j^*}}(\boldsymbol{u}) + \sqrt{\ln K\, R_{T,\mathcal{B}_{j^*}}^{\mathrm{dyn}}}\right) , \tag{6.16}$$

where $R_{T,\mathcal{B}_{j^*}}^{\mathrm{dyn}} = \sum_{t=1}^T \langle \boldsymbol{x}_t^{j^*} - \boldsymbol{u}_t^*, \boldsymbol{g}\rangle$ is the dynamic regret of algorithm $\mathcal{B}_{j^*}$ against the sequence $(\boldsymbol{u}_1^*, \ldots, \boldsymbol{u}_T^*)$.

Proof. We consider the regret of the master $\mathcal{A}$ against another ideal algorithm who always picks the best base algorithm $\mathcal{A}_{j^*}$, defined as follows

$$\sum_{t=1}^T \langle \boldsymbol{q}_t, \boldsymbol{C}_t\rangle - \sum_{t=1}^T C_{t,j^*} = \sum_{t=1}^T \sum_{j=1}^K q_{t,j} C_{t,j} - \sum_{t=1}^T C_{t,j^*} .$$

Applying Theorem 6.3, we immediately have the following result,

$$\sum_{t=1}^{T}\langle \boldsymbol{q}_t, \boldsymbol{C}_t\rangle \leq \sum_{t=1}^{T} C_{t,j^*} + 2\sqrt{\ln K \sum_{t=1}^{T} C_{t,j^*}} + 4\ln K + 2\sqrt{\ln K}\ . \tag{6.17}$$

Now, note that $C_{t,j^*} = \sum_{t=1}^{T}\langle \boldsymbol{x}_t^j - \boldsymbol{u}_t^*, \boldsymbol{g}_t\rangle = R^{\text{dyn}}_{T,\mathcal{B}_{j^*}}$.

Furthermore, we have that

$$\begin{aligned}\sum_{t=1}^{T}\langle \boldsymbol{q}_t, \boldsymbol{C}_t\rangle - \sum_{t=1}^{T} C_{t,j^*} &= \sum_{t=1}^{T}\sum_{j=1}^{K} q_{t,j}\langle \boldsymbol{x}_t^j - \boldsymbol{u}_t^*, \boldsymbol{g}_t\rangle - \sum_{t=1}^{T}\langle \boldsymbol{x}_t^{j^*} - \boldsymbol{u}_t^*, \boldsymbol{g}_t\rangle \\ &= \sum_{t=1}^{T}\langle \boldsymbol{x}_t, \boldsymbol{g}_t\rangle - \sum_{t=1}^{T}\langle \boldsymbol{x}_t^{j^*}, \boldsymbol{g}_t\rangle\ .\end{aligned}$$

Therefore, substituting this bound back in Eq. (6.17) and subtracting $\sum_{t=1}^{T}\langle \boldsymbol{u}, \boldsymbol{g}_t\rangle$ on both sides, we get

$$\begin{aligned}\sum_{t=1}^{T}\langle \boldsymbol{x}_t - \boldsymbol{u}, \boldsymbol{g}_t\rangle &\leq \sum_{t=1}^{T}\langle \boldsymbol{x}_t^{j^*} - \boldsymbol{u}, \boldsymbol{g}_t\rangle + 2\sqrt{\ln K R^{\text{dyn}}_{T,\mathcal{B}_{j^*}}} + 4\ln K + 2\sqrt{\ln K} \\ &= \mathcal{O}\left(R_{T,\mathcal{B}_{j^*}} + \sqrt{\ln K R^{\text{dyn}}_{T,\mathcal{B}_{j^*}}}\right)\ .\end{aligned}$$

□

Unfortunately, $R^{\text{dyn}}_{T,\mathcal{B}_{j^*}}$ can still be linear in time, in which case the second term in Eq. (6.16) would lead to a $\mathcal{O}(\sqrt{T})$ regret bound. On the other hand, note that

$$R^{\text{dyn}}_{T,\mathcal{B}_{j^*}} = \sum_{t=1}^{T}\langle \boldsymbol{x}_t^{j^*} - \boldsymbol{u}_t^*\rangle = R_{T,\mathcal{B}_{j^*}}(\boldsymbol{u}) + \sum_{t=1}^{T}\langle \boldsymbol{u} - \boldsymbol{u}_t^*, \boldsymbol{g}_t\rangle \ \leq R_{T,\mathcal{B}_{j^*}}(\boldsymbol{u}) + \sum_{t=1}^{T}\|\boldsymbol{u} - \boldsymbol{u}_t^*\|,$$

for any $\boldsymbol{u} \in V$. Ideally, if $\boldsymbol{u}_t^*$ stays fixed for the whole game, i.e., $\boldsymbol{u}_t^* = \operatorname{argmin}_{\boldsymbol{x}\in V}\sum_{i=1}^{T}\langle \boldsymbol{g}_i, \boldsymbol{x}\rangle$ then the algorithm described in Theorem 6.9 would achieve the bound asked in Freund [2016], since the dynamic regret would coincide with the static regret. However, in an adversarial environment we cannot expect this assumption to hold. Nonetheless, if the sequence $\boldsymbol{u}_{1:T} = (\boldsymbol{u}_1^*, \dots, \boldsymbol{u}_T^*)$ is not varying much over time, for example with $\sum_{t=1}^{T}\|\boldsymbol{u} - \boldsymbol{u}_t^*\| = \mathcal{O}(\ln T)$, then the argument would still be valid. On the other hand, if the sequence $\boldsymbol{u}_t$ is oscillating between multiple good experts, then there is no hope of getting something less than $\mathcal{O}(T)$.

Next, we are going to explore a different combiner, which is based on the same idea of keeping track of the (local) regret of the base algorithms.

A new combiner. We now illustrate how to use another combiner algorithm recently proposed in Bhaskara et al. [2020] and reported in Algorithm 15. The algorithm is based on a doubling trick on the (local) regret incurred by each base algorithm. It starts by choosing one base algorithm at random and keeps playing it until its measured (local) regret exceeds a certain threshold. Then, if other better performing base algorithms are still available, it selects again one of them at random and repeats the process. This process goes on until all the base algorithms exceed the threshold. At this point the threshold is doubled and the combiner is restarted. The main goal is to count the number of phases and show that it is logarithmic in the regret upper bound of the best base algorithm.

For technical reasons, Bhaskara et al. [2020] need to define the notion of monotone regret bound. Note that this notion is satisfied by all existing algorithms in the online linear optimization literature, whose bound does not depend on the competitor $\boldsymbol{u}$ at hand.

Definition 6.10 (Monotone regret bound)**.** *Given a set V and a sequence of cost vectors $\boldsymbol{g}_{1:T} = (\boldsymbol{g}_1, \boldsymbol{g}_2, \dots, \boldsymbol{g}_T)$, an online learning algorithm $\mathcal{B}$ is associated with a monotone regret*

Algorithm 15 Randomized combiner

Require: Online algorithms $\mathcal{B}_1, \dots, \mathcal{B}_K$.
Reset $\mathcal{B}_1$
Set $\gamma \leftarrow 1, \tau \leftarrow 1$
Initialize the candidate indices $C \leftarrow [K]$
Choose index i uniformly at random from C
for $t = 1, \dots, T$ **do**
  for $j \in C$ **do**
    Get $\boldsymbol{y}_\tau^j$, the τ-th output of $\mathcal{B}_j$
  end for
  Respond $\boldsymbol{x}_t \leftarrow \boldsymbol{y}_\tau^i$
  Observe loss $\ell_t(\boldsymbol{x}_t)$ and get (sub)gradient $\boldsymbol{g}_t$
  for $J \in C$ **do**
    Send $\boldsymbol{g}_\tau$ to $\mathcal{B}_j$ as τ-th cost
    Set $r_\tau^{j,\gamma} \leftarrow \sum_{\tau'=1}^{\tau} \langle \boldsymbol{y}_{\tau'}^j, \boldsymbol{g}_{\tau'} \rangle - \min_{\boldsymbol{u} \in V} \sum_{\tau'=1}^{\tau} \langle \boldsymbol{u}, \boldsymbol{g}_{\tau'} \rangle$
    if $r_\tau^{j,\gamma} > \gamma$ **then**
      Set $C \leftarrow C \setminus \{j\}$
    end if
  end for
  if $i \notin C$ **then**
    if $C = \emptyset$ **then**
      Set $C \leftarrow [K]$
      Set $\gamma \leftarrow 2\gamma$
    end if
    Set $\tau \leftarrow 1$
    Reset $\mathcal{B}_j$ for all $j \in C$
    Select index i uniformly at random from C
  end if
  Set $\tau \leftarrow \tau + 1$
end for

bound $\mathcal{S}([s,e], \boldsymbol{g}_{s:e})$, if $\mathcal{S}(\cdot,\cdot)$ is such that when $\mathcal{B}$ is run only on the costs $\boldsymbol{g}_s, \dots, \boldsymbol{g}_e$, producing outputs $\boldsymbol{x}_s, \dots, \boldsymbol{x}_e$, we have the guarantee:

$$\sum_{t=s}^{e} \langle \boldsymbol{x}_t, \boldsymbol{g}_t \rangle - \min_{\boldsymbol{u} \in V} \sum_{t=s}^{e} \langle \boldsymbol{u}, \boldsymbol{g}_t \rangle \le \mathcal{S}([s,e], \boldsymbol{g}_{1:T}) \ ,$$

and further it satisfies $\mathcal{S}([s',e'], \boldsymbol{g}_{1:T}) \le \mathcal{S}([s,e], \boldsymbol{g}_{1:T})$ for all sequences $\boldsymbol{g}_{1:T}$ whenever $[s',e'] \subseteq [s,e]$.

With the use of the above definition, Bhaskara et al. [2020] provide the following regret upper bound to Algorithm 15.

Theorem 6.11 (Bhaskara et al. [2020])**.** *Suppose $\mathcal{B}_1, \dots, \mathcal{B}_K$ are deterministic OLO algorithms with monotone regret bounds $\mathcal{S}_1, \dots, \mathcal{S}_K$. Given a constrained set V, suppose for all t, $\sup_{\boldsymbol{x}, \boldsymbol{y} \in V} \langle \boldsymbol{g}_t, \boldsymbol{x} - \boldsymbol{y} \rangle \le 1$. Then for any fixed sequence of costs $\boldsymbol{g}_{1:T} = (\boldsymbol{g}_1, \dots, \boldsymbol{g}_T)$ and any $\boldsymbol{u} \in V$ (i.e. an oblivious adversary), Algorithm 15 guarantees:*

$$\mathbb{E}[R_T(\boldsymbol{u})] \le \log_2(K+1) \cdot \left(4 + 4 \min_{i \in [K]} S_{\mathcal{B}_i}([1,T], \boldsymbol{g}_{1:T}) \right) \ . \tag{6.18}$$

Further, if the sequence of costs is allowed to depend on the algorithm's randomness (i.e., an adaptive adversary), then

$$R_T(\boldsymbol{u}) \le K \left(4 + 4 \min_{i \in [K]} S_{\mathcal{B}_i}([1,T], \boldsymbol{g}_{1:T}) \right) \ , \tag{6.19}$$

The above result says that by using Algorithm 15, we can run K different algorithms in parallel and retain the regret bound of the best of them, up to a multiplicative factor (in K

or $\log_2 K$), assuming that the bound is *monotone* as defined in Definition 6.10.

Therefore, we can immediately apply this strategy for the problem of variance bounds over actions in the setting of LEA. We have the following corollary.

Corollary 6.12. *Suppose we adopt the grid of base Adahedge algorithms described in Theorem 6.8, and combine them using Algorithm 15. Then the regret against any $\boldsymbol{u} \in \Delta_d$ is bounded as*

$$\mathbb{E}[R_T(\boldsymbol{u})] = \mathcal{O}\left(\min_{j\in[K]}\left(\sqrt{2^j} + \sqrt{\frac{1}{2^j}}\right)\sqrt{V_T^j \ln d}\right) , \tag{6.20}$$

where $V_T^j = \sum_{t=1}^T \sum_{i=1}^d x_t^j (g_{t,i} - \langle \boldsymbol{x}_t^j, \boldsymbol{g}_t\rangle)^2$.

Proof. The result follows by applying Theorem 6.11 and noticing that the regret bound of Adahedge is monotone, when we discard the dipendence on $\boldsymbol{u}$. Indeed, for any j we have that

$$\begin{aligned} R_{T,\mathcal{B}_j}(\boldsymbol{u}) &= \mathcal{O}\left(\left(\frac{\mathrm{KL}(\boldsymbol{u}, \boldsymbol{x}_1)}{\alpha} + \alpha\right)\sqrt{V_T^j}\right) \\ &\leq \mathcal{O}\left(\left(\frac{\ln d}{\sqrt{\frac{\ln d}{2^j}\ln d}} + \sqrt{\frac{\ln d}{2^j}}\right)\sqrt{V_T^j}\right) \\ &= \mathcal{O}\left(\left(\sqrt{2^j} + \sqrt{\frac{1}{2^j}}\right)\sqrt{V_T^j \ln d}\right) . \end{aligned}$$

□

The result in Eq. (6.20) might seem at first disappointing, since it does not involve quantiles at all. However, upon a better inspection it can be noted that the variance term in the bound is the one relative to the best base algorithm. The variance over actions $V_T^{j^*}$ depends on the distribution played by algorithm j^* in every round, which in turn depends on the α_{j^*} in the learning rate. By using the tuning suggested in Theorem 6.8, this could induce a quantile bound for $\mathcal{B}_{j^*}$.

Unfortunately, the requirement of having a *monotone* regret bound for the base algorithms prevents us from obtaining a quantile-bound. We point out that this requirement is necessary in order to provide a regret bound to Algorithm 15. In particular, it is used to bound the number of times the combiner algorithm is restarted, as shown in the next lemma.

Lemma 6.13 ([Bhaskara et al., 2020])**.** *Suppose $\mathcal{B}_1, \dots, \mathcal{B}_K$ are deterministic OLO algorithms with monotone regret bounds $\mathcal{S}_1, \dots, \mathcal{S}_K$. Define a phase as a time interval $[s, e] \subseteq [T]$ in which γ stays constant. Then, the total number of phases P of Algorithm 15 satisfies*

$$P \leq 2 + \max(-1, \log_2(\min_i \mathcal{S}_i([1,T], \boldsymbol{g}_{1:T}))) . \tag{6.21}$$

Proof. It is natural to divide the operations of Algorithm 15 into phases in which γ is constant. Let P denote the total number of phases. We claim that Eq. (6.21) holds. Suppose otherwise. Let $j = \operatorname{argmin}_i \mathcal{S}_i([1,T], \boldsymbol{g}_{1:T})$ be the algorithm with the least total regret. Let us consider the $(P-1)$-th phase. In this phase, $\gamma = 2^{P-2}$ (note that the counter starts from 0). Since $P > 2 + \log_2(\min_i \mathcal{S}_i([1,T], \boldsymbol{g}_{1:T}))$, we must have $\min_i \mathcal{S}_i([1,T], \boldsymbol{g}_{1:T}) < \gamma$. Consider the j-th sub-phase in this phase. Since γ will eventually increase, this sub-phase must eventually end. Therefore there must be some t and τ such that $t + \tau < T$ and

$$\sum_{\tau'=1}^{\tau} \langle \boldsymbol{w}_{\tau'}, \boldsymbol{g}_\tau\rangle - \min_{\boldsymbol{u}\in V} \sum_{\tau'=1}^{\tau} \langle \boldsymbol{u}, \boldsymbol{g}_\tau\rangle \geq \gamma ,$$

where $\boldsymbol{w}_{\tau'}$ is the output of $\mathcal{B}_j$ after seeing input $\boldsymbol{g}_t, \dots, \boldsymbol{g}_{t+\tau'-1}$. By the increasing property of S_j, we also have:

$$\sum_{\tau'=1}^{\tau} \langle \boldsymbol{w}_{\tau'}, \boldsymbol{g}_\tau\rangle - \min_{\boldsymbol{u}\in V} \sum_{\tau'=1}^{\tau} \langle \boldsymbol{u}, \boldsymbol{g}_\tau\rangle \leq \mathcal{S}_j([t, t+\tau], \boldsymbol{g}_{1:T}) \leq \mathcal{S}_j([1,T], \boldsymbol{g}_{1:T}) < \gamma , \tag{6.22}$$

which is a contradiction. Therefore $P \leq 2 + \max(-1, \log_2(\min_i S_i([1,T], \boldsymbol{g}_{1:T})))$. □

From the above lemma, it is easy to see that the assumption of monotonicity of the regret bound is crucially used in the last step in Eq. (6.22). This is not surprising: since the combiner is measuring the local regret, each base algorithm is competing against a stronger baseline $\boldsymbol{u}_t$ on the interval of interest, compared to the global minimizer $\boldsymbol{u} = \operatorname{argmin}_{\boldsymbol{x} \in V} \sum_{i=1}^{T} \langle \boldsymbol{g}_i, \boldsymbol{x} \rangle$.

6.4 Discussion

Solving the open problem posed in Freund [2016] would pave the way for the design of new online learning algorithms. In particular, it would help to solve other standing open problems in the community, such as switching regret bounds for the bandit setting with an unknown number of switches, or improved regret bound for contextual bandits as explained in [Foster et al., 2020, Section 4].

Methods based on parameter-free strategies such as coin betting do not seem to work in this setting, because they introduce a dependence on $\boldsymbol{u}$ in the variance term, as done in Koolen and van Erven [2015]. As a first step, we tried to verify if this bound is achievable at all, by using the idea of combining algorithms. This idea is not new but can be refined in many ways, as we have explored in this chapter. While it may be not elegant, it is effective in establishing whether a regret bound can be achieved. Unfortunately, the methods proposed at the end of this chapter all involve measuring the local regret of the base algorithms, and this seems to hinder progress on obtaining quantiles in the bound.

Appendix A

Omitted Proofs

A.1 Adahedge

We recall Lemma 6.4 and report its proof in the following.

Lemma 6.4. *Let $\boldsymbol{x}_t$ be a probability distribution on Δ_d and $\boldsymbol{g}_t \in \mathbb{R}^d$. Define $s_t \triangleq \max_{i\in[d]} g_{t,i} - \min_{i\in[d]} g_{t,i}$ and δ_t as follows,*

$$\delta_t \triangleq \begin{cases} \langle \boldsymbol{g}_1, \boldsymbol{x}_1 \rangle, & t = 1 \\ \lambda_t \ln\left(\sum_{j=1}^d x_{t,j} \exp(-g_{t,j}/\lambda_t)\right) + \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle, & t \geq 2 \end{cases} \tag{6.6}$$

Then, we have that $0 \leq \delta_t \leq s_t$.

Proof. Let $m_t = -\lambda_t \ln \sum_{i=1}^d x_{t,i} \exp(-g_{t,i}/\lambda_t)$ and $h_t = \sum_{i=1}^d x_{t,i} g_{t,i}$. Observe that

$$\begin{aligned} \sum_{i=1}^d x_{t,i} g_{t,i} &= \sum_{i=1}^d -\lambda_t x_{t,i} \ln \exp(-g_{t,i}/\lambda_t) \geq -\lambda_t \ln \sum_{i=1}^d x_{t,i} \exp(-g_{t,i}/\lambda_t) \\ &\geq -\lambda_t \ln \sum_{i=1}^d x_{t,i} \exp(-\min_{i\in[d]} g_{t,i}/\lambda_t) = g_t^- \ , \end{aligned}$$

where in the first inequality we used the Jensen's inequality. Therefore, we have $g_t^- \leq m_t \leq h_t \leq g_t^+$. Now, observe that $0 \leq h_t - g_t^- \leq g_t^+ - g_t^- = s_t$. On the other hand, $h_t - g_t^- \geq h_t - m_t = \delta_t \geq 0$. Therefore, the result follows. □

A.2 Combiner

In this section, we provide the missing proofs for Algorithm 15 in Section 6.3.2.

In order to provide a regret bound for the combiner described in Algorithm 15, we need the two following lemmas due to Bhaskara et al. [2020].

Lemma A.1 (Bhaskara et al. [2020])**.** *Let $F : [K] \times [T] \to [T]$ be such that $F(i,t) \geq t$ for all $i \in [K], t \in [T]$ and $C : 2^{[K]} \times [T] \to R$ be a function that satisfies $C(\emptyset, t) = 0$ for all t, $C(S,T) = 1$ for all S, $C(S, T+1) = 0$ for all S, and C satisfies the recursion:*

$$C(S,t) = 1 + \frac{1}{|S|} \sum_{i\in S} C(S \setminus \{j \in S | F(j,t) \leq F(i,t)\}, F(i,t) + 1) \ .$$

Then $C(\{1,\dots,K\}, t) \leq \log_2(K+1)$ for all t.

Proof. We define the auxiliary function $Z(N) = \sup_{t,|S|\leq N} C(S,t)$. Observe $Z(0) = 0, Z(1) = 1$, and $Z(N)$ is non-decreasing with N. Now suppose for purposes of induction that $Z(n) \leq \log_2(n+1)$ for $n < N$. Then we have

$$Z(N) \leq 1 + \sup_{N' \leq N} \frac{1}{N'} \sup_{t,|S|=N'} \sum_{i\in S} C(S \setminus \{j \in S | F(j,t) \leq F(i,t)\}, F(i,t) + 1)$$

$$
\begin{aligned}
&\leq 1 + \sup_{N' \leq N} \frac{1}{N'} \sup_{t, |S|=N'} \sum_{i \in S} Z(N' - |\{j \in S | F(j,t) \leq F(i,t)\}|) \\
&\leq 1 + \sup_{N' \leq N} \frac{1}{N'} \sup_{t, |S|=N'} \sum_{i \in S} Z(N' - i) \\
&\leq 1 + \sup_{N' \leq N} \frac{1}{N'} \sup_{t, |S|=N'} \sum_{i \in S} Z(N' - i + 1) \\
&\leq 1 + \sup_{N' \leq N} \log_2 \left(\frac{1}{N'} \sum_{i=1}^{N'} (N' - i + 1) \right) \\
&\leq 1 + \sup_{N' \leq N} \log_2((N'+1)/2) \\
&= \log_2(N+1) \ ,
\end{aligned}
$$

where the third inequality is from Z(n) being non-decreasing in n and the fifth inequality is from Jensen inequality on $\log_2(\cdot)$. To conclude, note that clearly $C(\{1, \ldots, K\}, t) \leq Z(K)$ for all t. □

Theorem 6.11 (Bhaskara et al. [2020]). *Suppose $\mathcal{B}_1, \ldots, \mathcal{B}_K$ are deterministic OLO algorithms with monotone regret bounds $\mathcal{S}_1, \ldots, \mathcal{S}_K$. Given a constrained set V, suppose for all t, $\sup_{\boldsymbol{x}, \boldsymbol{y} \in V} \langle \boldsymbol{g}_t, \boldsymbol{x} - \boldsymbol{y} \rangle \leq 1$. Then for any fixed sequence of costs $\boldsymbol{g}_{1:T} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_T)$ and any $\boldsymbol{u} \in V$ (i.e. an oblivious adversary), Algorithm 15 guarantees:*

$$
\mathbb{E}[R_T(\boldsymbol{u})] \leq \log_2(K+1) \cdot \left(4 + 4 \min_{i \in [K]} S_{\mathcal{B}_i}([1,T], \boldsymbol{g}_{1:T}) \right) \ . \tag{6.18}
$$

Further, if the sequence of costs is allowed to depend on the algorithm's randomness (i.e., an adaptive adversary), then

$$
R_T(\boldsymbol{u}) \leq K \left(4 + 4 \min_{i \in [K]} S_{\mathcal{B}_i}([1,T], \boldsymbol{g}_{1:T}) \right) \ , \tag{6.19}
$$

Proof. We can divide the operations in phases where γ is constant. Furthermore, each phase can be divided into subphases where i is constant (i.e. the same *base* algorithm is running). The regret during a phase with fixed γ can be bounded as follows. Suppose there are N subphases (note that $N \leq K$). Let $t_1, \ldots, t_N$ denote the starting times of subphases $1, \ldots, N$, then we have that

$$
\max_{\boldsymbol{u} \in V} \sum_{t=t_1}^{t_{N+1}-1} \langle \boldsymbol{x}_t - \boldsymbol{u}, \boldsymbol{g}_t \rangle \leq \sum_{i=1}^{N} r^{i,\gamma}_{t_{i+1}-t_i} \leq N(\gamma + 1) \ ,
$$

where the last inequality derives from the that $r^{i,\gamma}_{t_{i+1}-1-t_i} \leq r^{i,\gamma}_{t_{i+1}-t_i} \leq \gamma + \max_{\boldsymbol{y}, \boldsymbol{x} \in V} \langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{g}_{t_{i+1}-1} \rangle \leq \gamma + 1$. Therefore we need to bound N, the number of subphases in each phase (note that N is a random variable).

Let $F(i,t)$ denote the index $\tau \geq t$ when the regret experienced by algorithm $\mathcal{A}_i$ that is initialized at time t first exceeds γ, and let $C(S,t)$ be the expected number of sub-phases (counting the current one) left in the phase if a subphase starts at time t with the specified set of active indices S. We define $C(S, T+1) = C(\emptyset, t) = 0$ for all S and t for notational convenience. Note that $C(S,T) = 1$ for all S (since we are in the last round). Further, by definition, we have $\mathbb{E}[N] = C(\{1, 2, \ldots, K\}, t)$ for some t (corresponding to the start of the phase). We can write the expected value as an expectation conditioned on the event that algorithm $\mathcal{A}_i$ is selected at time t from the set S, multiplied by the probability of selecting it.

Formally, we claim that C satisfies:

$$
C(S,t) = 1 + \frac{1}{|S|} \sum_{i \in S} C(S \setminus \{j \in S | F(j,t) \leq F(i,t)\}, F(i,t) + 1) \ .
$$

Algorithm 16 Sleeping Experts

Require: Expert algorithm $\mathcal{B}$
for $t = 1, \dots, T$ **do**
  Let $\tilde{\boldsymbol{p}}_t$ be the output of $\mathcal{B}$ in round t
  Observe $a_{t,i}$ for all $i = 1, \dots, d$
  Output $\boldsymbol{p}_t$, where $p_{t,i} \propto a_t(i) p_{t,i}$
  Observe $g_{t,i}$ for all i such that $a_t(i) = 1$
  Set $g_{t,i} = \langle \boldsymbol{p}_t, \boldsymbol{g}_t \rangle$ for all i such that $a_t(i) = 0$
  Pass $\boldsymbol{g}_t$ to $\mathcal{B}$
end for

To see this, observe that each index $i \in S$ is equally likely to be selected for the fixed i throughout the subphase starting at time t. By definition of F, the subphase will end at time $F(i,t)$ if the selected index is i. Further, at the end of the subphase, S will be $S \setminus \{j \in S | F(j,t) \leq F(i,t)\}$. Therefore, conditioned on selecting index i for this sub-phase, the expected number of sub-phases is $1 + C(S \setminus \{j \in S | F(j,t) \leq F(i,t)\}, F(i,t)+1)$. Since each index is selected with probability $1/|S|$, the stated identity follows. Now we apply Lemma A.1 to conclude that $C(\{1, \dots, K\}, t) \leq \log_2(K+1)$ for all t, which implies $\mathbb{E}[N] \leq \log_2(K+1)$.

Now we are in a position to calculate the total regret. Let $1 = T_1, \dots, T_P$ be the start times of the P phases, and let $T_P + 1 - 1 = T$ for notational convenience. Then, using Lemma 6.13 we have:

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} \langle \boldsymbol{x}_t, \boldsymbol{g}_t \rangle - \min_{\boldsymbol{u} \in V} \sum_{t=1}^{T} \langle \boldsymbol{u}, \boldsymbol{g}_t \rangle\right] &\leq \sum_{p=1}^{P} \mathbb{E}[N_p](\gamma_p + 1) \\
&\leq (2^P + P) \cdot \log_2(K+1) \\
&\leq \log_2(K+1) \cdot \left(4 + 4 \min_i \mathcal{S}_i([1,T], \boldsymbol{g}_{1:T})\right) .
\end{aligned}$$

To prove the second bound for an adaptive adversary, observe that in the worst-case, we cannot have more than K sub-phases in any phase. The rest of the argument is identical. □

A.3 Sleeping Experts

The sleeping experts setting was first studied in Freund et al. [1997]. In this setting, contrarily to the regular LEA problem some experts may be not active in all the time-steps. We can still use a regular experts algorithm $\mathcal{B}$. However, the probability distribution over the experts should be modified accordingly. This can be done rather easily, by renormalizing the probability distribution given by the algorithm $\mathcal{B}$, in order to take into account only active experts. To make things clearer, we introduce a variable $a_t(i) \in \{0, 1\}$ for all $i = 1, \dots, d$ denoting whether the expert i is active at time t or not. Therefore, if the probability vector given by $\mathcal{B}$ at round t is $\tilde{\boldsymbol{p}}_t$, then the probability for expert i at time t will be $p_{t,i} \propto a_t(i) \tilde{p}_{t,i}$.

Note that in the worst case the regret bound for the regular expert problem is $\mathcal{O}(\sqrt{T \ln d})$, where d is the number of experts. However, in case an expert is not active most of the time, a bound in terms of T is overly pessimistic. Ideally, for each expert i we would like a regret bound of

$$\mathcal{O}\left(\sqrt{\sum_{t=1}^{T} a_t(i) \ln d}\right) , \tag{A.1}$$

which is a function only of the timesteps when the expert is "awake".

Moreover, we did not discuss what loss to assign to experts when they are not active. In order to achieve the bound in Eq. (A.1), intuitively the algorithm should not be penalized against an expert i when the latter is not active. Therefore, if we assign to this expert the same loss as the algorithm, then the instantaneous regret against it will be 0. In this way, we

have

$$R_T(\boldsymbol{e}_i) = \sum_{t=1}^{T} a_t(i)(\langle \boldsymbol{p}_t, \boldsymbol{g}_t \rangle - g_{t,i}) = \sum_{t=1}^{T} (\langle \boldsymbol{p}_t, \boldsymbol{g}_t \rangle - g_{t,i}) = \sum_{t=1}^{T} (\langle \tilde{\boldsymbol{p}}_t, \boldsymbol{g}_t \rangle - \boldsymbol{g}_{t,i}) \ ,$$

which is the regret of algorithm $\mathcal{B}$ in the regular expert setting. Using the standard exponential weights algorithm for experts, we will incur a regret of $\mathcal{O}(\sqrt{T \ln d})$. On the other hand, there are algorithms which can achieve bounds order or $\mathcal{O}(\sqrt{\sum_{t=1} a_t(i)})$ such as *Squint* from Koolen and van Erven [2015] or the *coin-betting* algorithm from Orabona and Pal [2016] (details omitted).

A.4 Inequalities

Lemma A.2 (Young's inequality)**.** *If $x, y \in [0, \infty)$ and $p, q \in (1, \infty)$ are such that $1/p+1/q = 1$, then*

$$xy \leq \frac{x^p}{p} + \frac{y^q}{q} \ .$$

Proof. Note that $xy = (x^p)^{1/p}(y^q)^{1/q}$. Therefore applying the weighted inequality of arithmetic and geometric means we get

$$(x^p)^{1/p}(y^q)^{1/q} \leq \frac{x^p}{p} + \frac{y^q}{q} \ .$$

□

Theorem A.3 (Holder's inequality)**.** *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, and let $p, q \in (1, \infty)$ such that $1/p + 1/q = 1$. Then the following holds*

$$|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq \|\boldsymbol{x}\|_p \|\boldsymbol{y}\|_q \ .$$

Proof. By using Young's inequality we get

$$\begin{aligned} |\langle \boldsymbol{x}, \boldsymbol{y} \rangle| &\leq |x_1 y_1| + \cdots + |x_d y_d| \\ &\leq \frac{|x_1|^p + \cdots + |x_d|^p}{p} + \frac{|y_1|^q + \cdots + |y_d|^q}{q} \\ &= \frac{|\boldsymbol{x}|_p^p}{p} + \frac{|\boldsymbol{y}|_q^q}{q} \ . \end{aligned}$$

In particular, in the special case that $\|\boldsymbol{x}\|_p = \|\boldsymbol{y}\|_q = 1$ we have $|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq 1/p+1/q = 1$, which proves the desired result. In the general case, we may assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ are non-zero. Then $\boldsymbol{x}/\|\boldsymbol{x}\|_p$ and $\boldsymbol{y}/\|\boldsymbol{y}\|_q$ are unit vectors for their respective norms, and therefore

$$\left| \langle \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_p}, \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_q} \rangle \right| \leq 1 \ .$$

Multiplying both sides by $\|\boldsymbol{x}\|_p \|\boldsymbol{y}\|_q$ yields the desired result. □

Theorem A.4. *The dual norm of the p-norm is the q-norm, with $1/p + 1/q = 1$.*

Proof. Let $\boldsymbol{x} \in \mathbb{R}^d$, with $\|\boldsymbol{x}\|_q \leq 1$. By Holder's inequality we have that

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle \leq |\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq \|\boldsymbol{y}\|_p \|\boldsymbol{x}\|_q \leq \|\boldsymbol{y}\|_p \ .$$

On the other hand, to show that the supremum is exactly $\|\boldsymbol{y}\|_p$, it suffices to find a single $\boldsymbol{z} \in \mathbb{R}^d$ such that $\langle \boldsymbol{z}, \boldsymbol{y} \rangle = \|\boldsymbol{y}\|_p$. To this aim, let $\boldsymbol{x} \in \mathbb{R}^d$ such that $x_i \triangleq \text{sign}(y_i) y_i^{p-1}$ for all $i = 1, \ldots, d$. We have that

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^d y_i \cdot \text{sign}(y_i) |y_i|^{p-1} = \sum_{i=1}^d |y_i|^p = \|\boldsymbol{y}\|_p^p \ . \tag{A.2}$$

On the other hand, we have that

$$\|\boldsymbol{x}\|_q^q = \sum_{i=1}^d |x_i|^q = \sum_{i=1}^d |\text{sign}(y_i)|y_i|^{p-1}|^q = \sum_{i=1}^d |y_i|^p = \|\boldsymbol{y}\|_p^p \ , \tag{A.3}$$

where the second to last equality derives from the fact that we have $q(p-1) = p$ since $1/p + 1/q = 1$. Now, consider $\boldsymbol{z} \triangleq \boldsymbol{x}/\|\boldsymbol{x}\|_q$. By construction, we have that $\|\boldsymbol{z}\|_q = 1$ and

$$\langle \boldsymbol{y}, \boldsymbol{z} \rangle = \sum_{i=1}^d y_i \frac{x_i}{\|\boldsymbol{x}\|_q} = \frac{1}{\|\boldsymbol{x}\|_q} \sum_{i=1}^d y_i x_i \ .$$

Now, note that from Eq. (A.3) we have that $\|\boldsymbol{x}\|_q = (\|\boldsymbol{x}\|_q^q)^{1/q} = (\|\boldsymbol{y}\|_p^p)^{1/q} = \|\boldsymbol{y}\|_p^{p/q}$, while from Eq. (A.2) $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \|\boldsymbol{y}\|_p^p$. Therefore,

$$\frac{1}{\|\boldsymbol{x}\|_q} \sum_{i=1}^{d} y_i x_i = \frac{1}{\|\boldsymbol{y}\|_p^{p/q}} \|\boldsymbol{y}\|_p^p = \|\boldsymbol{y}\|_p^{p-p/q} = \|\boldsymbol{y}\|_p \ ,$$

where the last step derives from the fact that $1/p + 1/q = 1$ implies $p - p/q = p(1 - 1/q) = p(1/p) = 1$. Therefore, there's a $\boldsymbol{z} \in \mathbb{R}^d$ with $\|\boldsymbol{z}\|_q = 1$ such that $\langle \boldsymbol{z}, \boldsymbol{y} \rangle = \|\boldsymbol{y}\|_p$ as desired, completing the proof.

□

Tsallis Entropy. The *Tsallis*-entropy is a generalization of the *Shannon* entropy:

$$\psi_q(\boldsymbol{x}) = \frac{1}{1-q} \left(1 - \sum_{i=1}^{d} x_i^q \right) \ , \tag{A.4}$$

for any $q \in [0, 1]$.

It can be seen that the *Shannon* entropy can be recovered in the limit of q going to 1

$$\lim_{q \to 1} \frac{1}{1-q} \left(1 - \sum_{i=1}^{d} x_i^q \right) = \sum_{i=1}^{d} x_i \ln x_i \ .$$

Proof. We begin by writing $\psi_q(\boldsymbol{x}) = f(q)/g(q)$, where $f(q) = 1 - \sum_{i=1}^{d} x_i^q$ and $g(q) = 1 - q$. Now, observe that $h(q) = x_i^q = \exp(q \ln x_i)$ and $h'(q) = x_i^q \ln x_i$, thus $f'(q) = -\sum_{i=1}^{d} x_i^q \ln x_i$. Also, $g'(q) = -1$ and then, by using *de l'Hopital* rule $\lim_{x \to 1} f(q)/g(q) = \sum_{i=1}^{d} x_i \ln x_i$. □

Appendix B

Omitted Experiments

B.1 AdaImplicit

Formulas. First, let's mention the update rules for IOMD with $V = \mathbb{R}^d$ and $\psi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$ for hinge loss, absolute loss, and square loss respectively [see, e.g., Crammer et al., 2006; Kulis and Bartlett, 2010].

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \min\left(\eta_t, \frac{\max(1 - y_t\langle \boldsymbol{z}_t, \boldsymbol{x}_t\rangle, 0)}{\|\boldsymbol{z}_t\|^2}\right) y_t \boldsymbol{z}_t,$$
$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \min\left(\eta_t, \frac{|\langle \boldsymbol{z}_t, \boldsymbol{x}_t\rangle - y_t|}{\|\boldsymbol{z}_t\|^2}\right) \boldsymbol{z}_t,$$
$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \frac{(\langle \boldsymbol{z}_t, \boldsymbol{x}_t\rangle - y_t)\boldsymbol{z}_t}{1 + \eta\|\boldsymbol{z}_t\|_2^2} \ .$$

Now let's consider the case that $V = \{\boldsymbol{x} : \|\boldsymbol{x}\|_2 \leq D/2\}$. In this case it is easy to see that the update becomes

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \boldsymbol{g}'_t - \alpha \boldsymbol{x}_{t+1},$$

where $\boldsymbol{g}'_t \in \partial \ell_t(\boldsymbol{x}_{t+1})$, $\alpha \boldsymbol{x}_{t+1} \in \partial i_V(\boldsymbol{x}_{t+1})$ and $\alpha \geq 0$. Hence, we have that

$$\boldsymbol{x}_{t+1} = \frac{\boldsymbol{x}_t}{\alpha + 1} - \frac{\eta_t}{\alpha + 1}\boldsymbol{g}'_t \ .$$

This implies that we can take the previous formulas and substitute $\frac{\boldsymbol{x}_t}{\alpha+1}$ to $\boldsymbol{x}_t$ and $\frac{\eta_t}{\alpha+1}$ to η_t. Then, the optimal $\alpha \geq 0$ is the smallest one that gives $\|\boldsymbol{x}_{t+1}\| \leq D/2$. Note that this is a 1-dimensional problem that can be easily solved numerically.

Results. In Fig. B.1 we show plots about other experiments on real data which were not shown in Chapter 4. Details about the datasets used can be found in Table B.1 and Table B.2.

Table B.1: Classification datasets

Name	Datapoints	Features
a9a	32,561	123
ijcnn1	49,990	22
cod-rna	59,535	8
covtype	581,012	54
skin_nonskin	245,057	3
phishing	11,055	68

Table B.2: Regression datasets

Name	Datapoints	Features
abalone	11,055	8
cadata	20,640	8
cpusmall	8,192	12
housing	506	13
space_ga	3,107	6

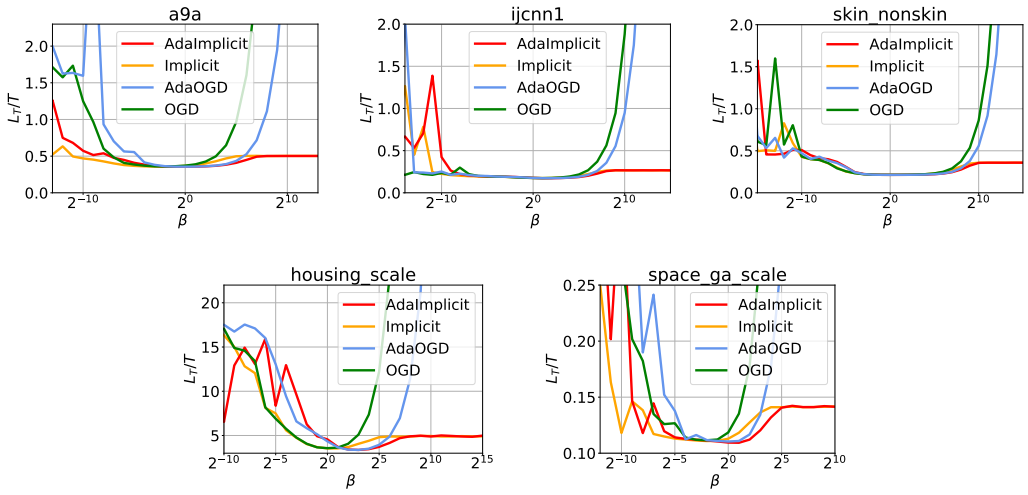


FIGURE B.1: Plots on classification tasks using the hinge loss (top) and regression tasks using the absolute loss (bottom).

Bibliography

A. Agarwal, H. Luo, B. Neyshabur, and R. E. Schapire. Corralling a band of bandit algorithms. In *Conference on Learning Theory*, pages 12–38. PMLR, 2017.

Z. Allen-Zhu, Z. Liao, and L. Orecchia. Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 237–245, 2015.

S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 227–236, 2007.

H. Asi and J. C. Duchi. Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity. *SIAM Journal on Optimization*, 29(3):2257–2290, 2019.

P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *J. Comput. Syst. Sci.*, 64(1):48–75, 2002.

B. Barak, M. Hardt, and S. Kale. The uniform hardcore lemma via approximate bregman projections. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1193–1200. SIAM, 2009.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

O. Besbes, Y. Gur, and A. Zeevi. Non-stationary stochastic optimization. *Operations research*, 63(5):1227–1244, 2015.

Aditya Bhaskara, Ashok Cutkosky, Ravi Kumar, and Manish Purohit. Online linear optimization with many hints. *arXiv preprint arXiv:2010.03082*, 2020.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

N. Campolongo and F. Orabona. Temporal variability in implicit online learning. *Advances in Neural Information Processing Systems*, 33, 2020.

N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, 1997.

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

Nicolò Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. In *Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings*, volume 3559, pages 217–232. Springer, 2005.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Kamalika Chaudhuri, Yoav Freund, and Daniel J Hsu. A parameter-free hedging algorithm. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 297–305. Curran Associates, Inc., 2009.

L. Cheng, S. V. N. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli. Implicit online learning with kernels. In *Advances in Neural Information Processing Systems 19*, pages 249–256, 2007.

C.-K. Chiang, T. Yang, C.-J. Lee, M. Mahdavi, C.-J. Lu, R. Jin, and S. Zhu. Online optimization with gradual variations. In *Proc. of the Conference on Learning Theory (COLT)*, volume 23, pages 6.1–6.20, 2012.

P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S. Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 273–282, 2011.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Ashok Cutkosky. Parameter-free, dynamic, and strongly-adaptive online learning. In *International Conference on Machine Learning*, volume 2, 2020.

Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *International Conference on Machine Learning*, pages 1405–1411, 2015.

S. de Rooij, T. van Erven, P. D. Grünwald, and W. M. Koolen. Follow the leader if you can, hedge if you must. *Journal of Machine Learning Research*, 15(37):1281–1316, 2014.

D. Drusvyatskiy. Convex analysis and nonsmooth optimization. *Lecture Notes*, 2020.

John C Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *COLT*, pages 14–26. Citeseer, 2010.

H. Fang, N. Harvey, V. S. Portella, and M. P. Friedlander. Online mirror descent and dual averaging: keeping pace in the dynamic case. *arXiv preprint arXiv:2006.02585*, 2020.

Dylan J Foster, Akshay Krishnamurthy, and Haipeng Luo. Open problem: Model selection for contextual bandits. In *Conference on Learning Theory*, pages 3842–3846. PMLR, 2020.

Yoav Freund. Open problem: Second order regret bounds based on scaling time. In *Conference on Learning Theory*, pages 1651–1654, 2016.

Yoav Freund, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. Using and combining predictors that specialize. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 334–343, 1997.

Pierre Gaillard, Gilles Stoltz, and Tim Van Erven. A second-order bound with excess losses. In *Conference on Learning Theory*, pages 176–196, 2014.

Eric Hall and Rebecca Willett. Dynamical models and tracking regret in online convex programming. In *International Conference on Machine Learning*, pages 579–587, 2013.

E. Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3–4):157–325, 2016.

Elad Hazan and Comandur Seshadhri. Adaptive algorithms for online decision problems. In *Electronic colloquium on computational complexity (ECCC)*, volume 14, 2007.

S. Hopkins, J. Li, and F. Zhang. Robust and heavy-tailed mean estimation made simple, via regret minimization. *Advances in Neural Information Processing Systems*, 33, 2020.

A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan. Online optimization: Competing with dynamic comparators. In *Artificial Intelligence and Statistics*, pages 398–406, 2015.

Kwang-Sung Jun, Francesco Orabona, Stephen Wright, and Rebecca Willett. Improved strongly adaptive online learning using coin betting. In *Artificial Intelligence and Statistics*, pages 943–951. PMLR, 2017.

N. Karampatziakis and J. Langford. Online importance weight aware updates. In *Proc. of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, pages 392—-399, Arlington, Virginia, USA, 2011. AUAI Press.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, January 1997.

W. M. Koolen and T. van Erven. Second-order quantile methods for experts and combinatorial games. In *Proc. of the Conference On Learning Theory (COLT)*, pages 1155–1175, 2015.

Wouter M Koolen and Tim Van Erven. Second-order quantile methods for experts and combinatorial games. In *Conference on Learning Theory*, pages 1155–1175, 2015.

B. Kulis and P. L. Bartlett. Implicit online learning. In *International Conference on Machine Learning*, pages 575–582, 2010.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

H. Luo. Introduction to online learning, lecture 4. *Lecture Notes*, 2017.

B. Martinet. Régularisation d'inéquations variationnelles par approximations successives. rev. française informat. *Recherche Opérationnelle*, 4:154–158, 1970.

H. B. McMahan. A unified view of regularized dual averaging and mirror descent with implicit updates. *arXiv preprint arXiv:1009.3240*, 2010.

H. B. McMahan and M. J. Streeter. Adaptive bound optimization for online convex optimization. In *COLT*, 2010.

H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica. Ad click prediction: a view from the trenches. In *Proc. of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

V. Mirrokni, R. P. Leme, A. Vladu, and S. Wong. Tight bounds for approximate carathéodory and beyond. In *International Conference on Machine Learning*, pages 2440–2448. PMLR, 2017.

Aryan Mokhtari, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro. Online optimization in dynamic environments: Improved regret rates for strongly convex problems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 7195–7201. IEEE, 2016.

J.-J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2004.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

F. Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

F. Orabona and D. Pál. Scale-free algorithms for online linear optimization. In *International Conference on Algorithmic Learning Theory*, pages 287–301. Springer, 2015.

F. Orabona and D. Pal. Coin betting and parameter-free online learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 577–585. Curran Associates, Inc., 2016.

F. Orabona and D. Pál. Scale-free online learning. *Theoretical Computer Science*, 716:50–69, 2018. Special Issue on ALT 2015.

Francesco Orabona and Dávid Pál. Optimal non-asymptotic lower bound on the minimax regret of learning with expert advice. *CoRR*, abs/1511.02176, 2015.

N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3): 127–239, 2014.

A. Rakhlin and K. Sridharan. Online learning with predictable sequences. In *Proc. of the Conference on Learning Theory (COLT)*, volume 30, pages 993–1019, 2013.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

Amir Sani, Gergely Neu, and Alessandro Lazaric. Exploiting easy data in online optimization. In *Advances in Neural Information Processing Systems*, pages 810–818, 2014.

S. Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University, 2007.

S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2), 2012.

S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7(Jul):1567–1599, 2006.

C. Song, J. Liu, H. Liu, Y. Jiang, and T. Zhang. Fully implicit online learning. *arXiv preprint arXiv:1809.09350*, 2018.

M. Streeter and B. McMahan. No-regret algorithms for unconstrained online convex optimization. In *Advances in Neural Information Processing Systems 25*, pages 2402–2410. Curran Associates, Inc., 2012.

P. Toulis and E. M. Airoldi. Asymptotic and finite-sample properties of estimators based on stochastic gradients. *The Annals of Statistics*, 45(4):1694–1727, 2017.

P. Toulis, E. M. Airoldi, and J. Rennie. Statistical analysis of stochastic gradient methods for generalized linear models. In *International Conference on Machine Learning*, pages 667–675, 2014.

M. K. Warmuth and A. K. Jagota. Continuous and discrete-time nonlinear gradient descent: Relative loss bounds and convergence. In *Electronic proceedings of the 5th International Symposium on Artificial Intelligence and Mathematics*, 1997.

T. Yang, L. Zhang, R. Jin, and J. Yi. Tracking slowly moving clairvoyant: Optimal dynamic regret of online learning with true and noisy gradient. In *International Conference on Machine Learning*, pages 449–457, 2016.

Lijun Zhang, Tianbao Yang, Jinfeng Yi, Rong Jin, and Zhi-Hua Zhou. Improved dynamic regret for non-degenerate functions. In *Advances in Neural Information Processing Systems*, pages 732–741, 2017.

Lijun Zhang, Shiyin Lu, and Zhi-Hua Zhou. Adaptive online learning in dynamic environments. In *Advances in neural information processing systems*, pages 1323–1333, 2018a.

Lijun Zhang, Tianbao Yang, Rong Jin, and Zhi-Hua Zhou. Dynamic regret of strongly adaptive methods. In *International Conference on Machine Learning*, pages 5882–5891, 2018b.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proc. of ICML*, pages 928–936, 2003.