BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

TEMPORAL ASPECTS OF ADAPTIVE ONLINE LEARNING: CONTINUITY AND REPRESENTATION

by

ZHIYU ZHANG

B.Eng., Tsinghua University, 2018

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2023

Approved by

First Reader

Ioannis Ch. Paschalidis, PhD
Distinguished Professor of Engineering
Professor of Electrical and Computer Engineering
Professor of Biomedical Engineering
Professor of Systems Engineering
Professor of Computing and Data Sciences

Second Reader

Ashok Cutkosky, PhD
Assistant Professor of Electrical and Computer Engineering
Assistant Professor of Computer Science
Assistant Professor of Systems Engineering
Assistant Professor of Computing and Data Sciences

Third Reader

Alexander Olshevsky, PhD
Associate Professor of Electrical and Computer Engineering
Associate Professor of Systems Engineering

Fourth Reader

Kayhan Batmanghelich, PhD
Assistant Professor of Electrical and Computer Engineering

Acknowledgments

First of all, I am deeply grateful to my advisors, Prof. Yannis Paschalidis and Prof. Ashok Cutkosky. It is my great fortune to learn and grow under their guidance.

Yannis led me into the field of machine learning and guided me through my initial stage of exploration. Even though my research interests gradually deviated from the main focus of the group, he graciously provided me with all the support I could hope for. Especially, he gave me tremendous freedom and trust to pursue curiosity-driven research, which forms the foundation of this dissertation. It shows how amazing a supportive environment can be, and I will try to carry this spirit through my future career.

A turning point of my exploration was the end of my second year, when Ashok joined BU, and my research pivoted to online learning. Ashok is a brilliant researcher in so many ways, and working with him has been a humbling and incredibly rewarding experience. Besides the knowledge I learned from him, I was also deeply inspired by his unceasing passion in research, which brought me a lot of energy and motivation to overcome the many obstacles on this journey. Although it is probably hard to achieve, being a scholar like Ashok will be a goal I strive for in the future.

I am also very grateful to other faculty members at BU: Prof. Christos Cassandras helped me navigating the first year when I was clueless. Prof. Francesco Orabona taught a fantastic course on online learning that sparked my interest in the first place. Prof. Francesco Orabona, Prof. Alex Olshevsky and Prof. Kayhan Batmanghelich kindly served on my committee at different stages. Especially, I am grateful to Prof. Alex Olshevsky for his trust and support through my postdoc application.

Outside research, these five years have been more challenging than I expected at the beginning. I sincerely thank my labmates and friends for all the support and

countless fun times, which made my whole experience so memorable. It is hard to list everyone here, but for those who happen to read this, thank you so much! I will cherish the memories forever.

Finally, I am grateful to my parents for their unconditional love. I know that they are with me wherever I go.

TEMPORAL ASPECTS OF ADAPTIVE ONLINE LEARNING: CONTINUITY AND REPRESENTATION

ZHIYU ZHANG

Boston University, College of Engineering, 2023

Major Professors: Ioannis Ch. Paschalidis, PhD
Distinguished Professor of Engineering
Professor of Electrical and Computer Engineering
Professor of Biomedical Engineering
Professor of Systems Engineering
Professor of Computing and Data Sciences

Ashok Cutkosky, PhD
Assistant Professor of Electrical and Computer Engineering
Assistant Professor of Computer Science
Assistant Professor of Systems Engineering
Assistant Professor of Computing and Data Sciences

ABSTRACT

Adaptive online learning, in a very broad sense, is the study of sequential decision making beyond the worst case. Compared to their classical minimax counterparts, adaptive algorithms typically require less manual tuning, while provably performing better in benign environments, or with prior knowledge. This dissertation presents new techniques for designing these algorithms. The central theme is the emphasis on the temporal nature of the problem, which has not received enough attention in the literature.

The first part of the dissertation focuses on temporal continuity. While modern online learning almost exclusively studies a discrete time repeated game, it is shown that designing algorithms can be simplified, and in certain cases optimized, by scaling the game towards a continuous time limit and solving the obtained differential equation. Concretely, we develop comparator adaptive algorithms for Online Convex Optimization, achieving optimal static regret bounds in the vanilla setting and its variant with switching costs. The benefits are extended to another classical online learning problem called Learning with Expert Advice.

The second part of the dissertation focuses on temporal representation. Different from the first part, here we consider the general objective of dynamic regret minimization, which forms the foundation of time series forecasting. It is shown that by introducing temporal features, the task can be transformed to static regret minimization on a user-specified representation space with growing dimension. Drawing novel connections to wavelet features, we develop a simple algorithm improving the state-of-the-art dynamic regret bound achieved by more sophisticated approaches. An application is the online fine-tuning of a black-box time series forecaster.

Contents

List of Tables

List of Figures

List of Abbreviations

BAH	.....	Buy-and-Hold (investment strategy)
BHE	.....	Backward Heat Equation
CDF	.....	Cumulative Distribution Function
DFT	.....	Discrete Fourier Transform
EG	.....	Exponentiated Gradient
FTL	.....	Follow the Leader
FTRL	.....	Follow the Regularized Leader
GC	.....	Geometric Covering
IID	.....	Independent and Identically Distributed
KL	.....	Kullback–Leibler (divergence)
KT	.....	Krichevsky-Trofimov (algorithm)
LEA	.....	Learning with Expert Advice
LHS	.....	Left Hand Side
ML	.....	Machine Learning
MRA	.....	Multi-Resolution Analysis
OCO	.....	Online Convex Optimization
ODE	.....	Ordinary Differential Equation
OGD	.....	Online Gradient Descent
OLO	.....	Online Linear Optimization
OLR	.....	Online Linear Regression
OMD	.....	Online Mirror Descent
PDE	.....	Partial Differential Equation
RHS	.....	Right Hand Side
VAW	.....	Vovk-Azoury-Warmuth (forecaster)

Chapter 1

Introduction

The study of *Machine Learning* (ML) originated from the wildest imagination of human beings. It is hoped that someday, we might be able to create an intelligent machine that can think, respond, and distill knowledge like us. While the exact definition of intelligence is still not settled, machine learning as a serious research field has thrived under a more concrete, although still challenging objective – creating programs that improve themselves by observing the environment, rather than only relying on hard-coded knowledge. There are numerous success stories fitting into this narrative. For example, by recording the observations into large datasets (Deng et al., 2009), one could train highly capable deep learning models for image classification (Krizhevsky et al., 2017). Another example is reinforcement learning, where the learning agent directly interacts with a stateful environment through trial-and-error; this has powered remarkable advances in game playing (Silver et al., 2016), protein structure prediction (Jumper et al., 2021) and automatic question answering (Brown et al., 2020).

Despite the diversity of these learning problems, one could view them as variants of an abstract, discrete time sequential decision making process. In each round, the learning agent picks a decision without knowing the environment's choice; and then, the environment reveals a feedback, which also measures the quality of the agent's decision. It is clear that sequential learning problems, such as reinforcement learning, are special cases of such a process with certain structured environment and

feedback protocol. The classical batch learning problems are also recovered when the standard *independent-and-identically-distributed* (IID) condition is imposed on the environment – in fact, even though the entire dataset is available without explicit sequential interactions, the common practice for training deep learning models is splitting the dataset into mini-batches and processing them sequentially, due to the typically gigantic size of modern datasets. Therefore, unlocking the full potential of machine learning requires a deeper understanding of sequential decision making, which adds to the importance of this classical research field.

A lot has been done on the sequential decision making problems motivated by machine learning. Depending on the level of abstraction, plenty of interaction models have been proposed, each with its own value. Specifically, a significant amount of research effort has converged to a setting called *Online Convex Optimization* (OCO), which is the focus of this dissertation. The basic setting (Zinkevich, 2003) is the following, while variants of this setting will be introduced later when needed. It is so popular that the research community sometimes equates it to *online learning*, which ought to be a broader concept. For the ease of exposition, we will also adopt this somewhat inappropriate terminology.

Definition 1 (OCO)**.** *Online Convex Optimization is a repeated game between a learning agent and an adversarial environment denoted by Env. In each (the t-th) round,*

1. *The agent makes a prediction $x_t \in \mathcal{X}$ based on the observations before the t-th round. $\mathcal{X}$ is a nonempty, closed and convex subset of $\mathbb{R}^d$.*
2. *The environment reveals a convex loss function $l_t : \mathcal{X} \to \mathbb{R}$, which depends deterministically on the agent's prediction history $x_1, \ldots, x_t$.*
3. *The agent suffers the loss $l_t(x_t)$.*

The game ends after T rounds, and then, the total loss of the agent is compared to that of an alternative sequence of predictions $u_1, u_2, \ldots, \in \mathcal{X}$, called a comparator sequence.

Without knowing the environment Env and the comparator sequence $\{u_t\}_{t\in\mathbb{Z}}$*, the goal of the agent is to achieve low regret, defined as*[1]

$$\text{Regret}_T(Env, u_{1:T}) := \sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u_t). \tag{1.1}$$

If this expression is at most a sublinear function of T*, then asymptotically (as* $T \to \infty$*), the average loss of the agent is at most that of the comparator sequence* $\{u_t\}_{t\in\mathbb{Z}}$*.*

Since Env and $\{u_t\}_{t\in\mathbb{Z}}$ *are hidden before predictions are made, the agent's strategy cannot depend on them. This motivates a robust objective – achieving sublinear regret against a large class of environments and comparator sequences. The choice of such classes will be made clear as we proceed.*

It is immediately noticeable that there are no statistical assumptions on the environment, which is in stark contrast to a vast body of research based on, for example, Gaussian modeling. Yet, the comparator sequence is selected after all the loss functions are revealed, which may seem unfair: how is it possible to beat a benchmark that has hindsight about the future? Indeed, this is a valid concern, and as a small hint, adaptive online learning will be crucial for characterizing what can be done. For now, we introduce the conventional perspective relying on restricting the comparator class – an important and better studied special case is the setting with fixed comparators.

Definition 2 (Static and dynamic regret)**.** *If the comparator sequence* $\{u_t\}_{t\in\mathbb{Z}}$ *satisfies* $u_t = u$ *for some* $u \in \mathcal{X}$*, then we define*

$$\text{Regret}_T(Env, u) := \sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u). \tag{1.2}$$

Following the convention of the field, (1.1) is called the dynamic regret, while (1.2) is called the static regret.

Intuitively, the problem is simpler, or more fair for the agent, when the static regret is used as the performance metric: even though the comparator can still be

[1] $u_{1:T}$ denotes the tuple $[u_1, \ldots, u_T]$.

chosen after observing all the loss functions, it has to be time-invariant, while the agent's prediction can change over time. Both the simpler static regret and the more general dynamic regret will be considered in this dissertation.

Why is OCO an interesting setting? A possible explanation is that, it is general enough to cover a wide range of downstream applications, while still being mathematically tractable. Here, we list three examples to contextualize its abstract setup. They include not only the narrowly defined "ML training" tasks, but also classical research topics from disparate fields.

- **Training ML models.** Taking linear regression as an example, suppose we are given a collection of IID covariate-label pairs, $(z_t, y_t) \in \mathbb{R}^d \times \mathbb{R}$, sampled from an unknown distribution. A linear model parameterized by x is the function[2] $f(z; x) = \langle z, x \rangle$, which maps a covariate z to a predicted label in $\mathbb{R}$, and induces the square loss $(\langle z, x \rangle - y)^2$ with respect to the true label y. Training the model amounts to finding a coefficient x^* that minimizes the risk $\mathbb{E}_{(z,y)}[(\langle z, x \rangle - y)^2]$, and by changing the model and the loss function, the same task can be defined beyond linear regression.

  Converting the above into the OCO framework, we obtain the following sequential training procedure. The learning agent's prediction x_t is the estimated coefficient of the linear model after observing $t-1$ samples, and the loss function l_t in OCO measures the quality of x_t in fitting the t-th sample, i.e., $l_t(x) = (\langle z_t, x \rangle - y_t)^2$. Applying a suitable OCO algorithm, the static regret (1.2) translates to

$$\sum_{t=1}^{T} (\langle z_t, x_t \rangle - y_t)^2 = \sum_{t=1}^{T} (\langle z_t, u \rangle - y_t)^2 + \text{Regret}_T(Env, u),$$

  where the regret term on the RHS can be sublinear in T regardless of the environment Env and the comparator u. Picking an optimal coefficient x^* as

[2] $\langle \cdot, \cdot \rangle$ denotes the inner product.

the comparator u, it means that (the average of) the LHS, which quantifies the performance of training, is asymptotically no greater than the empirical risk of the optimal coefficient. In other words, the training is successful.[3]

- **Weather forecasting.** The previous example demonstrates the use of static regret bounds, which is motivated by the inductive bias that there exists a time-invariant prediction x^* with low cumulative loss. This is indeed true in ML training, as we assume samples are IID, and implicitly, the relation between the covariate and the label can be well summarized by a linear model. Because of this, the static regret (1.2) is the most common performance metric for OCO, but there are also situations where this inductive bias fails, and we are required to consider the dynamic regret (1.1).

  A good example is time series forecasting. Suppose we want to forecast the daily temperature. In the OCO framework, the agent's prediction $x_t \in \mathbb{R}$ is our guess of the next day's temperature after $t-1$ days have passed. Then, the nature reveals a true temperature $x_t^* \in \mathbb{R}$, which induces a loss $f(x_t; x_t^*)$ on the agent based on a fixed function f; for instance, $f(x_t; x_t^*)$ can be the absolute error $|x_t - x_t^*|$ or the square error $(x_t - x_t^*)^2$. A natural candidate for the comparator sequence $\{u_t\}_{t\in\mathbb{Z}}$ is the true temperature sequence $\{x_t^*\}_{t\in\mathbb{Z}}$, and in that case, the cumulative forecasting loss is simply the dynamic regret, i.e.,

$$\sum_{t=1}^{T} f(x_t; x_t^*) = \mathrm{Regret}_T(Env, x_{1:T}^*).$$

  That is, upper-bounding this dynamic regret leads to guaranteed forecasting performance.

[3]Technically, we need the online-to-batch conversion (or simply, concentration inequalities) to relate the final step to the risk, which is standard (Orabona, 2019) and beyond the scope of this dissertation.

- **Sequential investment.** The third example aims to show that in the OCO framework, although the learning agent's decision is often called a "prediction", it does not necessarily mean that there is a ground truth that the agent needs to estimate.

  Consider the following sequential investment task: the agent's prediction $x_t \in \mathbb{R}$ is the amount of a particular stock that it holds at the start of the t-th day. Then, the environment reveals the price change $c_t \in \mathbb{R}$ of the stock, and the agent's total wealth increases by $c_t x_t$. If the agent doubles its prediction x_t, then the return also doubles, which could be either positive or negative. Therefore, there does not exist an optimal investment amount in hindsight, which is in contrast to the weather forecasting example above. A sublinear dynamic regret bound (1.1) means that asymptotically, the per-round return of the agent approaches that of the comparator investment strategy, i.e., holding the amount u_t on the t-th day.

  Clearly, no algorithm can guarantee sublinear dynamic regret against all comparator sequences, since otherwise there exists an investment strategy that always wins regardless of the environment. Nonetheless, if we only care about matching the performance of a smaller class of benchmark investment strategies, then it is possible to guarantee sublinear dynamic regret against such a class of restricted comparators.

Summarizing the above, we hope to convince the reader that OCO is a useful model worthy of detailed investigation. Moreover, it is also hinted that a considerable amount of subtleties exist in its seemingly innocent setup. For more motivation of the online learning problem, the readers are referred to excellent textbooks including (Cesa-Bianchi and Lugosi, 2006; Rakhlin and Sridharan, 2014b; Hazan, 2016; Orabona, 2019).

Next, we will concretely introduce different ways to solve OCO. The standard minimax optimal algorithms are surveyed in Section 1.1, which naturally motivates their adaptive counterparts (Section 1.2), as well as the important Bayesian interpretation of adaptive algorithms (Section 1.3). Common types of adaptive online learning are introduced in Section 1.4. Section 1.5 presents an overview of this dissertation, including the main contributions. Notations are introduced in Section 1.6.

1.1 Minimax online learning

OCO is an optimization problem, so it is natural that the geometry of the loss functions determines the achievable convergence rates. Throughout this dissertation, we will assume Lipschitzness on the loss functions, with respect to the Euclidean norm unless specified otherwise.

Assumption 1. *For all t, the loss function l_t selected by the environment is G-Lipschitz with respect to the Euclidean norm. G is known by the agent before the OCO game starts.*

Assumption 1 poses a limitation on the class of the adversarial environment we consider. This is standard in the literature, and also the only essential restriction we impose on top of the basic OCO setup. Compared to other structural assumptions on the loss functions (e.g., strong convexity or smoothness), the problem with Lipschitzness alone is in general the hardest (in terms of the achievable bound rather than the difficulty of the analysis). Without guaranteed curvatures, a typical procedure of the agent is to linearize the loss functions, turning OCO to a special case called *Online Linear Optimization* (OLO).

Definition 3 (OLO)**.** *Online Linear Optimization is the following special case of OCO: after picking the convex loss function l_t, instead of revealing it to the agent, the environment reveals a subgradient $g_t \in \partial l_t(x_t)$. The agent treats the surrogate linear loss $\langle g_t, \cdot \rangle$ as the loss function it observes.*

Due to convexity, we have for all $u \in \mathcal{X}$,

$$l_t(x_t) - l_t(u) \leq \langle g_t, x_t - u \rangle .$$

Therefore, it suffices to consider OLO instead of OCO without loss of generality – regret upper bounds of OLO are also regret upper bounds of OCO. This reduction is typically optimal when no additional structure beyond convexity is assumed.

The most standard algorithm for OLO is *Online Gradient Descent* (OGD), which was proposed in (Zinkevich, 2003) alongside the first OCO/OLO formulation. The algorithm has a very simple procedure: after observing the subgradient g_t, the next prediction is generated through a (Euclidean-projected) gradient step $x_{t+1} = \Pi_{\mathcal{X}}(x_t - \eta_t g_t)$. Suppose the learning rate is fixed ($\eta_t = \eta$), then regardless of the environment Env and the time-invariant comparator $u \in \mathcal{X}$, the static regret of OGD is upper bounded by (Orabona, 2019, Theorem 2.13)

$$\mathrm{Regret}_T(Env, u) \leq \frac{\|u - x_1\|_2^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|g_t\|_2^2 .$$

Recall that due to Assumption 1, $\|g_t\|_2 \leq G$. As for the first term on the RHS, let us consider only a subset $\mathcal{U}$ of comparators satisfying $\|u - x_1\|_2 \leq D$.[4] Then, with a known T, it is clear that the above regret bound is minimized when $\eta \propto DG^{-1}T^{-1/2}$, leading to $\sup_{Env,u} \mathrm{Regret}_T(Env, u) = O(DG\sqrt{T})$, which is a sublinear function of T. A time-varying learning rate can further relax the knowledge of the time horizon T, while achieving the same bound.[5]

The above tuning of the learning rate has a *minimax* flavor: implicitly, it is assumed

[4] In the language of online learning, the problem is *improper*, in the sense that the agent can operate on a larger domain than the comparator. It becomes *proper* if for all $x \in \mathcal{X}$, $\|x - x_1\|_2 \leq D$.

[5] In principle, this "anytime" result is actually a type of *adaptive* online learning with respect to the time horizon T. However, it is so simple and practically necessary that the community nowadays often takes this adaptivity for granted in the study of minimax algorithms. See Section 1.4 for a detailed discussion.

that the environment and the comparator are the hardest (worst) case possible, i.e., $\|g_t\|_2 = G$ and $\|u - x_1\|_2 = D$; and then, a suitable tuning of η ensures that even in this case, the regret is still sublinear in T. In other words, the design of the algorithm aims to minimize the worst case regret

$$\sup_{Env,u} \text{Regret}_T(Env, u), \tag{1.3}$$

therefore we call such algorithms "minimax algorithms".

In a similar spirit, the $O(DG\sqrt{T})$ static regret bound is known to be *minimax optimal* (Orabona, 2019, Section 5.1): for any OCO algorithm and any positive integer T, there exist an environment Env and a static comparator u satisfying all the aforementioned assumptions (including $\|g_t\|_2 \leq G$ and $\|u - x_1\|_2 \leq D$), such that $\text{Regret}_T(Env, u) = \Omega(DG\sqrt{T})$. Such minimax optimality is a key reason behind the popularity of (stochastic) gradient descent.

As one would expect, OGD is not the only minimax online learning algorithm (Hazan, 2016; Orabona, 2019). It can be generalized to *Online Mirror Descent* (OMD), which uses the geometry of the domain to design better incremental steps. In parallel, there are frameworks such as *Follow the Regularized Leader* (FTRL) and the *potential framework*, which use *dual space* updates rather than incremental steps. These general frameworks are not necessarily minimax by default, but their common instances are indeed minimax, sharing a very similar reasoning as OGD above. For the ease of exposition, we will primarily use OGD (or alternatively, "gradient descent") as the "minimax baseline" throughout this dissertation.

To summarize, focusing on the static regret, we loosely define algorithms that minimize the worst case regret (1.3) as *minimax algorithms*. More rigorously, one could define *minimax optimal regret bounds* in a standard game-theoretic manner. Pondering these concepts, there are a few obvious limitations. For example, the well-posedness of

the worst case regret (1.3) originated from a restriction on the comparator class. There are settings of the comparator class $\mathcal{U}$ such that the worst case regret of *any* algorithm is infinite or trivial. Furthermore, the problem instances we face in practice are usually not the worst case, therefore a minimax algorithm could be overly conservative, and the minimax optimal regret bound could be unnecessarily loose. Resolving these issues led to the decade-long study of adaptive online learning.

1.2 Adaptive online learning

"Adaptivity" is quite a common terminology in the literature, but a consensus on its meaning has not been reached. In control theory, there is the concept of "adaptive control", while numerous deep learning optimizers have been proposed with "adaptive" in their names. However, they do not mean the same as the idea of adaptive online learning, and even the latter lacks a universal definition within the community. Actually, to convey its main idea, the clearest presentation we found is from statistics (Johnstone, 2019, Section 6), which is a discussion on *adaptive statistical estimators*:

> An estimator that is exactly minimax for a given parameter set will depend, often quite strongly, on the details of that parameter set. While this is informative about the effect of assumptions on estimators, it is impractical for the majority of applications in which no single parameter set comes as part of the problem description.
>
> Fortunately, it turns out that certain such estimators can come close to being minimax over a whole class of parameter sets. We exchange exact optimality for a single problem for approximate optimality over a range of circumstances.

In short, an adaptive statistical estimator can achieve near minimax optimality simultaneously over a range of problem settings. Similarly, an *adaptive online learning*

algorithm should achieve near (typically, up to poly-logarithmic factors) minimax optimal regret bounds simultaneously over a range of OCO settings. Still within the static regret, let us consider a simple example: in our analysis of OGD, the tuning of OGD depends on D, which is the known size of the comparator class $\mathcal{U}$. This motivates a natural question on the existence of adaptive online learning algorithms: can a single algorithm work well simultaneously in a range of OCO settings with different-sized comparator classes? Actually, such D-adaptivity is a classical objective in adaptive online learning called *parameter-free online learning*, which will be considered frequently in this dissertation.

In addition, just like how both *minimax algorithms* and *minimax optimal bounds* are defined in the previous section, we can define *adaptive regret bounds* alongside the aforementioned *adaptive online learning algorithms*. Adaptive regret bounds refer to the generic regret bounds achieved by adaptive online learning algorithms, which, by definition, can be specialized to individual "restricted" OCO settings and almost match the minimax optimal bounds there.

The key benefit of adaptive online learning is that, the regret bound depends on the complexity of the *actually encountered problem instance*, rather than the complexity of the considered OCO setting. To be more specific, let us first look at the $O(DG\sqrt{T})$ static regret bound of OGD for the meaning of "complexity".

- D is an upper bound of $\|u - x_1\|_2$, which captures how hard the static comparator class is. A larger D amounts to allowing comparators to deviate more from the initialization of the agent, such that guaranteeing low regret becomes more difficult.

- G is the Lipschitz constant, which captures the complexity of the environment class. A larger G grants more power to the environment, which is unfavorable for the agent.

Together, the product DG measures the complexity of the OCO setting with static regret, G-Lipschitz losses, and comparators satisfying $\|u - x_1\|_2 \leq D$ – this product is the key quantity in the $O(DG\sqrt{T})$ bound.

In contrast, an adaptive[6] regret bound should not be tied to the complexity of any particular OCO setting. That is, if the actual comparator satisfies $\|u - x_1\|_2 \leq D'$ for some different unknown D', and the actual losses are G'-Lipschitz for some different unknown G', then an ideal adaptive bound needs to be $\tilde{O}(D'G'\sqrt{T})$, almost matching the minimax optimal rate $O(D'G'\sqrt{T})$. As a result, the corresponding adaptive algorithms are insensitive to any D and G given beforehand,[7] which are typically inexact estimates. Instead, the goal is to achieve generic static regret bounds like $\tilde{O}(\|u - x_1\|_2 \max_t \|g_t\|_2 \sqrt{T})$, where $\|u - x_1\|_2$ captures the complexity of the actual comparator u, and $\max_t \|g_t\|_2$ captures the complexity of the actual environment. It should be noted that the adaptivity with respect to both D and G is only used as an illustration; even here, there are finer measures of instance complexity than the ones above. Common types of adaptivity are surveyed in Section 1.4.

To summarize, adaptive online learning aims to replace problem complexities in the minimax optimal regret bounds by instance complexities, without knowing the latter beforehand. We say the algorithm or its bound *adapts* to a certain parameter of the problem if the latter in the minimax regret bound is replaced by a finer instance-dependent quantity. Such instance optimality is the main advantage of adaptive online learning, if we can algorithmically achieve it. Stemming from this central advantage, we list a few more interpretable benefits below. They have certain overlaps, and might be viewed as different ways to express similar ideas.

- The instance complexities are upper-bounded by the problem complexities, therefore adaptive regret bounds are no worse than minimax optimal regret

[6] With respect to both D and G.

[7] For an opposite case, compare it to OGD, where the standard learning rate is $\eta = DG^{-1}T^{-1/2}$.

bounds up to poly-logarithmic factors, and can improve the latter when the instance is "easy".

- Adaptive algorithms do not need many explicit restrictions on the problem setting, while achieving good theoretical performance as if such restrictions are imposed. Therefore, adaptive online learning can handle more general, "unrestricted" settings than minimax online learning.

  For example, the D-adaptive algorithms discussed in this section typically do not require any a priori upper bound on $\|u - x_1\|_2$. Instead, they bound $\sup_{Env} \text{Regret}_T(Env, u)$ by a near optimal function of u, which holds for all u in the domain $\mathcal{X}$, even if $\mathcal{X}$ is unbounded. In contrast, minimax algorithms like OGD require a restricted comparator class as input, and the optimality of the obtained regret bound is only established on this comparator class.

- Compared to minimax algorithms, the performance of adaptive algorithms depends weakly on their hyperparameter settings.

  For example, the regret bound of OGD depends polynomially on η and η^{-1}. In the practice of ML training, a grid search is often needed to determine η, due to the importance of this hyperparameter. It will be shown that the regret bound of D-adaptive algorithms depend only logarithmically on their hyperparameters. That is, adaptive algorithms are theoretically more robust to suboptimal hyperparameter tuning.

- Finally, the instance complexity is not a purely objective quantity – it depends on how much prior knowledge the learning agent has on the problem instance. The more prior knowledge it has, the easier the instance becomes. Therefore, adaptive online learning has an appealing *Bayesian interpretation*, with numerous potential applications. In our opinion, this is perhaps the strongest motivation

to study adaptive online learning, so the next section is devoted to this topic.

A bit more discussion before we proceed: in the application of OCO to downstream tasks, there exists a completely different, but still plausible definition of adaptivity. Given any sublinear (in T) bound on $\sup_{Env,u} \mathrm{Regret}_T(Env, u)$, it is sometimes argued that the learning agent "adapts" to the unknown environment by finding the environment-dependent optimal fixed prediction. There, adaptivity is not defined by the rate of convergence, but by the target of convergence; consequently, minimax algorithms in our definition are "adaptive" in that regard. As far as we understand, such a notion shares a similar spirit as the classical topic of adaptive control. In contrast, our definition of adaptivity is stronger, more quantitative, and closer to the convention in statistics.

1.3 Bayesian interpretation

As we briefly discussed in the previous section, adaptive online learning is especially motivated by its ability to incorporate *prior knowledge*. We now expand this argument with details.

In practice, machine learning tasks usually do not start from scratch. Before learning starts, we often have a reasonable guess on the task and the optimal decision, which should ideally make learning easier. Let us consider two examples.

- The workflow for a number of deep learning tasks consists of two steps, pretraining and fine-tuning. This is motivated by the lack of data or computational resources on the specific application of interest, which is quite common in practice. To address this issue, one could first pretrain a general purpose model on large open-access datasets with similar modality. Then, starting from the pretrained model, one could fine-tune it using the actual data, in order to enhance its performance on the more specific application. The rationale is that pretraining

provides a "warm start" for fine-tuning, which reduces the required amount of samples and training iterations. To rigorously justify this idea, the fine-tuning algorithm needs to provably utilize good initializations to accelerate.

- A trend in modern machine learning is to rely on more data rather than domain knowledge, as the later can be an imperfect abstraction of the real world. However, domain knowledge like physical simulation, if present, is still helpful for achieving better performance and reducing the amount of data required. Currently, many mature engineering solutions, like weather forecasting, are still based on simulating scientific models rather than machine learning. This poses a natural question: given an algorithm based on domain knowledge, can we provably incorporate it into the machine learning pipeline?

Adaptive online learning could provide an answer to the above questions. As introduced in the previous section, its performance guarantee depends on instance complexities rather than problem complexities. Here we argue that such an "instance" should be viewed more generally. It can be a problem instance, which is objective, and also the agent's inductive bias, which is subjective. Concretely, the $\|u - x_1\|_2$ term, which we described as the complexity of the comparator u, is actually determined by both u and the agent's initialization x_1. Therefore, it is more accurate (and more practically interesting) to say that, adaptive regret bounds depend on *the complexity of the actually encountered problem instance relative to the agent's prior knowledge*. The pretrained model and the scientific simulator discussed above are both prior knowledge independent of machine learning. They can be provably exploited by adaptive online learning algorithms, but not minimax algorithms.

We also emphasize the difference between the prior knowledge and the *oracle knowledge*. In this dissertation, the oracle knowledge refers to knowing the true problem instance, which is only revealed after learning completes. In contrast, the

prior knowledge is a guess before the game starts, which is "causal", but not necessarily correct. This shares the same idea as *Bayesian priors* in statistics.

1.4 Several types of adaptivity

Previous sections focused on the definition of adaptive online learning and its motivation. We now survey several common types of adaptivity in the literature. Each type has its own set of techniques, while intricate connections exist across different types. It is not a complete literature review – the latter goes way deeper into the details, and will be present in later chapters for clarity.

Starting from the static regret, we first introduce a concept that essentially belongs to adaptive online learning, but typically also pursued by minimax algorithms as well. Because of this, it is often excluded from modern discussions of adaptive online learning, therefore we number it as "Type 0".

(Type 0) Anytime regret In the definition of the OCO game (Definition 1), we deliberately treated the time horizon T vaguely, without specifying whether it is known by the learning agent or not. If T is known, then as introduced in Section 1.1, OGD with the learning rate $\eta \propto DG^{-1}T^{-1/2}$ guarantees the minimax optimal regret bound. A type of adaptivity originates from a natural follow-up question: can we achieve near minimax optimal regret bounds without knowing T? Such regret bounds are called *anytime*.

The anytime guarantee is of great importance in both theory and practice, as it characterizes the asymptotic convergence of the single algorithm we deploy. Achieving this adaptivity is in general not hard – we could either use the adaptive learning rate $\eta_t \propto DG^{-1}t^{-1/2}$ in OGD, or the so-called *doubling trick* (Shalev-Shwartz, 2011), which restarts the known-T algorithm on time intervals of doubling lengths. However, the combination of such anytime property with other types of adaptivity can be much

more involved, as we demonstrate in Chapter 2.

Now we turn to other types of adaptivity more commonly seen in modern discussions. They will be numbered sequentially as Type 1 to 4.

(Type 1) Comparator adaptivity This is the central theme of the dissertation. The main results will be developed in this regime, and the extension to other types of adaptivity will be discussed.

Also known as *parameter-freeness* in some existing works, comparator adaptivity is primarily meant for problems where either (i) the domain $\mathcal{X}$ is unbounded; or (ii) the domain is bounded with diameter D, but D is very large. A static regret bound is comparator adaptive if it depends on the comparator u with the order $\tilde{O}(\|u-\tilde{u}\|\sqrt{T})$, where $\|u-\tilde{u}\|$ characterizes the distance between the comparator to any reference point (i.e., prior) $\tilde{u}$ chosen at the beginning. Typically, the prior $\tilde{u}$ is the initialization x_1 of the algorithm. For example, a standard comparator adaptive bound (Orabona and Pál, 2016; Cutkosky and Orabona, 2018) guarantees for all $u \in \mathcal{X}$,

$$\sup_{Env} \mathrm{Regret}_T(Env, u) = O\left(\|u-\tilde{u}\|_2 G\sqrt{T\log(\|u-\tilde{u}\|_2 T)}\right),$$

which is optimal in a strong sense, and subsumes the $O(DG\sqrt{T})$ bound of OGD up to logarithmic factors. The cumulative loss of the learning agent can be bounded by the *oracle inequality*

$$\sum_{t=1}^{T} l_t(x_t) \le \inf_u \left[\sum_{t=1}^{T} l_t(u) + \sup_{Env} \mathrm{Regret}_T(Env, u)\right].$$

Notably, following the Bayesian interpretation (Section 1.3), the bound depends on the quality of the prior $\tilde{u}$. We say a prior is good if it is close to the optimal comparator in hindsight, i.e., the fixed prediction u^* with the lowest cumulative loss $\sum_{t=1}^{T} l_t(u^*)$, if it exists. In the context of training ML models, the latter means the

optimal model parameter which the optimizer aims to find, and we may obtain a guess of it via pretraining. A comparator adaptive algorithm allows using this guess to optimally speed up the convergence of the optimizer. In contrast, minimax algorithms like OGD fail to guarantee optimal bounds in this regime – as shown in Section 1.1, the bound of OGD only scales quadratically with respect to $\|u^* - \tilde{u}\|_2$ without a priori restrictions on the latter.

Comparator adaptive algorithms are also called parameter-free algorithms because unlike OGD, they usually do not have learning rates. For training ML models, the benefit is less amount of hyperparameter-tuning – comparator adaptive algorithms can work reasonably well even in their default setup (Orabona and Tommasi, 2017; Chen et al., 2022).

(Type 2) Gradient adaptivity The second type of adaptivity aims to improve the factor G in the minimax bound. Specifically, instead of depending explicitly on G and T, gradient adaptive bounds depend on observed gradient sums, e.g., $\sum_{t=1}^{T} \|g_t\|_2$ or $\sum_{t=1}^{T} \|g_t\|_2^2$.

For OCO with bounded domain this can be achieved easily, as one can choose an *adaptive learning rate* $\eta_t = D/\sqrt{\sum_{i=1}^{t} \|g_i\|_2^2}$ in OGD and guarantee $\sup_u \mathrm{Regret}_T(u) = O(D\sqrt{\sum_{t=1}^{T} \|g_t\|_2^2})$ (Streeter and McMahan, 2010), even without the a priori Lipschitz assumption $\|g_t\|_2 \le G$. A notable example empowered by this idea is AdaGrad (Duchi et al., 2011), which is a widely applied neural network optimizer. However, going beyond a bounded domain is more involved, as solutions (Cutkosky and Orabona, 2018; Mhammedi and Koolen, 2020) should go through the comparator adaptivity framework to achieve both forms of adaptivity simultaneously.

It is also possible to incorporate priors, which amounts to guessing the loss function l_t before the prediction x_t is made, and using that to determine a good x_t. Quantitatively, this replaces the gradient variance term $\sum_{t=1}^{T} \|g_t\|_2^2$ by $\sum_{t=1}^{T} \|g_t - \tilde{g}_t\|_2^2$,

where $\tilde{g}_{1:T}$ is a sequence of hallucinated gradients. Such an idea is also called *optimistic online learning* (Chiang et al., 2012; Rakhlin and Sridharan, 2013; Steinhardt and Liang, 2014).

All the previous types are discussed under the static regret. Next, let us move to the dynamic regret (1.1), without the implicitly assumed stationarity of the environment. Here the idea of adaptivity plays a more fundamental role, as there does not exist[8] a sublinear minimax optimal regret bound without restrictions on the comparator class. That is, for any OCO algorithm, there exist an environment Env and a dynamic comparator $\{u_t\}_{t\in\mathbb{Z}}$ satisfying Assumption 1, such that the dynamic regret $\text{Regret}_T(Env, u_{1:T}) = \Omega(T)$.

(Type 3) Dynamic regret The third type of adaptivity aims at dynamic regret bounds that depend on the structural simplicity of the comparator sequence $\{u_t\}_{t\in\mathbb{Z}}$. Typically, the comparator is considered "easy" if its *path length* $P = \sum_{t=1}^{T-1} \|u_{t+1} - u_t\|_2$ is low. Given an a priori upper bound on P, one could apply OGD with a learning rate that depends on this upper bound, resulting in minimax optimality (Zinkevich, 2003). The goal of adaptive online learning is relaxing such a requirement, while still performing as if this knowledge is given. Specifically, if the domain $\mathcal{X}$ is bounded with diameter D, then without requiring an additional upper bound on P, one could achieve (Zhang et al., 2018a)

$$\sup_{Env} \text{Regret}_T(Env, u_{1:T}) = \tilde{O}\left(G\sqrt{(D+P)DT}\right).$$

In the best case, $P = 0$, thus the RHS is $\tilde{O}(DG\sqrt{T})$, almost matching the classical OGD static regret bound. In the other extreme, $P = O(DT)$, and the regret bound is trivially $\tilde{O}(DGT)$.

[8]Excluding the trivial case where the domain only contains a single element.

Notice that P is determined by $u_{1:T}$, therefore on a broader scope, such bounds can also be called comparator adaptive, with certain overlap with the first type of adaptivity.

(Type 4) Local adaptivity The fourth type of adaptivity is also meant for nonstationary environments, but the definition is fundamentally different from the dynamic regret. It is implicitly assumed that the time horizon $[1:T]$ can be segmented into shorter subintervals where the environment is stationary, therefore we should consider the static regret on *local intervals*. Let us again consider the setting with D-bounded domain.

Specifically, the goal is to design an algorithm such that on *any* subinterval $[T_1:T_2] \subset [1:T]$, the *local static regret* on $[T_1:T_2]$, i.e.,

$$\text{Regret}_{[T_1:T_2]}(Env, u) := \sum_{t=T_1}^{T_2} l_t(x_t) - \sum_{t=T_1}^{T_2} l_t(u),$$

is sublinear in the interval length $T_2 - T_1 + 1$, regardless of Env and u. If the interval length $T_2 - T_1 + 1$ is known, then OGD with learning rate $\eta = DG^{-1}(T_2 - T_1 + 1)^{-1/2}$ achieves the minimax optimal local static regret $O(DG\sqrt{T_2 - T_1 + 1})$. Without knowing the interval length $T_2 - T_1 + 1$, there are adaptive algorithms (Daniely et al., 2015; Jun et al., 2017) achieving $\tilde{O}(DG\sqrt{T_2 - T_1 + 1})$, matching the minimax optimal rate up to poly-logarithmic factors.[9]

It is worth noting that this type of adaptivity is called *strongly adaptive online learning* in the literature (Daniely et al., 2015), improving the so-called *weakly adaptive online learning* (Hazan and Seshadhri, 2009). It seems that this terminology is purely due to historical reasons: the latter was proposed in the early stage of online learning, and it was probably hard to anticipate the development of adaptive online learning a

[9] As one could expect, the logarithmic factor depends on the time horizon T.

decade later. To differentiate this concept with other forms of adaptivity, we will also call it *local adaptivity*.

Finally, note that for Type 3 and 4, we did not discuss their Bayesian interpretations. This is a current research frontier, and the second part of this dissertation will study how to incorporate Bayesian priors into Type 3.

1.5 Overview of the dissertation

This dissertation is devoted to designing better adaptive online learning algorithms. The "better" here can be interpreted in multiple ways, including quantitatively improved regret bounds, simplified analysis, clearer intuition and easier integration with Bayesian priors. A few different techniques are presented, loosely centered around a central theme: ultimately, online learning is a "time-related" problem, therefore we should more closely examine the role of time in designing algorithms.

Specifically, the dissertation is divided into two parts.

- Part I focuses on *temporal continuity*. The key idea is that although OCO is a discrete time game-theoretic model, we can design algorithms by scaling it towards a continuous time limit, which involves less amount of guessing than the traditional approach. Such a view has been mostly overlooked by the online learning community, but recently, it started to pick up interests from a mathematical perspective, within the fields of differential equations and stochastic calculus. There, the emphasis is on rigorously connecting the obtained *Partial Differential Equation* (PDE) to the Bellman equation of the OCO game. This dissertation will focus more on the algorithmic implications, specifically presenting progresses in adaptive online learning.

  Chapter 2 presents the main algorithmic framework. By solving the continuous time PDE, we obtain a comparator adaptive (Type 1) online learning algorithm

based on an adaptive potential function. Quantitatively, its static regret bound achieves for the first time (i) the optimal *loss-regret tradeoff*, which will be rigorously defined; and (ii) the optimal leading constant. Accompanying lower bounds are also provided.

Chapter 3 applies the continuous time framework to a variant of OCO with switching costs, achieving the first optimal comparator adaptive regret bound there (in a certain strong sense). More importantly, this procedure shows how the continuous time scaling connects different online learning settings, allowing an easier transfer of algorithmic insights. The benefits are extended to another classical online learning problem called *Learning with Expert Advice* (LEA).

- Part II focuses on *temporal representation*. The key idea is to introduce time-dependent features into OCO, effectively turning it into *Online Linear Regression* (OLR). Although features are extremely common in machine learning, typically they are defined from *spacial variables*, with no explicit time dependence. Drawing novel connections to the wavelet theory, we show that this natural OLR approach turns out to be stronger than the state-of-the-art, more sophisticated methods.

  Specifically, Chapter 4 studies an objective called *unconstrained dynamic regret*, which unifies two classical types of adaptive online learning: the comparator adaptivity (Type 1) and the dynamic regret (Type 3). Such a regret bound is guaranteed to be *sparsity-adaptive* by our OLR algorithm; that is, given a collection of temporal features, the bound depends on how sparsely such features can represent the comparator sequence. The version with Haar wavelet features improves the best path length (i.e., P) dependent dynamic regret bounds from the literature.

Finally, Chapter 5 concludes the dissertation and discusses future directions.

Limitation of scope The scope of the dissertation is limited in the following ways. First, we only consider online learning with *full information* feedback, i.e., the entire loss function l_t or at least its gradient $g_t \in \partial l_t(x_t)$ is observed. The related *bandit* setting, where the feedback is the function value $l_t(x_t)$, is not considered. Second, we focus on intrinsically online and adversarial problems, therefore omit the specialization of our results to (IID) stochastic optimization. This is framed as the online-to-batch conversion in the literature, with its own nontrivial subtleties.

1.6 Notation

Here we list the common notations throughout this dissertation. More specialized notations are defined later when needed.

For two integers $a \le b$, $[a : b]$ is the set of all integers c such that $a \le c \le b$; the brackets are removed when on the subscript, denoting the tuple with indices in $[a : b]$.

$|S|$ denotes the cardinality of a finite set S.

$\mathbb{R}_+$ and $\mathbb{R}_{++}$ are the sets of nonnegative and strictly positive real numbers, respectively.

0 is a zero vector or matrix, which should be clear from the context.

We use $\|\cdot\|$ for the Euclidean norm of vectors and the spectral norm of matrices. These are the default norms, unless specified otherwise.

B^d denotes the unit d-dimensional Euclidean norm ball centered at the origin.

$\Pi_{\mathcal{V}}(x)$ is the Euclidean projection of x to a closed and convex set $\mathcal{V}$.

$\Delta(d)$ is the d-dimensional probability simplex.

$\mathrm{KL}(p||q)$ is the *Kullback–Leibler* divergence from the distribution q to p. When they are discrete,

$$\mathrm{KL}(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}.$$

$\lambda_{\max}(A)$ is the largest eigenvalue of a real symmetric matrix A.

Treating all vectors as column vectors, $\text{span}(A)$ represents the column space of a matrix A.

For a function f, let f^* be its Fenchel conjugate: $f^*(\theta) = \sup_x[\langle\theta, x\rangle - f(x)]$.

If the function f is convex, then $\partial f(x)$ denotes its subdifferential at x.

log denotes natural logarithm when the base is omitted, and $\log_+(\cdot) := 0 \vee \log(\cdot)$.

polylog$(\cdot)$ denotes a poly-logarithmic function of its input.

$\tilde{O}(\cdot)$ is the Big-Oh notation, neglecting poly-logarithmic factors. We will use "logarithmic" and "poly-logarithmic" interchangeably.

Part I

Temporal Continuity

Chapter 2

Comparator Adaptivity via PDE

This chapter is based on (Zhang et al., 2022b). The goal is to improve existing comparator adaptive static regret bounds, using a new design approach in continuous time. Section 2.1 motivates the problem and summarizes our results. Section 2.2 surveys related works. Section 2.3 introduces useful techniques in the literature, which will be applied frequently in this dissertation. The new continuous time framework is presented in Section 2.4, which converts algorithm design to solving a PDE. Section 2.5 presents a specific algorithm obtained from this framework, achieving a strong notion of optimality. Experiments are presented in Section 2.6. Section 2.7 concludes this chapter and discusses future directions. All the proofs are deferred to Section 2.8.

Setting We consider OLO (Definition 3) with Lipschitz losses (Assumption 1) and unknown time horizon T. Without loss of generality, we assume the Lipschitz constant $G = 1$ and the domain $\mathcal{X} = \mathbb{R}^d$, therefore the problem is also called *(anytime) unconstrained OLO*.[1] As discussed in Chapter 1, minimax guarantees become vacuous in the unconstrained setting, so the proposed algorithm needs to be comparator adaptive.

Furthermore, as shown in Section 1.4, comparator adaptive algorithms require a prior $\tilde{u}$, which is typically their initialization. Without loss of generality, this is set to

[1] It is known that (even time-varying) constraints on the domain can be imposed on any given unconstrained algorithm, without essentially changing its regret bound (Cutkosky and Orabona, 2018; Cutkosky, 2020). We survey this technique in Section 2.3. In other words, the unconstrained setting is no easier than the constrained setting.

the origin 0. General $\tilde{u}$ can be handled by simply shifting the coordinate system.

Notation Since the adaptivity to the adversarial environment Env is not considered in this chapter, we will bound the supremum of the static regret over the environment, defined as

$$\text{Regret}_T(u) := \sup_{Env} \text{Regret}_T(Env, u) = \sup_{Env} \left(\sum_{t=1}^{T} \langle g_t, x_t - u \rangle \right). \tag{2.1}$$

Notably, $\text{Regret}_T(0) = \sup_{Env} \sum_{t=1}^{T} \langle g_t, x_t \rangle$, which represents the *worst case cumulative loss* of the algorithm.

Also, we will use *potential functions* in the form of $V(t, S)$: $t \in \mathbb{N}_+$ is the time, and S is a *sufficient statistic*. If $V(t, S)$ is twice-differentiable, let $\nabla_t V$, $\nabla_{tt} V$, $\nabla_S V$ and $\nabla_{SS} V$ be its first and second order partial derivatives, with respect to its two arguments respectively.

2.1 Motivation and contribution

Comparator adaptive online learning was originally motivated by the need to handle unbounded domains. This is of great practical importance, as in ML training for example, the parameter of the model is usually unbounded a priori. Minimax algorithms fall short in this setting, as the standard analysis of OGD with time-invariant learning rates η guarantees (cf., Section 1.2)

$$\text{Regret}_T(u) \leq \frac{\|u\|^2}{2\eta} + \frac{1}{2}\eta G^2 T.$$

The optimal RHS is $O(\|u\| \sqrt{T})$, but this is not achievable simultaneously for all $u \in \mathbb{R}^d$, since the distance to an arbitrary comparator (i.e., $\|u\|$) is never known beforehand. Realistically, one could only set $\eta = O(1/\sqrt{T})$ and obtain $O(\|u\|^2 \sqrt{T})$ regret, which is suboptimal in the order of $\|u\|$.

Improving this result requires a drastically different procedure, and the predominant one is the *potential framework*. Given a potential function $V(t, S)$, the key idea is to accumulate the history into a "sufficient statistic" $S_t = -\sum_{i=1}^{t-1} g_i$ and predict the partial derivative $\nabla_S V(\cdot, \cdot)$ at (t, S_t), i.e., $x_t = \nabla_S V(t, S_t)$. Through this procedure, designing new algorithms is converted to a more tangible task of finding good potential functions. Specifically, with an arbitrary constant C, existing works (McMahan and Orabona, 2014; Orabona and Pál, 2016; Mhammedi and Koolen, 2020) adopted the one dimensional potential

$$V(t, S) = \frac{C}{\sqrt{t}} \exp\left(\frac{S^2}{2t}\right) \tag{2.2}$$

and its variants to achieve the comparator adaptive regret bound

$$\text{Regret}_T(u) \leq C + \|u\| O\left(\sqrt{T \log(C^{-1}\|u\|\sqrt{T})}\right). \tag{2.3}$$

This bound is anytime (see Type 0 in Section 1.4), i.e., it holds simultaneously for all time horizon $T \in \mathbb{N}_+$. Among all the achievable upper bounds with $\text{Regret}_T(0) \leq C$, the order of $\|u\|$ and T here is optimal up to multiplicative constants (Streeter and Mcmahan, 2012; Orabona, 2013). These results have been the gold standard in the literature, and due to their weak dependence on the hyperparameter C, the associated algorithms are called "parameter-free algorithms", with demonstrated practical advantages (Orabona and Tommasi, 2017).

Despite these strong results, there is still room for improvement though. For example, in ML training, requiring a constant $\text{Regret}_T(0)$ *all the time* amounts to a strong belief that the initialization of the model is close to the optimal parameter, which somewhat contradicts the use of an unconstrained domain in the first place. Reflected in the regret bound, the RHS of (2.3) can be more generally viewed as a *loss-regret tradeoff* between the values of $\text{Regret}_T(u)$ at small $\|u\|$ and large $\|u\|$: if

the cumulative loss $\text{Regret}_T(0)$ is allowed to increase with T, then one may obtain lower regret with respect to far-away comparators. This will be favorable in high dimensional problems, as good initializations become harder to obtain.

The question now becomes, what is the optimal loss-regret tradeoff, and how to efficiently achieve it? As a first attempt, one could further assume a known time horizon T, set $C = \sqrt{T}$ in (2.3) and obtain (McMahan and Orabona, 2014)

$$\text{Regret}_T(u) \leq \sqrt{T} + \|u\| O\left(\sqrt{T \log \|u\|}\right). \tag{2.4}$$

With respect to T alone, $\text{Regret}_T(u) = O(\sqrt{T})$. Since it matches the standard minimax lower bound for constrained OLO, it is reasonable to consider this loss-regret tradeoff as optimal. The real challenge is an anytime bound – existing arguments rely on the well-known *doubling trick*[2] (Shalev-Shwartz, 2011), which not only is notoriously impractical, but also leads to an extra multiplying constant. Perhaps due to this reason, regret bounds like (2.4) have received less attention than (2.3), despite their theoretical advantages.

This chapter aims at a practical and optimal approach towards an anytime version of (2.4), which requires a departure from existing techniques. Specifically, we will go back one step and reconsider the design of potential functions in unconstrained OLO. The classical workflow is based on heuristic guessing, which is challenging when the suitable potential is not an elementary function (e.g., involving complicated integrals or series). In contrast, we propose a framework based on continuous time scaling, which reduces the amount of guessing and allows designing more complicated potentials. Eventually, as a byproduct, we obtain a concrete new algorithm with the optimal loss-regret tradeoff.

Result and contribution As motivated above, our contributions are twofold.

[2]Running the fixed-T algorithm on consecutive time intervals of doubling lengths, i.e., $[2^i : 2^{i+1} - 1]$.

- We propose a framework that uses solutions of a specific *Partial Differential Equation* (PDE) as potential functions. To this end, we characterize minimax optimal potentials via a backward recursion, and a PDE arises in its continuous-time limit. Solutions of this PDE approximately solve the discrete-time recursion. Therefore, one may search for suitable potentials within such solutions and their variants, which is a more structured procedure than direct guessing.

- Using our framework, we design a one dimensional potential which is not elementary and hard to guess without the help of a PDE. For any hyperparameter $C > 0$, the induced algorithm guarantees

$$\text{Regret}_T(u) \leq C\sqrt{T} + \|u\| \sqrt{2T} \left[\sqrt{\log\left(1 + \frac{\|u\|}{\sqrt{2}C}\right)} + 2 \right].$$

  Our bound achieves the optimal loss-regret tradeoff (2.4) without the doubling trick. Moreover, by constructing a matching lower bound, we further show that the leading order term, including the constant multiplier $\sqrt{2}$, is tight. To our knowledge, the proposed algorithm is the first to achieve such optimality properties.

2.2 Related work

Unconstrained OLO Mcmahan and Streeter (2012) proposed the first comparator adaptive algorithm with $O(\|u\|\sqrt{T}\log(\|u\|T))$ regret, which was later improved to $O(\|u\|\sqrt{T\log(\|u\|T)})$ by a potential-based algorithm (McMahan and Orabona, 2014). This is the optimal rate (Mcmahan and Streeter, 2012; Orabona, 2013, 2019) under the constraint $\text{Regret}_T(0) \leq$ constant. The analysis was streamlined in (Orabona and Pál, 2016; Cutkosky and Orabona, 2018) through a *coin-betting game*, in (Foster et al., 2018) through the *Burkholder method*, and in (Chen et al., 2021; Jacobsen and Cutkosky, 2022) through aggregating non-adaptive algorithms. The obtained

algorithms find applications in differential privacy (Jun and Orabona, 2019; van der Hoeven, 2019), combining optimizers (Cutkosky, 2019b, 2020; Zhang et al., 2022a) and training neural networks (Orabona and Tommasi, 2017; Chen et al., 2022).

Among all these results, a shared limitation is the focus on $\text{Regret}_T(0) \leq \text{constant}$. Other loss-regret tradeoffs are less explored, both theoretically and practically. Moreover, the optimality of leading constants has not been considered.

Differential equations for online learning Recently, applying differential equations in online learning has received growing interests. The first idea was proposed by Kapralov and Panigrahy (2011), where a potential function for *Learning with Expert Advice* (LEA) (Littlestone and Warmuth, 1994) was designed by solving an *Ordinary Differential Equation* (ODE). As a key benefit, the obtained regret bound achieves a tradeoff with respect to different individual experts. The proposed techniques were later applied to the discounted setting (Andoni and Panigrahy, 2013) and the movement-constrained setting (Daniely and Mansour, 2019).

An improved approach uses PDEs (rather than ODEs) to generate time-dependent potential functions. Still considering the LEA problem, such works aim at the optimal regret bound nonasymptotic in the number of experts. Zhu (2014) first derived a PDE to characterize the continuous-time limit of LEA, whose arguments were streamlined by Drenska and Kohn (2020b). Exact solutions were obtained in special cases (Bayraktar et al., 2020a,b; Drenska and Kohn, 2020b), and more generally, algorithms based on approximate solutions were designed in (Rokhlin, 2017; Kobzar et al., 2020a,b). Follow-up works considered history-dependent experts (Drenska and Kohn, 2020a; Drenska and Calder, 2022), malicious experts (Bayraktar et al., 2020c, 2021) and drifting games (Wang and Kohn, 2022). Furthermore, Harvey et al. (2020) extended this idea to the anytime setting with two experts, using a different, stochastic derivation of the PDE.

Our use of PDE in unconstrained OLO is inspired by these works on LEA. Notably, there are two differences.

- Existing works considered settings that enforce a unique solution to the PDE, by requiring a fixed time horizon, e.g., (Zhu, 2014; Drenska and Kohn, 2020b; Kobzar et al., 2020a), or imposing boundary conditions (Harvey et al., 2020). In contrast, we directly consider a class of solutions which are generally not comparable to each other.

- In LEA, the goal of the PDE approach is to achieve the optimal uniform regret (with respect to all experts). In contrast, we use a PDE to achieve performance tradeoffs in adaptive online learning. Although tradeoffs among experts have been studied using ODEs, e.g., (Kapralov and Panigrahy, 2011), here we focus on the anytime setting where ODEs are not enough.

2.3 Standard techniques

Before starting, we survey two important techniques for comparator adaptive online learning, which will be applied repeatedly throughout this dissertation. Together with the loss-regret duality introduced in Section 2.4, they form the core machinery in the existing research of comparator adaptivity.

First, we present a polar decomposition trick, which reduces the general OLO problem from $\mathbb{R}^d$ to the 1D domain $\mathbb{R}$. This is due to (Cutkosky and Orabona, 2018). The idea is to separately learn the "magnitude" using a 1D comparator adaptive algorithm, and the "direction" using standard OGD on a norm ball. Note that here we assumed that $\|g_t\| \leq 1$.

Lemma 2.1 (Theorem 2 of (Cutkosky and Orabona, 2018))**.** *For all $T \in \mathbb{N}_+$, if $\mathcal{A}_{1d}$ guarantees regret bound $\text{Regret}_T(u) \leq R_T(u)$ for all $u \in \mathbb{R}$, then Algorithm 2.1 guarantees $\text{Regret}_T(u) \leq R_T(\|u\|) + \|u\|\sqrt{2T}$ for all $u \in \mathbb{R}^d$.*

Algorithm 2.1 Reducing unconstrained OLO from $\mathbb{R}^d$ to $\mathbb{R}$.

Require: A 1D unconstrained OLO algorithm $\mathcal{A}_{1d}$.

Define $\mathcal{A}_B$ as OGD on B^d with learning rate $\eta_t = 1/\sqrt{t}$, initialized at the origin.
for $t = 1, 2, \ldots$ **do**
  Obtain predictions $y_t \in \mathbb{R}$ from $\mathcal{A}_{1d}$ and $z_t \in \mathbb{R}^d$ from $\mathcal{A}_B$.
  Predict $x_t = y_t z_t \in \mathbb{R}^d$, observe the loss gradient $g_t \in \mathbb{R}^d$.
  Return $\langle g_t, z_t \rangle$ and g_t as the t-th loss gradient to $\mathcal{A}_{1d}$ and $\mathcal{A}_B$, respectively.
end for

The second technique is due to (Cutkosky and Orabona, 2018; Cutkosky, 2020), which imposes an additional domain constraint $\mathcal{V}$ on top of any OLO algorithm in a black box manner.

Algorithm 2.2 Adding constraints in OLO.

Require: An OLO algorithm $\mathcal{A}$ and an arbitrary nonempty, closed and convex domain $\mathcal{V}$.

for $t = 1, \ldots, T$ **do**
  Obtain the prediction $\tilde{x}_t$ from $\mathcal{A}$.
  Predict $x_t = \Pi_{\mathcal{V}}(\tilde{x}_t)$ and receive the loss subgradient g_t.
  Define a surrogate loss function h_t as

$$h_t(x) = \begin{cases} \langle g_t, x \rangle, & \text{if } \langle g_t, \tilde{x}_t \rangle \geq \langle g_t, x_t \rangle, \\ \langle g_t, x \rangle + \langle g_t, x_t - \tilde{x}_t \rangle \frac{\|x - \Pi_{\mathcal{V}}(x)\|}{\|x_t - \tilde{x}_t\|}, & \text{otherwise.} \end{cases}$$

  Obtain a subgradient $\tilde{g}_t \in \partial h_t(\tilde{x}_t)$ and return it to $\mathcal{A}$ as the t-th loss subgradient.
end for

Lemma 2.2 (Theorem 2 of (Cutkosky, 2020))**.** *Algorithm 2.2 has the following properties for all t: (1) h_t is a convex function on $\mathcal{V}$. (2) $\|\tilde{g}_t\| \leq \|g_t\|$. (3) For all $u \in \mathcal{V}$, $\langle g_t, x_t - u \rangle \leq \langle \tilde{g}_t, \tilde{x}_t - u \rangle$.*

2.4 Continuous time scaling

In general, our framework takes the discrete time unconstrained OLO problem to its continuous time limit, where potential function candidates are obtained by solving a PDE. It consists of three steps, detailed as follows. First, the OLO problem is

converted to a coin-betting problem – the latter is easier from a technical perspective, due to the absence of comparators.

2.4.1 Unconstrained coin-betting and duality

Unconstrained coin-betting is a two-person repeated game, with $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{C} = \mathsf{B}^d$ being the action space of the player and the adversary respectively. The player's policy $\boldsymbol{p}$ contains an initial bet $x_1 \in \mathcal{X}$ and a collection of functions $\{p_2, p_3, \ldots\}$, with $p_t : \mathcal{C}^{t-1} \to \mathcal{X}$. Similarly, the adversary's policy $\boldsymbol{a}$ is defined as a collection of functions $\{a_1, a_2, \ldots\}$, with $a_t : \mathcal{X}^t \to \mathcal{C}$. Randomized betting strategies are not considered.

Fixing policies $\boldsymbol{p}$ and $\boldsymbol{a}$ on both sides, the game runs as follows. In the t-th round, the player makes a bet $x_t = p_t(c_{1:t-1})$ based on past coin outcomes. Then, the adversary decides a new coin $c_t = a_t(x_{1:t})$, reveals it to the player, and the player gains $\langle c_t, x_t \rangle$ amount of money (effectively, the player loses money if $\langle c_t, x_t \rangle$ is negative). The performance metric of the player is the total gained wealth

$$\mathrm{Wealth}_T = \sum_{t=1}^{T} \langle c_t, x_t \rangle ,$$

where T is not pre-specified. In other words, the player aims to ensure an anytime wealth lower bound against all possible adversaries.

Research on adversarial betting has a long history. In a seminal work, Cover (1966) studied the setting with a fixed and known time horizon, where all achievable lower bounds can be characterized via dynamic programming. The anytime setting here is more involved, but due to a classical dual relation (McMahan and Orabona, 2014), solving it is equivalent to solving the unconstrained OLO problem we ultimately care about: one can construct a unique OLO algorithm (Algorithm 2.3) from any coin-betting algorithm $\mathcal{A}$, and characterize its performance through Lemma 2.3.

Lemma 2.3 (Theorem 9.6 of (Orabona, 2019))**.** *Let Ψ be any proper, closed and convex function. For all $T \in \mathbb{N}_+$, the following two statements are equivalent:*

Algorithm 2.3 From coin-betting to OLO.

Require: An algorithm $\mathcal{A}$ for unconstrained coin-betting.

1: **for** $t = 1, 2, \ldots$ **do**
2: Query $\mathcal{A}$ for its t-th bet x_t and predict it *exactly* in OLO.
3: Observe loss gradient g_t and suffer $\langle g_t, x_t \rangle$.
4: Let $c_t = -g_t$ and send it to $\mathcal{A}$ as the t-th coin outcome.
5: **end for**

1. *The unconstrained coin-betting algorithm $\mathcal{A}$ guarantees* $\mathrm{Wealth}_T \geq \Psi\left(\sum_{t=1}^{T} c_t\right)$ *against any adversary.*

2. *The unconstrained OLO algorithm constructed from $\mathcal{A}$ guarantees* $\mathrm{Regret}_T(u) \leq \Psi^*(u)$ *for all* $u \in \mathbb{R}^d$. *(Ψ^* is the Fenchel conjugate of Ψ.)*

This unconstrained coin-betting game generalizes an existing setting, which is prevalent in the analysis of unconstrained OLO (McMahan and Orabona, 2014; Orabona and Pál, 2016). The latter assigns an initial wealth C to the player, and the player's betting amount $|x_t|$ should be less than the total wealth it possesses at the beginning of the t-th round. A budget constraint of this form faithfully models many real-world investment problems, but since our ultimate goal is online learning rather than any particular financial application, such a constraint is not necessary. Relaxing it gives us greater flexibility to achieve general forms of regret tradeoffs beyond (2.3). Intuitively speaking, the player in our setting can make decisions solely based on the perceived risk-gain tradeoff, without being constrained by its budget.

2.4.2 Minimax coin-betting

The second step characterizes the unconstrained coin-betting game from a minimax perspective. Rather than the *value* of the game, let us consider a refined quantity called the *value function*.

Definition 4 (Value function)**.** *A function $V : \mathbb{N} \times \mathbb{R}^d \to \mathbb{R}$ is a value function of the unconstrained coin-betting game if*

1. $V(0,0) = 0$.
2. *For all* $t \in \mathbb{N}$, $V(t,\cdot)$ *is continuous on* $\mathbb{R}^d$.
3. *For all* $t \in \mathbb{N}$ *and* $S \in t \cdot \mathcal{C}$,

$$V(t,S) = \min_{x \in \mathcal{X}} \max_{c \in \mathcal{C}} \left[V(t+1, S+c) - \langle c, x \rangle \right]. \tag{2.5}$$

The recursive relation in Definition 4 is reminiscent of the *conditional value function* previously studied in online learning (Rakhlin et al., 2012; McMahan and Abernethy, 2013; Drenska and Kohn, 2020b) and minimax dynamic programming (Bertsekas, 2012). The key difference is that we care about the anytime performance, therefore a terminal condition to initiate the backward recursion (2.5) is missing. Rather than the *value-to-go*, we model the *value-so-far*. This largely complicates the analysis, as the solution of (2.5) is not unique anymore (e.g., $V(t,S) = \text{constant} \cdot S$). In general, similar to the concept of *Pareto optimality*, different value functions are not comparable as they represent different tradeoffs on the shape of the wealth lower bound (ultimately, the associated regret upper bound due to Lemma 2.3).

On the bright side, *any* value function can lead to a pair of player-adversary strategies with tight wealth lower and upper bounds. Given a good value function (or more generally, its approximation), a good betting algorithm can be naturally induced. All proofs are deferred to Section 2.8.

Lemma 2.4. *Given any value function* V *satisfying Definition 4,*

1. *There exists a player policy* $\boldsymbol{p}^*$ *such that for all* $\boldsymbol{a}$ *and* $T \in \mathbb{N}_+$,

$$\text{Wealth}_T \geq V\left(T, \sum_{t=1}^{T} c_t\right).$$

In addition, for all t, *the player's bet* $p_t^*(c_{1:t-1})$ *depends on the past coins only through their sum* $\sum_{i=1}^{t-1} c_i$.

2. There exists an adversary policy $\boldsymbol{a}^*$ *such that for all* $\boldsymbol{p}$ *and* $T \in \mathbb{N}_+$,

$$\text{Wealth}_T \leq V\left(T, \sum_{t=1}^{T} c_t\right).$$

To proceed, let us further define the *unit time* as the time interval between consecutive rounds in the coin betting game, and assign it to 1. In this way, the game can be analyzed on the real time axis $t \in \mathbb{R}_+$, which prepares us for the scaling.

2.4.3 The limiting PDE

Intuitively, solving the backward recursion (2.5) is difficult due to its discrete formulation. If a finer discretization on the time axis is adopted, then the recursion becomes "smoother" which is easier to describe using continuous-time analysis. To this end, we introduce a *scaled coin-betting game*. The scaling factors are due to an analogy to existing coin-betting results (McMahan and Abernethy, 2013), since we aim to recover the existing potential (2.2).

Definition 5 (Scaled game)**.** *Given* $\varepsilon > 0$*, the* ε*-scaled game is the unconstrained coin-betting game with unit time* ε^2 *and adversary action space* $\varepsilon \cdot \mathcal{C}$*. That is, actions are taken every* ε^2 *original unit time, and the adversary chooses the coin outcomes in a scaled set* $\varepsilon \cdot \mathcal{C}$ *instead of* $\mathcal{C}$*.*

Similar to Definition 4, we can define ε*-scaled value functions* V_ε on the scaled game. Moreover, let us extend its domain and assume it is twice-differentiable on $\mathbb{R}_{++} \times \mathbb{R}^d$. The backward recursion on V_ε becomes

$$V_\varepsilon(t, S) = \min_{x \in \mathcal{X}} \max_{c \in \mathcal{C}} \left[V_\varepsilon(t + \varepsilon^2, S + \varepsilon c) - \langle \varepsilon c, x \rangle\right].$$

Similar to (Zhu, 2014; Drenska and Kohn, 2020b), we take a Taylor approximation on

the RHS,

$$V_\varepsilon(t+\varepsilon^2, S+\varepsilon c) = \\ V_\varepsilon(t,S) + \varepsilon^2 \nabla_t V_\varepsilon(t,S) + \varepsilon \left\langle c, \nabla_S V_\varepsilon(t,S) \right\rangle + \frac{\varepsilon^2}{2} \left\langle \nabla_{SS} V_\varepsilon(t,S) \cdot c, c \right\rangle + o(\varepsilon^2),$$

which leads to

$$0 = \min_{x \in \mathcal{X}} \max_{c \in \mathcal{C}} \big[\left\langle c, \nabla_S V_\varepsilon(t,S) - x \right\rangle + \varepsilon \nabla_t V_\varepsilon(t,S) + \frac{\varepsilon}{2} \left\langle \nabla_{SS} V_\varepsilon(t,S) \cdot c, c \right\rangle + o(\varepsilon) \big].$$

As ε approaches 0, the dominant term on the RHS is $\min_{x \in \mathcal{X}} \max_{c \in \mathcal{C}} \left\langle c, \nabla_S V_\varepsilon(t,S) - x \right\rangle$, therefore the outer minimizing argument should be $x = \nabla_S V_\varepsilon(t,S)$. Taking $\varepsilon \to 0$ and plugging in $\mathcal{C} = \mathsf{B}^d$ (i.e., the unit d-dimensional Euclidean norm ball), the result is a second order nonlinear PDE for a *limiting value function*.

Definition 6 (Limiting value function)**.** *A function $\bar{V} : \mathbb{R}_{++} \times \mathbb{R}^d \to \mathbb{R}$ is a limiting value function of the unconstrained coin-betting game if*

$$\nabla_t \bar{V} = -\frac{1}{2} \max\{\lambda_{\max}(\nabla_{SS} \bar{V}), 0\}. \tag{2.6}$$

The PDE (2.6) can be regarded as a continuous-time approximation of the backward recursion (2.5), and solving it, while still being challenging, is more tractable than handling the discrete-time recursion itself. Given solutions of this PDE, one may invoke a perturbed analysis of Lemma 2.4 and obtain corresponding wealth lower bounds.

2.5 Optimal loss-regret tradeoff

To generate concrete algorithms using this framework, let us consider the one dimensional convex case where the nonlinear PDE (2.6) becomes linear. Results can be extended to the general dimensional case, due to the standard polar decomposition technique (Lemma 2.1). To further comply with the duality lemma (Lemma 2.3),

consider $\bar{V}$ that are convex with respect to the second argument. Then, the PDE (2.6) reduces to the one dimensional *Backward Heat Equation* (BHE)

$$\nabla_t \bar{V} = -\frac{1}{2} \nabla_{SS} \bar{V}. \tag{2.7}$$

Such a linear PDE has received considerable interest from a mathematical perspective (Miranker, 1961; Payne, 1975), since its initial value problem has an intriguing *ill-posed* issue. Related to our task, an insightful work by Harvey et al. (2020) showed that BHE gives rise to an optimal two-expert LEA algorithm – the proposed discretization techniques will be useful in our analysis as well. Our main observations are twofold:

- The PDE framework recovers both the OGD potential and the existing comparator adaptive potential (2.2), thus appears to be a very general approach for unconstrained OLO.
- The optimal potential that Harvey et al. adopted for two-expert LEA is also strong for *adaptive online learning*, resulting in an optimal unconstrained OLO algorithm for general dimensional problems.

2.5.1 Solutions to algorithms

Concretely, motivated by the classical comparator adaptive potential (2.2), let us consider the ansatz

$$\bar{V}(t, S) = t^{\alpha} g\left(c \cdot t^{\beta} S\right), \tag{2.8}$$

where α, β and c are constants, and $g : \mathbb{R} \to \mathbb{R}$ is a one dimensional function to be determined. For simplicity we omit shifting on S, t and the function value. In other words, once we find appropriate (α, β, c) and g, we immediately obtain a more general solution

$$\bar{V}(t, S) = C_0 + (t + \tau)^{\alpha} g\left(c \cdot t^{\beta} (S + S_0)\right),$$

with shifting constants C_0, τ and S_0. Moreover, any linear combination of two solutions is also a solution, allowing the user to interpolate their induced behavior.

Plugging in (2.8) and letting $z = c \cdot t^{\beta} S$, the BHE (2.7) reduces to a second order linear ODE for the function g:

$$c^2 t^{2\beta+1} g''(z) + 2\beta z g'(z) + 2\alpha g(z) = 0.$$

Letting $\beta = -1/2$ and $c = 1/\sqrt{2}$, it becomes the standard *Hermite* type

$$g''(z) - 2z g'(z) + 4\alpha g(z) = 0, \tag{2.9}$$

whose general solutions can be expressed in power series (Arfken et al., 2013, Chapter 7). By varying the parameter α, we obtain a rich class of limiting value functions $\bar{V}$.

To construct coin-betting policies, the key idea is to use $\bar{V}$ as a surrogate for the actual value function V (Definition 4) and apply the same argument as in Lemma 2.4. Specifically, the adversary should pick the coin outcome that maximizes the RHS of the backward recursion (2.5), which is

$$c_t \in \arg\max_{c \in \mathcal{C}} \left[\bar{V}\left(t, \sum_{i=1}^{t-1} c_i + c \right) - \langle c, x_t \rangle \right]. \tag{2.10}$$

Since $\bar{V}$ is convex and $\mathcal{C} = [-1, 1]$, the adversary can simply focus on the boundary coins $\{-1, 1\}$, leading to the adversary policy presented in Algorithm 2.4.

Algorithm 2.4 PDE-based adversary betting policy.

Require: A limiting value function $\bar{V}$ for 1D unconstrained coin-betting.

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

2: Receive the player's bet x_t and choose the coin outcome as

$$c_t \in \arg\max_{c \in \{-1,1\}} \left[\bar{V}\left(t, \sum_{i=1}^{t-1} c_i + c \right) - \langle c, x_t \rangle \right]. \tag{2.11}$$

3: **end for**

As for the player, the optimal bet is the one that minimizes the objective function in (2.10), which is equivalent to the *discrete derivative* (denoted by $\bar{\nabla}$) shown in Algorithm 2.5. Intuitively, the discrete derivative serves as an approximation of the standard derivative in classical potential methods. Therefore, Algorithm 2.5 essentially has a potential-based structure, with the potential function $\bar{V}$ generated from a PDE. Alternatively, Algorithm 2.5 can be interpreted as a discrete approximation of *Follow the Regularized Leader* (FTRL) (Abernethy et al., 2008b) whose regularizer is the Fenchel conjugate of $\bar{V}(t, \cdot)$. The equivalence of potential functions and regularizers has been discussed in (Orabona, 2019, Section 7.3).

Algorithm 2.5 PDE-based player betting policy.

Require: A limiting value function $\bar{V}$ for 1D unconstrained coin-betting.

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

2: Choose the bet

$$\begin{aligned} x_t &= \bar{\nabla}_S \bar{V}\left(t, \sum_{i=1}^{t-1} c_i\right) \\ &:= \frac{1}{2}\left[\bar{V}\left(t, \sum_{i=1}^{t-1} c_i + 1\right) - \bar{V}\left(t, \sum_{i=1}^{t-1} c_i - 1\right)\right]. \end{aligned} \tag{2.12}$$

3: Observe the coin outcome c_t and store it.

4: **end for**

2.5.2 Example

Before any technical analysis, let us demonstrate the generality of this framework through a few examples. We show how existing algorithms can be derived from this framework, and more importantly, there is a potential function which permits an optimal loss-regret tradeoff.

For any α, let $\bar{V}_\alpha$ be a limiting value function obtained from (2.9). Let $C > 0$ be any positive scaling constant.

Warm up: $\alpha = 1$**.** The Hermite ODE (2.9) has a solution $g(z) = C(2z^2 - 1)$, resulting in $\bar{V}_1(t, S) = C(S^2 - t)$. Accordingly, Algorithm 2.5 bets $x_t = 2C \sum_{i=1}^{t-1} c_i = x_{t-1} + 2Cc_{t-1}$, which is equivalent to OGD with the learning rate $2C$. Notably, $\bar{V}_1$ also satisfies Definition 4; that is, $\bar{V}_1$ is not only a limiting value function, but also a value function for the discrete-time game. Therefore, both Algorithm 2.4 and Algorithm 2.5 can be directly analyzed through Lemma 2.4.

Recovering existing potentials: $\alpha = -1/2$**.** The Hermite ODE can be solved by $g(z) = C \exp(z^2)$, resulting in $\bar{V}_{-1/2}(t, S) = C \cdot t^{-1/2} \exp[S^2/(2t)]$. Such a potential recovers the existing popular choice (2.2), and its time shifted version $C \cdot (t + \tau)^{-1/2} \exp[S^2/(2(t + \tau))]$ also recovers the *shifted potential* (Orabona and Pál, 2016). Different from the previous example, $\bar{V}_{-1/2}$ does not satisfy Definition 4. Therefore, we should characterize its approximation error on the backward recursion (2.5) in order to quantify the performance of the induced player policy.

A new potential: $\alpha = 1/2$**.** The two linearly independent solutions of the Hermite ODE are both useful. First, $g(z) = \sqrt{2}Cz$ and $\bar{V}(t, S) = CS$. Such a potential leads to betting a fixed amount in coin-betting and shifting the coordinate system in unconstrained OLO. For now, let us focus on the other solution which is more interesting.

$$g(z) = C \left[2z \cdot \int_0^z \exp(x^2) dx - \exp(z^2) \right],$$

and the corresponding potential is

$$\bar{V}_{1/2}(t, S) = C\sqrt{t} \left[2 \int_0^{\frac{S}{\sqrt{2t}}} \left(\int_0^u \exp(x^2) dx \right) du - 1 \right]. \tag{2.13}$$

It is based on the integral of the *imaginary error function* $\operatorname{erfi}(u) = \int_0^u \exp(x^2)dx$,[3] therefore we will also call it the *erfi potential*.

Notably, Harvey et al. (2020) constructed a two-expert LEA algorithm from $\bar{V}_{1/2}$, which achieves the optimal uniform regret. As for unconstrained OLO, we will show that using $\bar{V}_{1/2}$ in Algorithm 2.5 leads to superior performance compared to $\bar{V}_{-1/2}$, both in theory and in practice. Without the help of a PDE, such a potential has not been discovered in adaptive online learning before.

2.5.3 Discretization

Now let us plug the potential functions derived above into the betting policy (Algorithm 2.5). To begin with, define discrete derivatives of a limiting value function $\bar{V}$ as

$$\bar{\nabla}_t \bar{V}(t,S) := \bar{V}(t,S) - \bar{V}(t-1,S),$$

$$\bar{\nabla}_{SS} \bar{V}(t,S) := \bar{V}(t,S+1) + \bar{V}(t,S-1) - 2\bar{V}(t,S).$$

When doing this we extend the domain of $\bar{V}(t,S)$ to $t=0$, and assign $\bar{V}(0,0)=0$ without loss of generality.

The key component of this analysis is the *Discrete Itô formula* (Klenke, 2013; Harvey et al., 2020). We slightly modify it for the coin-betting problem.

Lemma 2.5 (Lemma D.3 and D.4 of (Harvey et al., 2020), adapted)**.** *Consider applying the player betting policy (Algorithm 2.5) against any adversary betting policy.*

[3] Compared to the usual definition, an extra multiplying constant is removed to simplify the writing.

For all $t \in \mathbb{N}$,

$$\bar{V}\left(t+1, \sum_{i=1}^{t+1} c_i\right) - \bar{V}\left(t, \sum_{i=1}^{t} c_i\right) \leq c_{t+1}x_{t+1} + \underbrace{\left[\bar{\nabla}_t \bar{V}\left(t+1, \sum_{i=1}^{t} c_i\right) + \frac{1}{2}\bar{\nabla}_{SS}\bar{V}\left(t+1, \sum_{i=1}^{t} c_i\right)\right]}_{\lozenge}. \quad (2.14)$$

Moreover, equality is achieved when $c_{t+1} \in \{-1, 1\}$.

Summing (2.14) over $t \in [0 : T-1]$, the LHS becomes a telescopic sum which returns $\bar{V}(T, \sum_{i=1}^{T} c_i)$, and the RHS contains $\text{Wealth}_T = \sum_{t=1}^{T} c_t x_t$ which we aim to bound – the remaining task is to quantify the sum $\lozenge$ in the bracket. Comparing $\lozenge$ to the BHE (2.7), one can see that $\lozenge$ represents the "discrete approximation error". Bounding this error is case-dependent, and we will only consider $\bar{V}_{1/2}$ here.

Lemma 2.6. *For all $t \in \mathbb{N}_+$ and $S \in [1-t, t-1]$, $\bar{V}_{1/2}$ with any parameter $C > 0$ satisfies*

$$\bar{\nabla}_t \bar{V}_{1/2}(t, S) + \frac{1}{2}\bar{\nabla}_{SS}\bar{V}_{1/2}(t, S) \leq 0,$$

and

$$\bar{\nabla}_t \bar{V}_{1/2}(t, S) + \frac{1}{2}\bar{\nabla}_{SS}\bar{V}_{1/2}(t, S) \geq \begin{cases} -C, & t = 1, \\ -\frac{C}{8}(t-1)^{-3/2} \exp\left(\frac{S^2}{2(t-1)}\right)\left(\frac{S^2}{t-1} + 1\right), & t > 1. \end{cases}$$

Combining the above, we immediately obtain a wealth lower bound for the player policy constructed from $\bar{V}_{1/2}$. Its proof is due to a telescopic sum therefore omitted.

Lemma 2.7. *For all $T \in \mathbb{N}_+$, Algorithm 2.5 constructed from $\bar{V}_{1/2}$ guarantees a wealth lower bound*

$$\text{Wealth}_T \geq \bar{V}_{1/2}\left(T, \sum_{t=1}^{T} c_t\right),$$

against any adversary policy $\boldsymbol{a}$.

2.5.4 Back to OLO

Finally, we convert the above results in unconstrained coin-betting back to unconstrained OLO, using the duality lemma (Lemma 2.3). Furthermore, the standard polar decomposition technique (Lemma 2.1) is adopted, which extends comparator adaptive regret bounds in $\mathbb{R}$ to $\mathbb{R}^d$. In combination, we arrive at our main unconstrained OLO algorithm (Algorithm 2.6), whose regret bound is presented as Theorem 2.1.

Algorithm 2.6 PDE-based unconstrained OLO algorithm.

Require: A 1D limiting value function $\bar{V}$ which satisfies (2.7).
1: Define $\mathcal{A}_B$ as OGD on B^d with learning rate $\eta_t = 1/\sqrt{t}$, initialized at the origin.
2: Initialize a parameter ("sufficient statistic") $S_1 = 0$.
3: **for** $t = 1, 2, \ldots$ **do**
4: Let $y_t = \left[\bar{V}\left(t, S_t + 1\right) - \bar{V}\left(t, S_t - 1\right)\right]/2$.
5: Query $\mathcal{A}_B$ for its t-th prediction and assign it to z_t.
6: Predict $x_t = y_t z_t \in \mathbb{R}^d$.
7: Observe the loss gradient $g_t \in \mathbb{R}^d$.
8: Return g_t as the t-th loss gradient to $\mathcal{A}_B$, and let $S_{t+1} = S_t - \langle g_t, z_t \rangle$.
9: **end for**

Theorem 2.1. *For all $T \in \mathbb{N}_+$ and $u \in \mathbb{R}^d$, Algorithm 2.6 constructed from $\bar{V}_{1/2}$ guarantees*

$$\mathrm{Regret}_T(u) \leq C\sqrt{T} + \|u\| \sqrt{2T} \left[\sqrt{\log\left(1 + \frac{\|u\|}{\sqrt{2}C}\right)} + 2 \right].$$

Theorem 2.1 offers two advantages over existing results.

1. It is a intrinsically anytime bound with the optimal[4] loss-regret tradeoff, i.e., $\mathrm{Regret}_T(u) = O\left(\|u\| \sqrt{T \log \|u\|}\right)$, shaving a $\sqrt{\log T}$ factor from (2.3). Actually, as discussed in Section 2.1, prior works *can* achieve this optimal tradeoff, but they rely on the impractical doubling trick (Shalev-Shwartz, 2011) for an anytime

[4] In the sense that the asymptotic rate on T alone is optimal. That is, compared to the optimally tuned gradient descent algorithm with regret $O(\|u\| \sqrt{T})$, the *price of being comparator adaptive* is only an extra $\sqrt{\log \|u\|}$ factor.

bound. In contrast, our algorithm does not need any restart, thus making the optimal loss-regret tradeoff practical.

2. In addition, Theorem 2.1 also attains the optimal leading term, including the multiplying constant $\sqrt{2}$. To our knowledge, this is the first comparator adaptive bound with the leading constant optimality. After deriving the algorithm-independent lower bound, we will present the precise statement in Theorem 2.3.

2.5.5 Lower bound

Complementing the upper bound, we now derive regret lower bounds. They can be either independent or dependent on the online learning algorithm – while traditional regret lower bounds are *algorithm-independent*, our discretization approach allows handling *algorithm-dependent* lower bounds quite easily. That is, we can characterize the performance of the main algorithm in the worst case environment. This in particular shows the tightness of our analysis in a strong sense.

Let us first consider algorithm-independent results.

Algorithm-independent lower bound Due to the duality lemma, the regret lower bound follows from a wealth upper bound. The latter is presented as Lemma 2.8, which has a similar shape as (Mcmahan and Streeter, 2012; Orabona, 2013), but considers a different tradeoff, with tighter constants. The refinement is due to a better characterization of the tail probability of random walks, through the *Berry-Esseen theorem*.

Lemma 2.8. *For all* $\lambda \geq \exp[(\sqrt{2}+1)/2]$, $T \geq 8\pi\lambda^2 \log\lambda$, *and any player betting policy* $\boldsymbol{p}$ *that guarantees* $\text{Wealth}_T \geq -C\sqrt{T}$ *(e.g., Algorithm 2.5 constructed from* $\bar{V}_{1/2}$*), there exists an adversary betting policy* $\boldsymbol{a}$ *such that the following statement holds. In the coin-betting game induced by the policy pair* $(\boldsymbol{p}, \boldsymbol{a})$,

1. $|\sum_{t=1}^{T} c_t| \geq \sqrt{2T\log\lambda}$;

2. $\text{Wealth}_T \le 2\sqrt{2\pi}\lambda\sqrt{\log\lambda}\cdot C\sqrt{T}$.

Converting the above wealth upper bound to OLO, we have

Theorem 2.2. *For all* $\eta \in (0,1)$, $U \ge 12\eta^{-1}C$, $T \ge 2\eta^2U^2C^{-2}\log(\eta UC^{-1})$ *and any unconstrained OLO algorithm* $\mathcal{A}$ *that guarantees* $\text{Regret}_T(0) \le C\sqrt{T}$ *(e.g., Algorithm 2.6 constructed from* $\bar{V}_{1/2}$*), there exists an adversarial OLO environment Env and a comparator* $u \in \mathbb{R}^d$ *such that* $\|u\| = U$ *and*

$$\text{Regret}_T(Env, u) \ge (1-\eta)\|u\|\sqrt{2T\log\frac{\eta\|u\|}{2\sqrt{\pi}C}}.$$

This result alone is perhaps a bit hard to interpret. Let us put it beside our main regret upper bound (Theorem 2.1) to make the message clearer. Here, $\text{Regret}_T^{\mathcal{A}}(Env, u)$ denotes the regret induced by an algorithm $\mathcal{A}$, in the environment Env, and with the comparator u.

Theorem 2.3. *Define* $\mathcal{A}_{1/2}$ *as Algorithm 2.6 constructed from* $\bar{V}_{1/2}$*, then Theorem 2.1 leads to*

$$\limsup_{U\to\infty}\limsup_{T\to\infty}\sup_{\|u\|=U,Env}\frac{\text{Regret}_T^{\mathcal{A}_{1/2}}(Env,u)}{\|u\|\sqrt{T\log\|u\|}} \le \sqrt{2}.$$

Conversely, for all C *and any unconstrained OLO algorithm* $\mathcal{A}$ *(e.g.,* $\mathcal{A}_{1/2}$*) that guarantees* $\text{Regret}_T^{\mathcal{A}}(0) \le C\sqrt{T}$ *for all* T*, we have*

$$\liminf_{U\to\infty}\liminf_{T\to\infty}\sup_{\|u\|=U,Env}\frac{\text{Regret}_T^{\mathcal{A}}(Env,u)}{\|u\|\sqrt{T\log\|u\|}} \ge \sqrt{2}.$$

It shows that under the optimal loss-regret tradeoff (i.e., $\text{Regret}_T^{\mathcal{A}}(0) \le C\sqrt{T}$), our regret upper bound has the optimal leading constant, which is the first in the literature.

Algorithm-dependent lower bound As for algorithm-dependent results, we aim to evaluate how tightly our regret upper bounds characterize the performance of our main potential function $\bar{V}_{1/2}$. To slightly simplify the analysis, let us consider Algorithm 2.7 below, which is essentially the 1D version of our main algorithm

(Algorithm 2.6), but without the need of a direction learner. With the definition $f_T(S) := \bar{V}_{1/2}(T, S)$, its regret upper bound (Corollary 2.4) is a simple corollary of the wealth lower bound (Lemma 2.7) and the duality lemma (Lemma 2.3). The proof is omitted.

Algorithm 2.7 PDE-based 1D unconstrained OLO algorithm.

Require: A 1D limiting value function $\bar{V}$ which satisfies (2.7).

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

2: Predict

$$x_t = \frac{1}{2}\left[\bar{V}\left(t, -\sum_{i=1}^{t-1} g_i + 1\right) - \bar{V}\left(t, -\sum_{i=1}^{t-1} g_i - 1\right)\right].$$

3: Observe the loss gradient g_t and store it.

4: **end for**

Corollary 2.4. *For all $T \in \mathbb{N}_+$ and $u \in \mathbb{R}$, Algorithm 2.7 constructed from $\bar{V}_{1/2}$ guarantees*

$$\mathrm{Regret}_T(u) \le f_T^*(|u|).$$

For this particular algorithm, we can construct a regret lower bound (Theorem 2.5) by combining the Discrete Itô formula (Lemma 2.5), the lower bound of the discretization error (Lemma 2.6) and the duality lemma (Lemma 2.3).

Theorem 2.5. *For all $T \in \mathbb{N}_+$ and $|u| \le (3/8)C(T+3)\exp(T/2)$, there exists an adversarial environment Env such that Algorithm 2.7 constructed from $\bar{V}_{1/2}$ has the regret lower bound*

$$\mathrm{Regret}_T(Env, u) \ge f_T^*(|u|) - O(|u| \log |u|).$$

It shows that everywhere on an exponentially growing class of comparators, the regret upper bound (Corollary 2.4) is tight up to an additive, T-independent term. More intuitively, the key message is that Corollary 2.4 tightly characterizes the performance of our main potential $\bar{V}_{1/2}$.

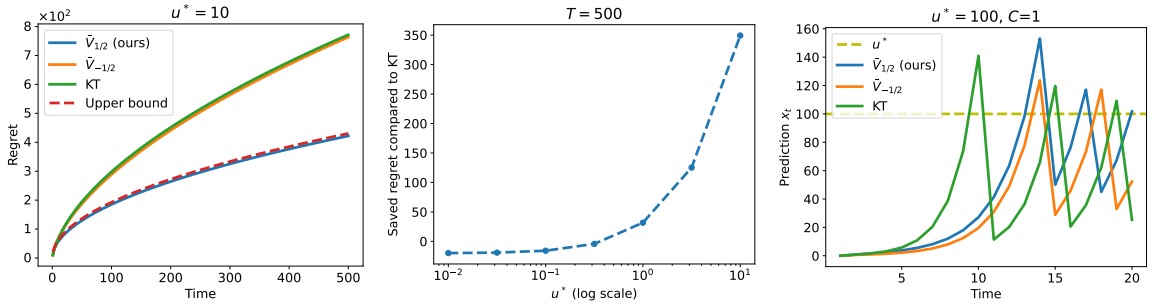
Notice that here we fix the online learning algorithm and consider the worst case environment. Results of this form are seldom studied in minimax online learning.

The reason is that, minimax online learning cares about the worst case regret (1.3), which is a real number. Therefore, both the minimax upper bound and the algorithm-independent lower bound are expressed as real numbers, whose gap is generally not hard to characterize. In contrast, we care about adaptive online learning, where upper and lower bounds are both expressed as functions. The characterization of their gap is much richer, therefore it is meaningful to study algorithm-dependent lower bounds. This is technically easier than algorithm-independent results, so we can obtain a tighter gap with the upper bound, and mathematically, the statement is easier to interpret (compare Theorem 2.5 to Theorem 2.2).

2.6 Experiment

We now test our 1D unconstrained OLO algorithm (Algorithm 2.7) on a synthetic OCO task, based on the standard reduction from OCO to OLO. Its simplicity allows us to directly compute the regret, thus clearly demonstrate the benefit of the erfi potential $\bar{V}_{1/2}$ over the existing potential $\bar{V}_{-1/2}$.

Consider the OCO problem with time invariant loss function $|x_t - u^*|$, where $u^* \in \mathbb{R}$ is a constant hidden from the online learning algorithm. Reduced into OLO (see Section 1.1), the environment Env picks the loss gradient $g_t = 1$ if $x_t \geq u^*$, while $g_t = -1$ otherwise. The most natural comparator is the hidden constant u^*, and the induced regret of OLO can be nicely interpreted as the *cumulative loss* of OCO. That is, $\mathrm{Regret}_T(Env, u^*) = \sum_{t=1}^T g_t(x_t - u^*) = \sum_{t=1}^T |x_t - u^*|$. We will test three algorithms: (i) Algorithm 2.7 constructed from $\bar{V}_{1/2}$ (our main contribution); (ii) Algorithm 2.7 constructed from $\bar{V}_{-1/2}$; and (iii) the classical *Krichevsky-Trofimov* (KT) algorithm (Orabona and Pál, 2016) which is an optimistic version of (ii) with similar guarantees. Each algorithm requires one hyperparameter: we set $C = 1$ for the first two, and set the "initial wealth" as $\sqrt{e}$ for KT, which makes a fair comparison.



(a) Fix u^* and vary T. **(b)** Fix T and vary u^*. **(c)** The progression of x_t.

Figure 2·1: One dimensional synthetic task with loss $|x_t - u^*|$. Specifically, Subfigure (b) fixes $T = 500$ and plots $\text{Regret}_T(Env, u^*)$ of KT minus $\text{Regret}_T(Env, u^*)$ of our algorithm ($\bar{V}_{1/2}$) as a function of u^*.

Since $\text{Regret}_T(Env, u^*)$ depends on both u^* and T, there are multiple ways to visualize our results. In Figure 2·1a, we fix $u^* = 10$ and plot $\text{Regret}_T(Env, u^*)$ as a function of T (lower is better). For comparison, we also plot the regret upper bound based on $\bar{V}_{1/2}$ (Corollary 2.4). Consistent with our theory, (i) the upper bound (red dashed) closely captures the actual performance of our algorithm (blue); (ii) the two baselines (orange and green) exhibit similar performance, and our algorithm improves both when $u^* = 10$.

In Figure 2·1b, we fix $T = 500$ and plot the difference between the regret of KT and our algorithm (i.e., $\text{Regret}_T(Env, u^*)|_{KT} - \text{Regret}_T(Env, u^*)|_{ours}$ as a function of u^*, higher means our algorithm improves the KT baseline by a larger margin). The obtained curve demonstrates the benefit of our special loss-regret tradeoff: while sacrificing the regret at small $|u^*|$, our algorithm significantly improves the baseline when u^* is far-away. Notably, the magnitude of $|u^*|$ represents the quality of initialization: with an oracle guess $\tilde{u}$, one can shift the origin to $\tilde{u}$, and the effective distance to u^* becomes $|\tilde{u} - u^*|$. Figure 2·1b shows that in order to beat our algorithm, the baseline has to guess u^* beforehand with error at most 1, which is hard. Therefore, our algorithm prevails in most situations.

To strengthen the intuition, let us fix $u^* = 100$ and take a closer look at the progression of predictions x_t (Figure 2·1c). Similar to both baselines, our algorithm approaches u^* with exponentially growing speed at the beginning, which is a key benefit of comparator adaptive algorithms over gradient descent (Orabona and Tommasi, 2017, Section 5). However, after overshooting, the prediction of our algorithm exhibits a much smaller "dip". This aligns with the intuition, as our algorithm allows higher $\text{Regret}_T(0)$. In other words, compared to the baselines, our algorithm has a weaker belief that the initialization is correct; instead, it believes more in the incoming information. Such a property leads to advantages when the initialization is indeed far from the optimum.

2.7 Summary

This chapter presents a framework that generates potential function candidates by solving a PDE. It reduces the amount of guessing in the current workflow, thus simplifying the discovery and analysis of more complicated potentials. To demonstrate its power, we use this framework to design a concrete algorithm – it achieves the optimal loss-regret tradeoff without any impractical doubling trick, and furthermore, attains the optimal leading constant. Such properties lead to practical advantages when a good initialization is not available.

Overall, the continuous-time perspective adopted in this chapter, in combination with a series of recent works (Drenska and Kohn, 2020b; Harvey et al., 2020), could be a powerful tool for adaptive online learning in general. Several interesting directions are left open:

- In order to further improve practicality, can we use the new potential to achieve gradient-adaptive (Cutkosky and Orabona, 2018; Cutkosky, 2019a; Mhammedi and Koolen, 2020) bounds?

- Can the PDE framework achieve adaptivity or tradeoffs in a broader range of online learning problems? For example, with bandit feedback, delays, etc. The problem with switching costs will be considered in the next Chapter.

- Is there a more principled way to handle the obtained PDEs, without enough boundary conditions? Can we automate the discovery and verification of new potentials?

2.8 Proofs

2.8.1 Generic betting results

Lemma 2.4

Proof of Lemma 2.4. We only prove the first part by induction. The proof of the second part is similar, therefore omitted. Let us restate the backward recursion (2.5),

$$V(t,S) = \min_{x\in\mathcal{X}} \max_{c\in\mathcal{C}} \left[V(t+1, S+c) - \langle c, x\rangle\right].$$

Starting from $t=0$ and $S=0$, let x_1 be the outer minimizing argument. Then, for all adversary policy a_1 such that $c_1 = a_1(x_1)$, we have $V(1,c_1) = V(1,c_1) - V(0,0) \leq \langle c_1, x_1\rangle$.

Now consider the following induction hypothesis: there exists $T \in \mathbb{N}_+$, initial bet x_1 and functions $p_2^*, \ldots, p_T^*$ such that for all $\boldsymbol{a}$,

$$\sum_{t=1}^{T} \langle c_t, x_t\rangle \geq V\left(T, \sum_{t=1}^{T} c_t\right).$$

Plugging $(t,S) = (T, \sum_{t=1}^{T} c_t)$ into the backward recursion,

$$V\left(T, \sum_{t=1}^{T} c_t\right) = \min_{x_{T+1}\in\mathcal{X}} \max_{c_{T+1}\in\mathcal{C}} \left[V\left(T+1, \sum_{t=1}^{T+1} c_t\right) - \langle c_{T+1}, x_{T+1}\rangle\right].$$

Given the value function V, there exists x_{T+1} only depending on T and $\sum_{t=1}^{T} c_t$ such

that for all c_{T+1},

$$V\left(T, \sum_{t=1}^{T} c_t\right) \geq V\left(T+1, \sum_{t=1}^{T+1} c_t\right) - \langle c_{T+1}, x_{T+1} \rangle .$$

Define the policy p^*_{T+1} in this way, we have

$$\sum_{t=1}^{T+1} \langle c_t, x_t \rangle \geq V\left(T+1, \sum_{t=1}^{T+1} c_t\right). \qquad \square$$

Lemma 2.5

Proof of Lemma 2.5. Starting from the LHS of (2.14),

$$\begin{aligned} LHS = \bar{V}\left(t+1, \sum_{i=1}^{t+1} c_i\right) &- \frac{1}{2}\left[\bar{V}\left(t+1, \sum_{i=1}^{t} c_i + 1\right) + \bar{V}\left(t+1, \sum_{i=1}^{t} c_i - 1\right)\right] \\ &+ \frac{1}{2}\left[\bar{V}\left(t+1, \sum_{i=1}^{t} c_i + 1\right) + \bar{V}\left(t+1, \sum_{i=1}^{t} c_i - 1\right)\right] - \bar{V}\left(t, \sum_{i=1}^{t} c_i\right) \\ = \bar{V}\left(t+1, \sum_{i=1}^{t+1} c_i\right) &- \frac{1}{2}\left[\bar{V}\left(t+1, \sum_{i=1}^{t} c_i + 1\right) + \bar{V}\left(t+1, \sum_{i=1}^{t} c_i - 1\right)\right] \\ &+ \bar{\nabla}_t \bar{V}\left(t+1, \sum_{i=1}^{t} c_i\right) + \frac{1}{2}\bar{\nabla}_{SS}\bar{V}\left(t+1, \sum_{i=1}^{t} c_i\right). \end{aligned}$$

The remaining task is to show

$$\bar{V}\left(t+1, \sum_{i=1}^{t+1} c_i\right) - \frac{1}{2}\left[\bar{V}\left(t+1, \sum_{i=1}^{t} c_i + 1\right) + \bar{V}\left(t+1, \sum_{i=1}^{t} c_i - 1\right)\right] \leq c_{t+1} x_{t+1}.$$

Plugging in the player's bet x_{t+1} (2.12), it suffices to show that

$$\bar{V}\left(t+1, \sum_{i=1}^{t+1} c_i\right) \leq \frac{1 + c_{t+1}}{2}\bar{V}\left(t+1, \sum_{i=1}^{t} c_i + 1\right) + \frac{1 - c_{t+1}}{2}\bar{V}\left(t+1, \sum_{i=1}^{t} c_i - 1\right),$$

which follows from the convexity of $\bar{V}$. Equality is achieved when $c_{t+1} \in \{-1, 1\}$. $\square$

Lemma 2.6

Proof of Lemma 2.6. Plugging in the definition of discrete derivative,

$$\bar{\nabla}_t \bar{V}_{1/2}(t,S) + \bar{\nabla}_{SS} \bar{V}_{1/2}(t,S)/2 = \frac{1}{2}\bar{V}_{1/2}(t,S+1) + \frac{1}{2}\bar{V}_{1/2}(t,S-1) - \bar{V}_{1/2}(t-1,S). \quad (2.15)$$

Step 1: upper bound. For clarity, define a function $f : \mathbb{R} \to \mathbb{R}$ as

$$f(z) = 2z \int_0^z \exp(x^2)dx - \exp(z^2).$$

Then, using the definition of $\bar{V}_{1/2}$, it suffices to show that

$$f\left(-\frac{1}{\sqrt{2}}\right) + f\left(\frac{1}{\sqrt{2}}\right) \leq 0,$$

and for all $t > 1$,

$$f\left(\frac{S-1}{\sqrt{2t}}\right) + f\left(\frac{S+1}{\sqrt{2t}}\right) \leq 2\sqrt{1-\frac{1}{t}} f\left(\frac{S}{\sqrt{2(t-1)}}\right).$$

The first inequality can be easily verified by computing the values of $f(1/\sqrt{2})$ and $f(-1/\sqrt{2})$. As for the second inequality, we use an existing result (Harvey et al., 2020, Lemma C.4): for all $x \in \mathbb{R}$ and $z \in [0, 1)$,

$$f\left(\frac{x-z}{\sqrt{2}}\right) + f\left(\frac{x+z}{\sqrt{2}}\right) \leq 2\sqrt{1-z^2} f\left(\frac{x}{\sqrt{2(1-z^2)}}\right).$$

Taking $x = S/\sqrt{t}$ and $z = 1/\sqrt{t}$ completes the proof.

Step 2: lower bound. From Taylor's theorem,

$$\begin{aligned}\bar{V}_{1/2}(t,S+1) = \bar{V}_{1/2}(t,S) + \nabla_S \bar{V}_{1/2}(t,S) \\ + \frac{1}{2}\nabla_{SS}\bar{V}_{1/2}(t,S) + \frac{1}{6}\nabla_{SSS}\bar{V}_{1/2}(t,S) + \frac{1}{24}\nabla_{SSSS}\bar{V}_{1/2}(t,S+a),\end{aligned}$$

$$\begin{aligned}\bar{V}_{1/2}(t,S-1) = \bar{V}_{1/2}(t,S) - \nabla_S \bar{V}_{1/2}(t,S) \\ + \frac{1}{2}\nabla_{SS}\bar{V}_{1/2}(t,S) - \frac{1}{6}\nabla_{SSS}\bar{V}_{1/2}(t,S) + \frac{1}{24}\nabla_{SSSS}\bar{V}_{1/2}(t,S-b),\end{aligned}$$

$$\bar{V}_{1/2}(t-1,S) = \bar{V}_{1/2}(t,S) - \nabla_t \bar{V}_{1/2}(t,S) + \frac{1}{2}\nabla_{tt}\bar{V}_{1/2}(t-c,S),$$

where $a, b, c \in [0,1]$. Plugging these into (2.15) and using the condition $\nabla_t \bar{V}_{1/2} = -\nabla_{SS}\bar{V}_{1/2}/2$ (since $\bar{V}_{1/2}$ is a solution of the backward heat equation), we have

$$\begin{aligned}
&\bar{\nabla}_t \bar{V}_{1/2}(t,S) + \bar{\nabla}_{SS}\bar{V}_{1/2}(t,S)/2 \\
&\qquad = \frac{1}{48}\nabla_{SSSS}\bar{V}_{1/2}(t,S+a) + \frac{1}{48}\nabla_{SSSS}\bar{V}_{1/2}(t,S-b) - \frac{1}{2}\nabla_{tt}\bar{V}_{1/2}(t-c,S).
\end{aligned}$$

It can be verified that $\nabla_{SSSS}\bar{V}_{1/2}(t,S) \geq 0$ for all (t,S), and

$$\nabla_{tt}\bar{V}_{1/2}(t,S) = \frac{C}{4}t^{-3/2}\exp\left(\frac{S^2}{2t}\right)\left(\frac{S^2}{t}+1\right).$$

Therefore,

$$\begin{aligned}
\bar{\nabla}_t \bar{V}_{1/2}(t,S) + \bar{\nabla}_{SS}\bar{V}_{1/2}(t,S)/2 &\geq -\frac{C}{8}\max_{c\in[0,1]}(t-c)^{-3/2}\exp\left(\frac{S^2}{2(t-c)}\right)\left(\frac{S^2}{t-c}+1\right) \\
&= -\frac{C}{8}(t-1)^{-3/2}\exp\left(\frac{S^2}{2(t-1)}\right)\left(\frac{S^2}{t-1}+1\right). \qquad \square
\end{aligned}$$

2.8.2 Regret upper bound

Theorem 2.1

Proof of Theorem 2.1. The proof follows from combining Lemma 2.3, Lemma 2.7 and Lemma 2.1.

Specifically, let us first guarantee the performance of the y_t sequence. For clarity, given any T, define a one dimensional function f_T as $f_T(S) = \bar{V}_{1/2}(T,S)$. Combining Lemma 2.3 and Lemma 2.7, for any $T \in \mathbb{N}_+$ and $w \in \mathbb{R}$ we have

$$\sum_{t=1}^{T}\langle g_t, z_t\rangle\, y_t - \sum_{t=1}^{T}\langle g_t, z_t\rangle\, w \leq f_T^*(w).$$

Then, due to Lemma 2.1, for all $T \in \mathbb{N}_+$ and $u \in \mathbb{R}^d$ Algorithm 2.6 guarantees

$$\mathrm{Regret}_T(u) \leq f_T^*(\|u\|) + \|u\|\sqrt{2T}.$$

The remaining task is to bound the Fenchel conjugate f_T^*. For all $w \in \mathbb{R}$,

$$f_T^*(w) = \sup_{S \in \mathbb{R}} Sw - f_T(S).$$

Let S^* be the maximizing argument. Without loss of generality (due to symmetry), assume $w \geq 0$ and therefore $S^* \geq 0$. We have

$$w = \nabla f_T(S^*) = \sqrt{2}C \int_0^{S^*/\sqrt{2T}} \exp(z^2)dz.$$

For any $x \geq 0$, consider the function $f(x) = \int_0^x \exp(z^2)dz$. It is lower bounded by $g(x) = \exp(x^2 - x) - 1$, as $f(0) = g(0)$, and

$$f'(x) = \exp(x^2) \geq \exp(x^2 - x)(2x - 1) = g'(x),$$

due to the inequality $\exp(x) \geq 2x - 1$. Therefore,

$$\frac{w}{\sqrt{2}C} = \int_0^{S^*/\sqrt{2T}} \exp(z^2)dz \geq \exp\left[\left(\frac{S^*}{\sqrt{2T}} - \frac{1}{2}\right)^2 - \frac{1}{4}\right] - 1,$$

$$S^* \leq \sqrt{2T}\left[\sqrt{\frac{1}{4} + \log\left(1 + \frac{w}{\sqrt{2}C}\right)} + \frac{1}{2}\right].$$

Now consider $f_T^*(w)$. Since $f_T(S^*) \geq -C\sqrt{T}$ and $\sqrt{x + (1/4)} \leq \sqrt{x} + (1/2)$,

$$f_T^*(w) = S^*w - f_T(S^*) \leq S^*w + C\sqrt{T} \leq C\sqrt{T} + w\sqrt{2T}\left[\sqrt{\log\left(1 + \frac{w}{\sqrt{2}C}\right)} + 1\right].$$

Combining everything completes the proof. □

2.8.3 Regret lower bound

Lemma 2.8

Proof of Lemma 2.8. Let us first generalize the unconstrained coin-betting game to allow random adversary on the coin space $\{-1, 1\}$. That is, based on past player bets $x_1, \ldots, x_t$, the adversary decides a distribution on $\{-1, 1\}$ and samples c_t from this

distribution.

Now, consider the setting where the player applies any policy $\boldsymbol{p}$ that guarantees $\text{Wealth}_T \geq -C\sqrt{T}$, and the adversary picks coin outcomes according to a Rademacher distribution: regardless of $x_1, \ldots, x_t$, the coin c_t equals -1 and 1 with probability $1/2$ respectively. Then for all $T \in \mathbb{N}_+$, let $k = \sqrt{2T \log \lambda}$.

$$\begin{aligned}
0 &= \mathbb{E}\left[\sum_{t=1}^{T} c_t x_t\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{T} c_t x_t \middle| \left|\sum_{t=1}^{T} c_t\right| \geq k\right] \mathbb{P}\left[\left|\sum_{t=1}^{T} c_t\right| \geq k\right] \\
&\qquad + \mathbb{E}\left[\sum_{t=1}^{T} c_t x_t \middle| \left|\sum_{t=1}^{T} c_t\right| < k\right] \mathbb{P}\left[\left|\sum_{t=1}^{T} c_t\right| < k\right] \\
&\geq \mathbb{E}\left[\sum_{t=1}^{T} c_t x_t \middle| \left|\sum_{t=1}^{T} c_t\right| \geq k\right] \mathbb{P}\left[\left|\sum_{t=1}^{T} c_t\right| \geq k\right] - C\sqrt{T}.
\end{aligned}$$

Applying Lemma 2.9, using $\lambda \geq \exp[(\sqrt{2}+1)/2]$ and $T \geq 8\pi\lambda^2 \log \lambda$,

$$\begin{aligned}
\mathbb{P}\left[\left|\sum_{t=1}^{T} c_t\right| \geq k\right] &\geq \sqrt{\frac{2}{\pi}} \frac{\sqrt{2\log\lambda}}{1+2\log\lambda} \lambda^{-1} - \frac{1}{\sqrt{T}} \\
&\geq \frac{1}{\sqrt{2\pi\log\lambda}} \lambda^{-1} - \frac{1}{\sqrt{T}} \geq \frac{1}{2\sqrt{2\pi\log\lambda}} \lambda^{-1}.
\end{aligned}$$

$$\mathbb{E}\left[\sum_{t=1}^{T} c_t x_t \middle| \left|\sum_{t=1}^{T} c_t\right| \geq k\right] \leq \frac{C\sqrt{T}}{\mathbb{P}\left[\left|\sum_{t=1}^{T} c_t\right| \geq k\right]} \leq 2\sqrt{2\pi}\lambda\sqrt{\log\lambda} \cdot C\sqrt{T}.$$

Therefore, for any player policy $\boldsymbol{p}$ there exists an adversary policy $\boldsymbol{a}$ which induces $|\sum_{t=1}^{T} c_t| \geq \sqrt{2T\log\lambda}$ and $\text{Wealth}_T \leq 2\sqrt{2\pi}\lambda\sqrt{\log\lambda} \cdot C\sqrt{T}$. □

The above proof critically relies on a tail lower bound for random walks, which is presented in Lemma 2.9. Compared to similar results (Mcmahan and Streeter, 2012; Orabona, 2013), it has the tight exponent (1/2) in the exponential function.

Lemma 2.9. *For all $T \in \mathbb{N}_+$, let $z_1, \ldots, z_T$ be i.i.d. Rademacher random variables.*

Then for any $k > 0$,

$$\mathbb{P}\left[\left|\sum_{t=1}^{T} z_t\right| \geq k\right] \geq \sqrt{\frac{2}{\pi}}\frac{k\sqrt{T}}{k^2+T}\exp\left(-\frac{k^2}{2T}\right) - \frac{1}{\sqrt{T}}.$$

Proof of Lemma 2.9. Due to the Central Limit Theorem, $(\sum_{t=1}^{T} z_t)/\sqrt{T}$ converges in distribution to standard normal $\mathcal{N}(0,1)$. Concretely, the nonasymptotic convergence rate can be characterized via the Berry-Esseen Theorem (Korolev and Shevtsova, 2012): Let $F_T(x)$ be the CDF of $(\sum_{t=1}^{T} z_t)/\sqrt{T}$ and $\Phi(x)$ be the standard normal CDF, then,

$$\sup_{x\in\mathbb{R}} |F_T(x) - \Phi(x)| \leq \frac{1}{2\sqrt{T}}.$$

For the tail probability of standard normal distribution, there is a standard lower bound (Gordon, 1941) through the Mills ratio, which can be verified via a derivative argument: For all $x > 0$,

$$1 - \Phi(x) \geq \frac{1}{\sqrt{2\pi}}\frac{1}{x + x^{-1}}\exp\left(-\frac{x^2}{2}\right).$$

Therefore,

$$\begin{aligned}
\mathbb{P}\left[\left|\sum_{t=1}^{T} z_t\right| \geq k\right] &= 2\cdot\left[1 - F_T(k/\sqrt{T})\right] \\
&\geq 2\cdot\left[1 - \Phi(k/\sqrt{T}) - \frac{1}{2\sqrt{T}}\right] \\
&\geq \sqrt{\frac{2}{\pi}}\frac{k\sqrt{T}}{k^2+T}\exp\left(-\frac{k^2}{2T}\right) - \frac{1}{\sqrt{T}}.
\end{aligned}$$

□

Theorem 2.2

Proof of Theorem 2.2. We start by proving the regret lower bound for one dimensional unconstrained OLO. Extension to the general d-dimensional problem will be considered later.

For the one dimensional problem, we first invoke a particular version of Lemma 2.8 on unconstrained coin-betting. Specifically, for any constants $\eta \in (0,1)$ and $u \in \mathbb{R}/\{0\}$ we define λ in Lemma 2.8 as

$$\lambda = \frac{\eta\,|u|}{2\sqrt{\pi}C}.$$

For convenience of notation we also define

$$T_0 = \frac{2\eta^2 |u|^2}{C^2} \log\left(\frac{\eta |u|}{2\sqrt{\pi}C}\right).$$

Then, Lemma 2.8 yields the following result: For all $\eta \in (0,1)$, $|u| \geq 2\sqrt{\pi}\exp[(\sqrt{2}+1)/2]\eta^{-1}C$, $T \geq T_0$ and any coin-betting player policy $\boldsymbol{p}$ that guarantees $\text{Wealth}_T \geq -C\sqrt{T}$, there exists a coin-betting adversary policy $\boldsymbol{a}$ such that in the game induced by $(\boldsymbol{p}, \boldsymbol{a})$,

1. $|\sum_{t=1}^T c_t| \geq \sqrt{2T\log\lambda}$;
2. $\text{Wealth}_T \leq \eta |u| \sqrt{2T\log\lambda}$.

Using Algorithm 2.3, we can equivalently convert OLO to coin-betting by letting $c_t = -g_t$. Then, the above result immediately translates to the following statement on one dimensional unconstrained OLO: For all $\eta \in (0,1)$, $|u| \geq 2\sqrt{\pi}\exp[(\sqrt{2}+1)/2]\eta^{-1}C$, $T \geq T_0$ and any unconstrained OLO algorithm $\mathcal{A}$ that guarantees the cumulative loss bound $\sum_{t=1}^T g_t x_t \leq C\sqrt{T}$, there exists an OLO adversary Env such that in the induced game,

1. $|\sum_{t=1}^T g_t| \geq \sqrt{2T\log\lambda}$;
2. $-\sum_{t=1}^T g_t x_t \leq \eta |u| \sqrt{2T\log\lambda}$.

Let us consider the regret of $\mathcal{A}$ in this setting with respect to comparators u and $-u$. Using the above result,

$$\begin{aligned}
\max\{\text{Regret}_T(Env, u), \text{Regret}_T(Env, -u)\} &= \sum_{t=1}^T g_t x_t + \max\left\{-\sum_{t=1}^T g_t u, \sum_{t=1}^T g_t u\right\} \\
&= \sum_{t=1}^T g_t x_t + \left|\sum_{t=1}^T g_t\right| |u| \\
&\geq (1-\eta)|u|\sqrt{2T\log\lambda} \\
&= (1-\eta)|u|\sqrt{2T\log\frac{\eta|u|}{2\sqrt{\pi}C}}.
\end{aligned}$$

Thus we have proved the desirable result when $d = 1$.

Extending this result to d-dimension follows from a standard technique: consider environments Env whose loss vectors g_t are only nonzero in one coordinate. Let

$g_t = [g_{t,1}, \ldots, g_{t,d}]$, and assume $g_{t,2} = \ldots = g_{t,d} = 0$. Then, for any OLO algorithm that operates in this environment and competes against $u = [u_1, 0, \ldots, 0]$,

$$\mathrm{Regret}_T(Env, u) = \sum_{t=1}^{T} \langle g_t, x_t \rangle - \sum_{t=1}^{T} \langle g_t, u \rangle = \sum_{t=1}^{T} g_{t,1} x_{t,1} - \sum_{t=1}^{T} g_{t,1} u_1,$$

$\|u\| = |u_1|$, and the cumulative loss satisfies $\sum_{t=1}^{T} \langle g_t, x_t \rangle = \sum_{t=1}^{T} g_{t,1} x_{t,1}$. Therefore, any d-dimensional algorithm that guarantees $\mathrm{Regret}_T(0) \leq C\sqrt{T}$ is translated into a one dimensional algorithm with the same guarantee, and our one dimensional regret lower bound can be applied. □

Theorem 2.3

Proof of Theorem 2.3. Let us first consider the upper bound. Plugging in Theorem 2.1,

$$\begin{aligned}
& \limsup_{U\to\infty} \limsup_{T\to\infty} \sup_{\|u\|=U, Env} \frac{\mathrm{Regret}_T^{\mathcal{A}_{1/2}}(Env, u)}{\|u\|\sqrt{T \log \|u\|}} \\
\leq\ & \limsup_{U\to\infty} \limsup_{T\to\infty} \sup_{\|u\|=U, Env} \left(\frac{C + 2\sqrt{2}\|u\|}{\|u\|\sqrt{\log \|u\|}} + \sqrt{2 \log \left(1 + \frac{\|u\|}{\sqrt{2}C}\right) \log^{-1} \|u\|} \right) \\
\leq\ & \lim_{U\to\infty} \frac{C + 2\sqrt{2}U}{U\sqrt{\log U}} + \lim_{U\to\infty} \sqrt{2 \log \left(1 + \frac{U}{\sqrt{2}C}\right) \log^{-1} U} = \sqrt{2}
\end{aligned}$$

As for the lower bound, we use Theorem 2.2. We first fix any C and any $\mathcal{A}$ satisfying the condition in the theorem to be proved. For all $\eta \in (0, 1)$, with $U \geq 12\eta^{-1}C$ and $T \geq 2\eta^2 U^2 C^{-2} \log(\eta U C^{-1})$,

$$\begin{aligned}
\sup_{\|u\|=U, Env} \frac{\mathrm{Regret}_T^{\mathcal{A}}(Env, u)}{\|u\|\sqrt{T \log \|u\|}} &\geq (1-\eta)\sqrt{2 \log \frac{\eta U}{2\sqrt{\pi}C} \log^{-1} U} \\
&= (1-\eta)\sqrt{2\left(1 + \frac{\log \eta}{\log U} - \frac{\log(2\sqrt{\pi}C)}{\log U}\right)}.
\end{aligned}$$

Taking lim inf on both sides, for all $\eta \in (0, 1)$,

$$\liminf_{U\to\infty} \liminf_{T\to\infty} \sup_{\|u\|=U, Env} \frac{\mathrm{Regret}_T^{\mathcal{A}}(Env, u)}{\|u\|\sqrt{T \log \|u\|}} \geq \sqrt{2}(1-\eta).$$

Rewriting this statement, we have: for all $\varepsilon \geq 0$ and $\eta \in (0, 1)$, there exists U_0

depending on ε and η such that for all $U \geq U_0$,

$$\liminf_{T\to\infty} \sup_{\|u\|=U, Env} \frac{\mathrm{Regret}_T^{\mathcal{A}}(Env, u)}{\|u\|\sqrt{T\log\|u\|}} \geq \sqrt{2} - \sqrt{2}\eta - \varepsilon.$$

Finally, using the definition of lim inf completes the proof. □

Theorem 2.5

Proof of Theorem 2.5. For convenience, let us define the function

$$h_T(S) = \bar{V}_{1/2}(T, S) + \frac{3C}{8}\exp\left(\frac{S^2}{2T}\right)\left(\frac{S^2}{T} + 1\right) + 2C.$$

Directly applying Lemma 2.10 yields the following result. For all $T \in \mathbb{N}_+$ and $S \in [-T, T]$, there exists $g_1, \ldots, g_T \in [-1, 1]$ such that (i) $-\sum_{t=1}^T g_t = S$; and (ii) Algorithm 2.7 constructed from $\bar{V}_{1/2}$ satisfies $\sum_{t=1}^T g_t x_t \geq -h_T(S)$ against loss gradients $g_{1:T}$.

Define a variable u^* as

$$u^* = h_T'(S) = \sqrt{2}C\int_0^{S/\sqrt{2T}} \exp(x^2)dx + \frac{3CS}{8T}\exp\left(\frac{S^2}{2T}\right)\left(\frac{S^2}{T} + 3\right).$$

Since S is arbitrary within the interval $[-T, T]$, u^* can take any value within $[-U, U]$, where $U = (3/8)C(T+3)\exp(T/2)$. Due to a standard result from convex analysis (Rockafellar, 2015, Theorem 23.5), $h_T(S) + h_T^*(u^*) = Su^*$. Therefore,

$$\mathrm{Regret}_T(u^*) = \sum_{t=1}^T g_t x_t - \sum_{t=1}^T g_t u^* \geq -h_T(S) + Su^* = h_T^*(u^*).$$

The remaining task is to lower bound $h_T^*(\cdot)$.

Without loss of generality, assume $u \geq 0$. Let us define a variable $\tilde{S}$ through the equation

$$u = \sqrt{2}C\int_0^{\tilde{S}/\sqrt{2T}} \exp(z^2)dz.$$

Then, using the proof of Theorem 2.1,

$$h_T^*(u) = \sup_{S\in\mathbb{R}} Su - h_T(S) \geq \tilde{S}u - h_T(\tilde{S}) = f_T^*(u) - \frac{3C}{8}\exp\left(\frac{\tilde{S}^2}{2T}\right)\left(\frac{\tilde{S}^2}{T}+1\right) - 2C,$$

and

$$\tilde{S} \leq \sqrt{2T}\left[\sqrt{\log\left(1+\frac{u}{\sqrt{2}C}\right)}+1\right].$$

Combining the above completes the proof. □

The above proof relies on the following algorithm-dependent wealth upper bound for coin-betting.

Lemma 2.10. *Consider unconstrained coin-betting. For all $T \in \mathbb{N}_+$ and $S \in [-T, T]$, we can construct $c_1 \in \mathcal{C}$ and $c_2, \ldots, c_T \in \{-1, 1\}$ such that*

1. *$\sum_{t=1}^T c_t = S$;*
2. *If the player applies Algorithm 2.5 constructed from $\bar{V}_{1/2}$ and the adversary plays the aforementioned coin sequence $c_{1:T}$, then*

$$\text{Wealth}_T \leq \bar{V}_{1/2}(T, S) + \frac{3C}{8}\exp\left(\frac{S^2}{2T}\right)\left(\frac{S^2}{T}+1\right) + 2C.$$

Proof of Lemma 2.10. We first construct the coin sequence. For all $S \in [-T, T]$, there exists an integer $\tilde{S}$ such that $|\tilde{S}| \leq T$, $(|\tilde{S}|+1) \bmod 2 = T \bmod 2$ and $|S - \tilde{S}| \leq 1$. We define the coins using three phases.

1. $c_1 = S - \tilde{S}$;
2. For all $1 < t \leq T - |\tilde{S}|$, let $c_t = \text{sign}(c_1) \cdot (-1)^{t-1}$;
3. If $\tilde{S} \neq 0$, then for all t such that $T - |\tilde{S}| < t \leq T$, let $c_t = \tilde{S}/|\tilde{S}|$.

Based on this coin sequence, there are three immediate observations:

1. The sum of coins from the second phase is 0, and the sum of coins from the third phase is $\tilde{S}$; therefore, $\sum_{t=1}^T c_t = S$.
2. If $\tau \leq T - |\tilde{S}|$ then $|\sum_{t=1}^\tau c_t| \leq 1$.

3. If $T - |\tilde{S}| < \tau \leq T$ then $|\sum_{t=1}^{\tau} c_t| = |S| - T + \tau$.

Next, we derive the wealth upper bound induced by such a coin sequence and the player policy (Algorithm 2.5). Starting from the first round, $x_1 = 0$, therefore $\text{Wealth}_1 = 0$. $\text{Wealth}_1 = \bar{V}_{1/2}(1, c_1) - \bar{V}_{1/2}(1, c_1) \leq \bar{V}_{1/2}(1, c_1) - \bar{V}_{1/2}(1, 0) = \bar{V}_{1/2}(1, c_1) + C$. Considering the rest of the rounds, there are two cases: (i) $|S| \leq \sqrt{T}$; (ii) $|S| > \sqrt{T}$.

Case (i) In this case we first show that for all integer τ in $[1 : T]$, $|\sum_{t=1}^{\tau} c_t| \leq \sqrt{\tau}$. Due to the second observation above, this condition holds for all $\tau \leq T - |\tilde{S}|$, and we only need to focus on $T - |\tilde{S}| < \tau \leq T$ (the third phase) where $|\sum_{t=1}^{\tau} c_t| = |S| - T + \tau \leq \sqrt{T} - T + \tau$; since $T - \sqrt{T} \geq \tau - \sqrt{\tau}$, we further have $|\sum_{t=1}^{\tau} c_t| \leq \sqrt{\tau}$. Based on this result, telescoping Lemma 2.5 (notice that equality is achieved) and using Lemma 2.6, we have

$$\begin{aligned}
\text{Wealth}_T &\leq \bar{V}_{1/2}\left(T, \sum_{t=1}^{T} c_t\right) + C + \frac{C}{8}\sum_{t=1}^{T-1} t^{-3/2} \exp\left(\frac{(\sum_{i=1}^{t} c_i)^2}{2t}\right)\left(\frac{(\sum_{i=1}^{t} c_i)^2}{t} + 1\right) \\
&\leq \bar{V}_{1/2}\left(T, \sum_{t=1}^{T} c_t\right) + C + \frac{\sqrt{e}C}{4}\sum_{t=1}^{T-1} t^{-3/2} \leq \bar{V}\left(T, \sum_{t=1}^{T} c_t\right) + \left(\frac{3\sqrt{e}}{4} + 1\right) C.
\end{aligned}$$

Case (ii) In this case we show that for all integer τ in $[1 : T]$, $|\sum_{t=1}^{\tau} c_t|/\sqrt{\tau} \leq |\sum_{t=1}^{T} c_t|/\sqrt{T}$. Similar to Case (i), we consider $\tau \leq T - |\tilde{S}|$ and $T - |\tilde{S}| < \tau \leq T$ separately. When $\tau \leq T - |\tilde{S}|$, we have $|\sum_{t=1}^{\tau} c_t|/\sqrt{\tau} \leq 1 \leq |S|/\sqrt{T} = |\sum_{t=1}^{T} c_t|/\sqrt{T}$. On the other hand, when $T - |\tilde{S}| < \tau \leq T$ it suffices to show that

$$\frac{|S| - T + \tau}{\sqrt{\tau}} \leq \frac{|S|}{\sqrt{T}}.$$

The LHS monotonically increases with respect to τ, and when $\tau = T$ the inequality holds with equality. In summary, the required condition $|\sum_{t=1}^{\tau} c_t|/\sqrt{\tau} \leq |\sum_{t=1}^{T} c_t|/\sqrt{T}$ holds for all $\tau \in [1 : T]$.

Based on this result, telescoping Lemma 2.5 and using Lemma 2.6, we have

$$\begin{aligned}
\text{Wealth}_T &\leq \bar{V}_{1/2}\left(T, \sum_{t=1}^{T} c_t\right) + C + \frac{C}{8} \exp\left(\frac{(\sum_{i=1}^{T} c_i)^2}{2T}\right)\left(\frac{(\sum_{i=1}^{T} c_i)^2}{T} + 1\right)\sum_{t=1}^{T-1} t^{-3/2} \\
&\leq \bar{V}_{1/2}\left(T, \sum_{t=1}^{T} c_t\right) + C + \frac{3C}{8} \exp\left(\frac{(\sum_{i=1}^{T} c_i)^2}{2T}\right)\left(\frac{(\sum_{i=1}^{T} c_i)^2}{T} + 1\right).
\end{aligned}$$

Combining Case (i) and Case (ii) completes the proof. $\square$

Chapter 3

Comparator adaptivity with Switching Costs

This chapter is based on (Zhang et al., 2022a,c), and naturally extends the PDE framework from the previous chapter. There are two goals:

- We aim to achieve comparator adaptivity in a variant of the standard OCO setting, with switching costs. Such a setting finds plenty of applications in downstream sequential decision making tasks with *states* and *dynamics* (Agarwal et al., 2019). From the perspective of online learning, the existence of switching costs penalizes the typical optimism in adaptive algorithms, which requires a delicate tradeoff in algorithm design.

- On the methodology, we aim to show that the idea of continuous time scaling leads to not only better quantitative rates (as shown in Chapter 2), but also clear intuition on the inherent connections between different online learning problems. This allows naturally transferring algorithmic insights from the generic OCO problem to its variants.

Section 3.1 motivates the problem and summarizes our contributions. Section 3.2 surveys existing results. Section 3.3 presents the first comparator adaptive algorithm for OCO with switching costs, due to (Zhang et al., 2022a), which does not require the continuous time analysis. Section 3.4 improves this result using a streamlined algorithm discovered in continuous time; this is due to (Zhang et al., 2022c). Section 3.5 extends

these results on OCO to the expert problem, also with switching costs. Experiments are presented in Section 3.6. Section 3.7 concludes this chapter and discusses future directions. All the proofs are deferred to Section 3.8.

Setting We consider *OCO with switching costs*, which is a variant of the standard OCO problem studied in the previous chapter.

Definition 7 (OCO with switching costs)**.** *OCO with switching costs is the following variant of OCO: after the t-th round, besides suffering the instantaneous loss $l_t(x_t)$, the learning agent also suffers a switching cost $\lambda \|x_t - x_{t-1}\|_p$, where the weight λ and the norm $\|\cdot\|_p$ are known to the agent at the beginning. Without loss of generality, $x_0 = 0$.*

The goal of the agent is still bounding the static regret with respect to fixed comparators selected in hindsight. With switching costs, such an objective is called the augmented regret:

$$\sum_{t=1}^{T} l_t(x_t) + \lambda \sum_{t=1}^{T-1} \|x_{t+1} - x_t\|_p - \sum_{t=1}^{T} l_t(u).$$

It is immediately clear that the standard reduction from OCO to OLO is still applicable. Furthermore, due to the polar decomposition trick (surveyed in Section 2.2), most of the results will be developed for the 1D setting, where the augmented regret (for the reduced OLO problem) simplifies to

$$\mathrm{Regret}_T^{\lambda}(Env, u) := \sum_{t=1}^{T} g_t(x_t - u) + \lambda \sum_{t=1}^{T-1} |x_{t+1} - x_t| .$$

Taking the supremum over the environment, we will primarily bound

$$\mathrm{Regret}_T^{\lambda}(u) := \sup_{Env} \left(\sum_{t=1}^{T} g_t(x_t - u) + \lambda \sum_{t=1}^{T-1} |x_{t+1} - x_t| \right). \tag{3.1}$$

As for the rest of the problem setting, the time horizon T is unknown, and the loss functions are assumed to be G-Lipschitz (i.e., Assumption 1) with respect to

$\|\cdot\|_q$, where $\|\cdot\|_q$ is the dual norm of $\|\cdot\|_p$. Specifically, $p = 1$ and 2 will be considered. Unlike the previous chapter, we do not assume $G = 1$ in order to make it easier to see the "unit" throughout the analysis. Also, same as the previous chapter, we do not incorporate any prior $\tilde{u}$, which means that algorithms are initialized at the origin, and the comparator complexity is characterized by $\|u\|_p$.

Our goal is to achieve a comparator adaptive (Type 1 in Section 1.4) augmented regret bound: omitting the dependence on G, λ and p,

$$\sum_{t=1}^{T} l_t(x_t) + \lambda \sum_{t=1}^{T-1} \|x_{t+1} - x_t\|_p - \sum_{t=1}^{T} l_t(u) = \tilde{O}(\|u\|_p \sqrt{T}).$$

Section 3.5 extends our results on OCO to another classical online learning problem called *Learning with Expert Advice* (LEA). The setting will be introduced there for clarity.

Notation This chapter involves a potential-based analysis similar to the previous chapter, but slightly more involved. Let us further define discrete derivatives to simplify the writing.

$$\begin{aligned} \bar{\nabla}_t V(t,S) &:= V(t,S) - V(t-1,S), \\ \bar{\nabla}_S V(t,S) &:= \frac{1}{2}\left[V(t,S+1) - V(t,S-1)\right], \\ \bar{\nabla}_{SS} V(t,S) &:= V(t,S+1) + V(t,S-1) - 2V(t,S). \end{aligned} \tag{3.2}$$

Conventional derivatives are still denoted without the bar, e.g., $\nabla_t V(t,S)$ and $\nabla_S V(t,S)$.

3.1 Motivation and contribution

Online learning with switching costs is a classical research topic. Practically, switching costs are useful whenever the smooth operation of a sequential decision system is favored, such as in network routing, control of electrical grid, portfolio management with transaction costs, etc. Recently they also show up as submodules in online learning problems with *long term effects*, such as *nonstochastic control* (Agarwal et al., 2019). The rationale is that, if the negative effect of our decisions could remain in a state and propagate to the future, then we should be more conservative, thus change our decisions slowly between consecutive rounds.

It is well known that OGD can incorporate switching costs by simply scaling its learning rate (Anava et al., 2015a). However, it is much harder to extend comparator adaptive algorithms accordingly. Just like all kinds of adaptive algorithms (Duchi et al., 2011; Daniely et al., 2015), the key idea of comparator adaptivity is to quickly respond to the incoming information and hedge aggressively. Switching costs, on the other hand, encourage the agent to stay still. Therefore, achieving comparator adaptivity with switching costs requires a delicate balance between these two opposite considerations. A more quantitative discussion of this challenge will be provided in Section 3.3.

Similar tradeoffs between adaptivity and switching costs have led to intriguing results in the past. For example, Gofer (2014) showed that the gradient variance adaptivity (Type 2 in Section 1.4) well-studied in the switching-free setting is impossible with normed switching costs, thus establishing a clear separation caused by the latter. Daniely and Mansour (2019) showed that a common analytical technique for switching costs is incompatible to the local adaptivity (Type 4 in Section 1.4). In this chapter, we investigate whether comparator adaptivity can be extended to this setting, which was unclear a priori.

Result and contribution We develop two comparator adaptive algorithms for OCO with switching costs. Both of them are concretely developed for the 1D setting, i.e., bounding (3.1). The extension to higher dimensions is fairly standard (Orabona, 2019, Chapter 9), which will be discussed in Section 3.4.4.

- The first algorithm was proposed in (Zhang et al., 2022a), which is based on the traditional, betting-fraction-type analysis (Orabona and Pál, 2016). Given any hyperparameter $C > 0$, it guarantees for all $T \in \mathbb{N}_+$ and $u \in \mathbb{R}$,[1]

$$\mathrm{Regret}_T^{\lambda}(u) \leq (G+\lambda) \cdot \left[C + |u|\, O\left(\sqrt{T} \log(|u|\, C^{-1} T) \right) \right].$$

  Despite being the first to achieve comparator adaptivity with switching costs, it does not match the optimal logarithmic factors in the switching-free setting (compare it to Theorem 2.1), thus fails to achieve Pareto optimality.

- The second algorithm was proposed in (Zhang et al., 2022c), which we will emphasize in this chapter. Still with a hyperparameter $C > 0$, the 1D algorithm guarantees

$$\mathrm{Regret}_T^{\lambda}(u) \leq \sqrt{(4\lambda G + 2G^2)T} \left[C + |u| \left(\sqrt{4 \log\left(1 + \frac{|u|}{C} \right)} + 2 \right) \right],$$

  which improves the first algorithm by simultaneously achieving several additional forms of optimality.

  Notably, the improvement is due to a novel *dual space scaling* strategy. This is actually not guessed, but systematically discovered by a continuous-time analysis similar to the previous chapter. In the continuous-time limit, it becomes

[1] Although formally this comparator adaptive result (Theorem 3.1) is proved on a bounded domain $[0, \bar{R}]$, it is based on an unconstrained algorithm and the standard technique to impose constraints (Algorithm 3.3). It is not hard to extract the underlying unconstrained algorithm, which satisfies this statement. Furthermore, compared to Theorem 3.1, here we set the regularization weight $\gamma = 0$, and the confidence parameter is denoted by C rather than ε for easier comparison with other comparator adaptive algorithms.

evident what kinds of algorithmic structures from the switching-free setting are transferable to the setting with switching costs. Indeed, revealing generalizable knowledge is a key benefit of the continuous-time analysis, which was not demonstrated in the previous chapter. As an added bonus, both this algorithm and its analysis are considerably simpler than the first algorithm we propose.

Section 3.5 extends these results to the expert problem with switching costs, achieving the first comparator adaptive regret bound there. Even without switching costs, we improve existing comparator adaptive bounds (for the expert problem) by a better divergence characterization.

Complementing these theoretical results, we test our two OCO algorithms in a portfolio management task with transaction costs, with both synthetic and real datasets.

3.2 Related work

For the background of comparator adaptive online learning, the reader is referred to Section 1.4 and 2.2. Here we focus on surveying prior works related to switching costs.

Switching cost Switching costs in online decision making have been studied from many different angles. For example, besides online learning, the online algorithm community has investigated settings like *smoothed online optimization* (Chen et al., 2018; Goel et al., 2019; Li et al., 2020) and *convex body chasing* (Bubeck et al., 2019; Sellke, 2020), where the loss function l_t is observed *before* the agent picks the prediction x_t. There, the switching cost is the key consideration that prevents the trivial strategy $x_t \in \arg\min_x l_t(x)$. As for online learning, an additional complication is that x_t (e.g., the investment portfolio) should be selected without knowing l_t (e.g., tomorrow's stock price).

Even within online learning, there are several ways to model switching costs. In cases like network routing, every switch means changing the packet route, which can be costly. Therefore, one needs a *lazy* agent whose amount of switches (or its expectation) (Kalai and Vempala, 2005; Geulen et al., 2010; Altschuler and Talwar, 2018; Chen et al., 2020; Sherman and Koren, 2021) is as low as possible – a good modeling candidate is $\mathbf{1}[x_t \neq x_{t+1}]$. Alternatively, one could take a *smooth* view (Andrew et al., 2013; Bhaskara et al., 2021; Wang et al., 2021; Zhang et al., 2021) where the agent can perform as many switches as it wishes, as long as the cumulative distance of switching is low – in this view, switching costs can be a norm $\|x_t - x_{t+1}\|$ or its smoothed variant $\|x_t - x_{t+1}\|^2$. We will specifically consider switching costs modeled by the L_1 and L_2 norms.

Despite these results on minimax online learning, existing works on the combination of adaptivity and switching costs are quite sparse. As one should carefully trade off these two opposite requirements, there have been interesting impossibility results (Gofer, 2014; Daniely and Mansour, 2019), highlighted in the previous section. Adding to this topic but in an opposite direction, we will show that comparator adaptivity can indeed by achieved.

Relation to downstream problems More generally, incorporating switching costs amounts to considering a *history-dependent* adversary: it can pick loss functions that depend not only on the instantaneous prediction x_t, but also on the previous prediction x_{t-1}. One could further generalize this setting to *online learning with memory* (Cesa-Bianchi et al., 2013; Anava et al., 2015a), where the loss depends on a fixed-length prediction history, and finally to *dynamical systems* (Agarwal et al., 2019; Simchowitz et al., 2020; Simchowitz, 2020), where the entire history matters. In fact, a common procedure in the field of nonstochastic control (Agarwal et al., 2019) is to bound the risk in the future by a properly scaled switching cost. Achieving

comparator adaptivity with switching costs can benefit these important downstream problems as well; for example, by making algorithms "easy to combine" (Cutkosky, 2019b, 2020; Zhang et al., 2022a).

3.3 The first solution

Now we are ready to dive into the details. Before presenting any algorithm, we first discuss the challenge of our task from a more quantitative perspective. In some sense, it justifies the complications in our algorithms and their analysis.

3.3.1 Quantitative challenge

Consider the 1D worst case augmented regret defined in (3.1), copied below.

$$\mathrm{Regret}_T^{\lambda}(u) := \sup_{Env} \left(\sum_{t=1}^{T} g_t(x_t - u) + \lambda \sum_{t=1}^{T-1} |x_{t+1} - x_t| \right).$$

Neglecting the dependence on G, our goal is to show a comparator adaptive bound $\tilde{O}\left(|u|\sqrt{\lambda T}\right)$.

For minimax algorithms like OGD on a bounded domain, one can use scaled adaptive learning rates $\eta_t \propto 1/\sqrt{\lambda t}$ to ensure that both sums on the RHS are $O\left(\sqrt{\lambda T}\right)$ regardless of the environment, thus obtaining a minimax optimal $O\left(\sqrt{\lambda T}\right)$ augmented regret bound. However, such a divide-and-conquer approach does not apply to comparator adaptive algorithms, as one cannot *separately* show the desirable bound on the two sums. To see this, suppose one could guarantee the second sum alone is at most $1+|u|\, O\left(\sqrt{T \log(|u|\, T)}\right)$, which is the form of the standard comparator adaptive regret bound without switching costs (see Chapter 2); here we only focus on the dependence on $|u|$ and T. Since this cumulative switching cost is an algorithmic quantity *independent of the comparator*, we can take infimum with respect to u and obtain a "budget" of 1 for this sum. Following this argument, $|x_T| \le |x_1| + \sum_{t=1}^{T-1} |x_t - x_{t+1}| = O(1)$. That

is, the algorithm should only predict around the origin, which clearly leads to large regret with respect to far-away comparators, under certain loss sequences.

The challenge can be motivated in another way. As shown in (Orabona, 2019, Figure 9.1), the one-step switching cost $|x_t - x_{t+1}|$ of comparator adaptive algorithms can grow exponentially with respect to t, whereas such a quantity is uniformly bounded in OGD. In fact, the exponential growth is the key mechanism for comparator adaptive algorithms to cover an unconstrained domain fast enough (thus improving minimax algorithms). This is however problematic when switching is also penalized, as one can no longer control the switching cost by uniformly scaling $|x_t - x_{t+1}|$.

3.3.2 Algorithm and bound

Addressing the above challenge, our first contribution is Algorithm 3.1, which is the first comparator adaptive algorithm for OLO with switching costs. Here we still consider the 1D setting. Its higher dimensional extension will be discussed in Section 3.4.4, similar to the standard techniques from (Orabona, 2019, Chapter 9).

Algorithm 3.1 needs three inputs.

- The confidence hyperparameter ε, which serves the same purpose as the hyperparameter C in Chapter 2.
- The regularization weight γ, which is needed for the extension to $\mathbb{R}^d$ with L_2 norm switching costs. For interpretability (and easier comparison with our second algorithm), one can always let $\gamma = 0$.
- A bounded domain $\mathcal{V}_{1d} = [0, \bar{R}]$. This leads to a cumulative switching cost bound (the second part of Theorem 3.1), which is critical for further constructing *locally adaptive* algorithms (Type 4 in Section 1.4) with switching costs.[2] The

[2]This is beyond the scope of this dissertation. The reader is referred to (Zhang et al., 2022a) for this construction, including its application to the *tracking control* of linear systems.

Algorithm 3.1 Our first algorithm: 1D comparator adaptive OLO with switching costs.

Require: Hyperparameters $\gamma \geq 0$ and $\varepsilon > 0$; a 1D domain $\mathcal{V}_{1d} = [0, \bar{R}]$.

1: Initialize internal variables as $\text{Wealth}_0 = \varepsilon$, and $\beta_1, x_1, \tilde{x}_1 = 0$. Define $C = G + \lambda + \gamma$.

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

3: Make a prediction x_t, observe a loss gradient g_t. Define the surrogate loss $\tilde{g}_t$ as

$$\tilde{g}_t = \begin{cases} g_t, & \text{if } g_t \tilde{x}_t \geq g_t x_t, \\ 0, & \text{otherwise.} \end{cases}$$

4: Let $\hat{\beta}_{t+1} = -\sum_{i=1}^{t} \tilde{g}_i / (2C^2 t)$. Define $\mathcal{B}_{t+1} = [0, 1/(C\sqrt{2t})]$ and let $\beta_{t+1} = \Pi_{\mathcal{B}_{t+1}}(\hat{\beta}_{t+1})$.

5: Assign Wealth_t as the solution to the following equation (uniqueness shown in Lemma 3.1),

$$\text{Wealth}_t = (1 - \tilde{g}_t \beta_t - \gamma \beta_t / \sqrt{t}) \text{Wealth}_{t-1} - \lambda |\beta_t \text{Wealth}_{t-1} - \beta_{t+1} \text{Wealth}_t|. \quad (3.3)$$

6: Let $\tilde{x}_{t+1} = \beta_{t+1} \text{Wealth}_t$ and $x_{t+1} = \Pi_{\mathcal{V}_{1d}}(\tilde{x}_{t+1})$.

7: **end for**

comparator adaptive regret bound we care about in this chapter (the first part of Theorem 3.1) does not rely on the domain being bounded – it is fairly straightforward to extract an unconstrained algorithm from Algorithm 3.1, with a comparator adaptive regret bound on $\mathcal{X} = \mathbb{R}$.

Essentially, Algorithm 3.1 is based on the betting fraction framework from (Orabona and Pál, 2016), but with substantial modifications. To get the gist of this algorithm, let us briefly ignore the surrogate loss $\tilde{g}_t$ from Line 3 and the projection of $\tilde{x}_{t+1}$ from Line 6 (i.e., assume $\bar{R} = \infty$). With $g_t = \tilde{g}_t$ and $x_{t+1} = \tilde{x}_{t+1}$, Algorithm 3.1 becomes an OLO algorithm on $\mathbb{R}_+$ with predictions recommended by the following betting scheme: A bettor has money Wealth_t in the t-th round. After choosing a betting fraction β_{t+1}, he bets money $x_{t+1} = \beta_{t+1} \text{Wealth}_t$ on the next loss gradient g_{t+1}. The favorable outcome is $g_{t+1} x_{t+1}$ being negative which means the OLO algorithm suffers *negative loss*. Therefore, after observing g_{t+1}, the bettor treats $-g_{t+1} x_{t+1}$ as the money he

gains and updates his wealth accordingly. Since large switching is also undesirable, the bettor further loses money proportional to the change of his betting amount; this is an important and novel step in our approach. Using this procedure, regret minimization is converted to wealth maximization. By choosing the betting fraction β_t properly, one can obtain a controlled augmented regret.

Rigorously, we first prove that the wealth update step in Algorithm 3.1 is well-posed.

Lemma 3.1. *For all $t \geq 1$, Equation (3.3) has a unique solution and the solution is positive.*

Then, the formal guarantee of Algorithm 3.1 is the following. The proof, including several auxiliary lemmas, are deferred to Section 3.8. Without loss of generality, let $x_{T+1} = 0$.

Theorem 3.1. *For all $\gamma \geq 0$ and $0 < \varepsilon \leq G\bar{R}$, applying Algorithm 3.1 yields the following guarantee.*

1. *For all $T \in \mathbb{N}_+$ and $u \in \mathcal{V}_{1d}$, with the constant $C = G + \lambda + \gamma$,*

$$\sum_{t=1}^{T} \left(g_t x_t - g_t u + \lambda \left| x_t - x_{t+1} \right| + \frac{\gamma}{\sqrt{t}} \left| x_t \right| \right) \leq \varepsilon + uC\sqrt{2T} \left(\frac{3}{2} + \log \frac{\sqrt{2} uCT^{5/2}}{\varepsilon} \right).$$

2. *For all $a \leq b$, $\sum_{t=a}^{b} \left| x_t - x_{t+1} \right| \leq 48\bar{R}\sqrt{b - a + 1}$.*

The highlights of Theorem 3.1 are summarized as follows.

1. Part 1 is the first comparator adaptive bound for OLO with switching costs: the augmented regret grows almost linearly in $|u|$ which is the optimal rate up to logarithmic factors (see Chapter 2). In other words, without knowing the optimal comparator u^* in advance, Algorithm 3.1 automatically adapts to it, and the performance bound almost matches the optimally-tuned OGD whose learning

rate depends on u^*. Note that the latter is a hypothetical (unimplementable) baseline, since the optimal comparator u^* in hindsight is unknown before all the losses are revealed. Nonetheless, our algorithm is still able to (nearly) match it using an implementable procedure.

Furthermore, Part 1 does not need a bounded domain; the same bound holds even with $\bar{R} = \infty$, making Algorithm 3.1 an appealing approach for general unconstrained settings as well.

2. As for Part 2, we bound the switching costs alone over any time interval, which is technically nontrivial. Our surrogate loss $\tilde{g}_t$ (Line 3) is due to an existing black-box reduction from unconstrained OLO to constrained OLO (Section 2.3). However, the proof of Part 2 requires a *non-black-box* use of this procedure: we investigate how using the surrogate loss $\tilde{g}_t$ instead of the true loss g_t changes the growth rate of Wealth_t, an internal quantity of the unconstrained OLO algorithm. This is the first analysis that takes such a perspective, and the techniques could be of separate interest.

Comparing Part 1 to the comparator adaptive regret bounds for the switching-free setting (surveyed in Section 2.2), one could also see a limitation of this result: the logarithmic factors are not optimal. With a cumulative loss budget $\varepsilon = 1$, the optimal comparator adaptive regret bound is $O(|u| \sqrt{T \log(|u| T)})$, whereas Part 1 above is $O(|u| \sqrt{T} \log(|u| T))$. That is, Part 1 does not achieve the Pareto optimal frontier of the loss-regret tradeoff. To close this gap, we develop another algorithm with streamlined intuition from the continuous time.

3.4 Better algorithm from discretization

Same as the first algorithm, we consider the 1D setting, with the goal of bounding the worst case augmented regret (3.1). The difference is that, instead of using betting

fractions, we now follow the potential framework explored by a parallel line of works (McMahan and Orabona, 2014; Foster et al., 2018; Mhammedi and Koolen, 2020; Zhang et al., 2022b), similar to Chapter 2.

3.4.1 Switching-adjusted potential

To recap, potential algorithms are defined by a potential function $V(t, S)$, where t represents the time index, and S represents a "sufficient statistic" that summarizes the history. In each round, the algorithm computes $S_{t-1} = -\sum_{i=1}^{t-1} g_i/G$, and the prediction x_t is the derivative $\nabla_S V$ evaluated at (t, S_{t-1}). We will specifically consider Algorithm 3.2, which is a variant based on the discrete derivative $\bar{\nabla}_S V$, cf. (3.2). It is essentially the same as Algorithm 2.7 in Chapter 2.

Algorithm 3.2 Our second algorithm: 1D comparator adaptive OLO with switching costs.

Require: A hyperparameter $C > 0$, the Lipschitz constant G, and a potential function $V(t, S)$ that implicitly depends on λ and G. Initialize $S_0 = 0$.

1: **for** $t = 1, 2, \ldots$ **do**
2: Predict $x_t = \bar{\nabla}_S V(t, S_{t-1})$, and receive the loss gradient g_t. Let $S_t = S_{t-1} - g_t/G$.
3: **end for**

One could think of the potential framework as the dual approach of FTRL – the potential function and the regularizer are naturally Fenchel conjugates. While the FTRL analysis relies on a one-step regret bound on the *primal space* (the domain $\mathcal{X}$, cf. (Orabona, 2019, Lemma 7.1)), the potential framework constructs a similar one-step relation on the *dual space* (the space of S_t, cf. (Zhang et al., 2022b, Lemma 3.1)). Along this interpretation, our key idea is to incorporate switching costs by *scaling on the dual space*, rather than only on the primal space. That is, given a potential function that works without switching costs, we scale the sufficient statistic sent to its second argument by a function of λ.

To better demonstrate this idea, let us first consider a quadratic potential $V(t, S) =$

$(1/2) \cdot CGS^2$. The potential method suggests the prediction $x_t = \nabla_S V(t, S_{t-1}) = C\sum_{i=1}^{t-1} g_i = x_{t-1} - Cg_{t-1}$, which is simply OGD with learning rate C. Scaling on the primal space means scaling V directly, while scaling on the dual space means scaling the sufficient statistic S. It is clear that both cases are *equivalent* to scaling the effective learning rate, which is the standard way to incorporate switching costs in bounded domain gradient descent. In other words, for this gradient descent potential, the two types of scaling are essentially the same.

Now, to achieve optimal comparator adaptivity, we need a better potential where scaling on the dual space actually makes a difference. With a parameter α that will eventually depend on λ, we consider Algorithm 3.2 induced by the potential

$$V_\alpha(t, S) = C\sqrt{\alpha t}\left[2\int_0^{S/\sqrt{4\alpha t}}\left(\int_0^u \exp(x^2)dx\right)du - 1\right]. \tag{3.4}$$

When the Lipschitz constant $G = 1$, Chapter 2 showed that $\alpha = 1/2$ leads to comparator adaptivity without switching costs. Here we use $\alpha = 4\lambda G^{-1} + 2$, which amounts to scaling *both* the primal space and the dual space: on the primal space, we scale up the overall prediction by $\Theta(\sqrt{\lambda G^{-1} + 1})$, and on the dual space we scale down the sufficient statistic S by $\Theta(1/\sqrt{\lambda G^{-1} + 1})$. The latter gives us the optimal comparator adaptive bound (i.e., Pareto-optimal rate in $|u|$ and T), while the former helps us obtain the optimal rate in λ. Due to incorporating λ into the potential function V_α, we call our approach the *switching-adjusted potential method*.

Although the dual space scaling strategy and the particular structure of V_α may seem mysterious at first glance, they are actually *derived* from a continuous-time analysis. To proceed, we will first present the performance guarantee in the next subsection, and then revisit the derivation of this strategy in Section 3.4.3.

3.4.2 Optimal comparator adaptive bound

Despite its simplicity, our second algorithm improves the first one (Theorem 3.1) by a considerable margin.

Theorem 3.2. *If $\alpha = 4\lambda G^{-1} + 2$, then Algorithm 3.2 induced by the potential V_α, cf., (3.4), guarantees*

$$\mathrm{Regret}_T^\lambda(u) \le \sqrt{(4\lambda G + 2G^2)T}\left[C + |u|\left(\sqrt{4\log\left(1 + \frac{|u|}{C}\right)} + 2\right)\right],$$

for all $u \in \mathbb{R}$ and $T \in \mathbb{N}_+$.

Theorem 3.2 simultaneously achieves several forms of optimality.

1. Pareto-optimal loss-regret tradeoff: considering the dependence on u and T, $\mathrm{Regret}_T^\lambda(u) = O\left(|u|\sqrt{T\log|u|}\right)$, while the *cumulative loss* $\mathrm{Regret}_T^\lambda(0)$ satisfies $\mathrm{Regret}_T^\lambda(0) = O(\sqrt{T})$. An existing lower bound (Theorem 2.2) shows that even without switching costs, all algorithms satisfying a $O(\sqrt{T})$ loss bound must suffer a $\Omega\left(|u|\sqrt{T\log|u|}\right)$ regret bound. In this sense, our algorithm attains a *Pareto-optimal* loss-regret tradeoff, in a strictly generalized setting with switching costs.

2. On T alone: $\mathrm{Regret}_T^\lambda(u) = O(\sqrt{T})$. Despite achieving comparator adaptivity, the asymptotic rate on T is still the optimal one, matching the well-known minimax lower bound.

3. On λ alone: $\mathrm{Regret}_T^\lambda(u) = O(\sqrt{\lambda})$. Our bound has the optimal dependence on the switching cost weight (Geulen et al., 2010, Theorem 5).

To compare Theorem 3.2 to the result of our first algorithm (Part 1 of Theorem 3.1), we have to convert them to the same loss-regret tradeoff, i.e., both guaranteeing $\mathrm{Regret}_T^\lambda(0) = O(1)$ or $\mathrm{Regret}_T^\lambda(0) = O(\sqrt{T})$. Here we take the first approach. Let

us only consider the dependence on u and T. By a doubling trick, the bound of Theorem 3.2 can be converted to $C + |u| \, O\left(\sqrt{T \log(C^{-1} |u| \, T)}\right)$, which improves the rate $C + |u| \, O\left(\sqrt{T} \log(C^{-1} |u| \, T)\right)$ from Theorem 3.1.[3] Specifically, the converted upper bound also attains Pareto-optimality in this regime, i.e., matching the lower bound $\Omega\left(|u| \sqrt{T \log(|u| \, T)}\right)$ in (Orabona, 2013), whereas Theorem 3.1 does not.

The proof of Theorem 3.2 is deferred to Section 3.8, which mostly follows a discretization argument similar to Chapter 2. The key machinery is still the Discrete Itô formula (Lemma 2.5), but with a different characterization of the discretization error that replaces Lemma 2.6.

Concretely, if we define the discretization error

$$\Delta_t := \bar{\nabla}_t V_\alpha(t, S_{t-1}) + \frac{1}{2} \bar{\nabla}_{SS} V_\alpha(t, S_{t-1}) + G^{-1} \lambda \left[\bar{\nabla}_S V_\alpha(t, S_{t-1} + 1) - \bar{\nabla}_S V_\alpha(t, S_{t-1} - 1) \right], \quad (3.5)$$

then there is the cumulative loss bound

$$\text{Regret}_T^\lambda(0) \le \sum_{t=1}^{T} \left(g_t x_t + \lambda \, |x_t - x_{t+1}| \right) \le -G \cdot V_\alpha(T, S_T) + G \sum_{t=1}^{T} \Delta_t.$$

Controlling the discretization error relies on the key lemma.

Lemma 3.2. *If $\alpha \ge 4\lambda G^{-1} + 2$, then for all t and against any adversary, $\Delta_t \le 0$.*

Finally, the augmented regret bound follows from the standard loss-regret duality (Lemma 2.3).

3.4.3 Continuous-time derivation

Now we show the derivation of our dual space scaling strategy from a continuous-time perspective. Technically, the procedure is analogous the previous chapter, but the

[3] Note that here we use C instead of ε to represent the confidence parameter in Theorem 3.1.

demonstrated insights are novel. Before starting, we need a generalized definition of the discrete derivative, with a tunable gap increment δ.

$$\bar{\nabla}_S^{\delta} V(t,S) := \frac{1}{2\delta}\left[V(t,S+\delta) - V(t,S-\delta)\right].$$

Note that the choice of $\delta = 1$ recovers $\bar{\nabla}_S V(t,S)$ in Algorithm 3.2. The Lipschitz constant G will be set to 1 for the ease of exposition.

Step 1: discrete-time recursive inequality First, let us consider the following inequality that characterizes "admissible" potentials for Algorithm 3.2. For all t and S,

$$V(t-1,S) \geq \max_{g\in[-1,1]} \left\{V(t,S-g) + g\bar{\nabla}_S^1 V(t,S) + \lambda\left|\bar{\nabla}_S^1 V(t,S) - \bar{\nabla}_S^1 V(t+1,S-g)\right|\right\}. \quad (3.6)$$

Finding solutions of this inequality is sufficient for constructing regret bounds. To see this, suppose the above holds for some V. We can then plug in $S = S_{t-1}$ and guarantee that for all $g_t \in [-1,1]$,

$$g_t x_t + \lambda\left|x_t - x_{t+1}\right| \leq V(t-1,S_{t-1}) - V(t,S_t).$$

A telescopic sum further leads to a cumulative loss bound $\mathrm{Regret}_T^{\lambda}(0) \leq V(0,0) - V(T,S_T)$, and a regret bound on $\mathrm{Regret}_T^{\lambda}(u)$ then follows from the standard loss-regret duality (Lemma 2.3).

Step 2: ε-scaled recursion Since we ideally need optimal potential functions that satisfy the inequality (3.6) without any slack, let us turn (3.6) into an equality and try to approximately solve it. Intuitively this is a challenging task, as there is no natural way to parameterize the dependence of V on the discrete time t. However, if

we decrease the discrete time interval, solutions V will be "smoother" and easier to describe. Concretely, let $\varepsilon > 0$ be a parameter that will later approach 0. On (3.6), we scale

1. the unit time by ε^2;

2. the loss gradient g, the switching cost weight λ and the gap increment by ε.

Both scaling factors are justified in the switching-free setting. Notably, since g and λ have the same "unit", it is natural that they are scaled by the same rate. With that, we obtain a scaled recursion

$$V(t-\varepsilon^2, S) = \max_{g\in[-1,1]} \left\{ V(t, S-\varepsilon g) + \varepsilon g \bar{\nabla}^{\varepsilon}_{S} V(t,S) + \varepsilon\lambda \left| \bar{\nabla}^{\varepsilon}_{S} V(t,S) - \bar{\nabla}^{\varepsilon}_{S} V(t+\varepsilon^2, S-\varepsilon g) \right| \right\}. \tag{3.7}$$

Step 3: continuous-time PDE To proceed, we take the second-order Taylor approximation on all components of (3.7). Both the zeroth and the first order terms of ε naturally vanish. Only keeping the second order terms, we have

$$\nabla_t V(t,S) + \max_{g\in[-1,1]} \left(\frac{1}{2} g^2 \nabla_{SS} V(t,S) + \lambda \left| g \nabla_{SS} V(t,S) \right| \right) = 0.$$

As typical potential functions are convex in the sufficient statistic S, it is reasonable to impose an additional condition $\nabla_{SS} V(t,S) \geq 0$. Then, the above becomes the 1D backward heat equation (BHE)

$$\nabla_t V + \alpha \nabla_{SS} V = 0,$$

where $\alpha = \lambda + 1/2$. Compared to the switching-free setting, we obtain the same PDE, but change the *negative thermal diffusivity* α from $1/2$ to $1/2 + \lambda$. This concisely

characterizes the effect of switching costs on the structure of the online learning problem.

Step 4: solving the PDE The final step is to solve the BHE. With a hyperparameter c, consider solutions of the form

$$V_\alpha(t, S) = t^c g\left(\frac{S}{\sqrt{4\alpha t}}\right).$$

Plug it in, the BHE reduces to the Hermite ODE

$$g''(z) - 2zg'(z) + 4cg(z) = 0,$$

which is *independent of* α. This is a crucial observation, as it reveals the correct way to generalize the knowledge from the switching-free setting to the setting with switching costs. More specifically,

- In the switching-free setting, we can take a solution $g(z)$ of the Hermite ODE, plug in the argument $z = S/\sqrt{2t}$ and obtain a potential function V_α.
- When switching costs are considered, the above derivation suggests us to take the *same* function $g(z)$ as before, and plug in a scaled argument $z = S/\sqrt{4\alpha t}$. This is precisely dual space scaling.

Finally, as shown in Chapter 2, a particularly good choice of c is $1/2$. Using this choice yields the switching-adjusted potential (3.4).

Remark 3.1. *To summarize, through this derivation we aim to demonstrate a key benefit of the continuous-time analysis: it makes the generalization of algorithmic structures easier. This was not presented in Chapter 2, but could be useful in the broader online learning context.*

Meanwhile, we do not intend to overclaim its strength – although the continuous-time analysis provides useful intuition, we ultimately care about discrete-time regret bounds. Discretizing such arguments relies on an obscure argument that has not

been made concrete yet: "V_α derived in the continuous time also serves as a good potential in the discrete time." Indeed, verifying this property is technically nontrivial (Section 3.4.2), and doing so requires a slightly more conservative choice of α (i.e., $4\lambda + 2$) than what is suggested above.

3.4.4 Extension

All the results so far (of our second algorithm) are established in the 1D unconstrained setting. We now show how to extend them to the constrained setting and the high dimensional setting. The latter is fairly standard. The former is more involved, which also leads to a similar cumulative switching cost bound as Part 2 of Theorem 3.1, but with a much simplified analysis. This property is critical for the tracking control application considered in (Zhang et al., 2022a).

Constrained domain First, consider a constrained domain $\mathcal{X} \subset \mathbb{R}$. We can use the standard black-box reduction (Algorithm 2.2) on top of our 1D unconstrained algorithm (Algorithm 3.2), such that the *exact* bound in Theorem 3.2 carries over (w.r.t. any constrained comparator $u \in \mathcal{X}$). Concretely, the pseudo-code is shown as Algorithm 3.3, where Π denotes the absolute value projection in 1D.

Algorithm 3.3 1D constrained OLO with switching costs.

Require: A hyperparameter $C > 0$, a closed and convex domain $\mathcal{X} \subset \mathbb{R}$, and an unconstrained algorithm $\mathcal{A}$ (Algorithm 3.2 induced by $V_{4\lambda G^{-1}+2}$ and the hyperparameter C). Let x^* be an arbitrary point in $\mathcal{X}$.

1: **for** $t = 1, 2, \ldots$ **do**
2: Query $\mathcal{A}$ for its prediction $\tilde{x}_t$.
3: Predict $x_t = \Pi_{\mathcal{X}}(\tilde{x}_t + x^*)$ and receive a loss gradient g_t.
4: Define a surrogate loss gradient $\tilde{g}_t$ as

$$\tilde{g}_t = \begin{cases} g_t, & \text{if } g_t(\tilde{x}_t + x^*) \geq g_t x_t, \\ 0, & \text{otherwise,} \end{cases}$$

and send $\tilde{g}_t$ to $\mathcal{A}$.
5: **end for**

Similar strategies apply to higher-dimensional problems, but here we emphasize the 1D special case due to an additional property: if the domain $\mathcal{X}$ has a finite diameter D, then the switching cost alone of the combined algorithm has a $\tilde{O}(D\sqrt{\tau})$ bound on any time interval of length τ, similar to Part 2 of Theorem 3.1. This could be useful when switching costs have high priority (Sherman and Koren, 2021; Wang et al., 2021) and should be independently bounded. Moreover, it allows the combination of comparator adaptive algorithms (Zhang et al., 2022a) in settings with long term prediction effects.

Theorem 3.3. *Let x^* be an arbitrary point in $\mathcal{X}$. For all $C > 0$, Algorithm 3.3 guarantees*

$$\text{Regret}_T^{\lambda}(u) \leq \sqrt{(4\lambda G + 2G^2)T}\left[C + |u - x^*|\left(\sqrt{4\log\left(1 + \frac{|u-x^*|}{C}\right)} + 2\right)\right],$$

for all $u \in \mathcal{X}$ and $T \in \mathbb{N}_+$. Moreover, if $\mathcal{X}$ has a finite diameter D, then on any time interval $[T_1 : T_2] \subset \mathbb{N}_+$, the same algorithm guarantees

$$\sum_{t=T_1}^{T_2-1} |x_t - x_{t+1}| \leq 22\sqrt{T_2 - T_1}\left[2D + C + 2D\sqrt{\log(1 + DC^{-1})}\right].$$

General dimensional domain In the general dimensional domain, the switching cost is modeled by the L_p norm, as introduced at the beginning of this chapter. Furthermore, the loss gradients, which are in $\mathbb{R}^d$, have the dual norm of L_p bounded by G.

When $p = 1$, we have $\mathcal{X} = \mathbb{R}^d$, $\|g_t\|_\infty \leq G$, and the switching costs are measured by the L_1 norm. We run Algorithm 3.2 on each coordinate separately (Streeter and Mcmahan, 2012), and scale the hyperparameter C by $1/d$. The pseudo-code is presented as Algorithm 3.4. The proof is a simple summation over all coordinates, therefore omitted.

Algorithm 3.4 d-dimensional OLO with L_1 norm switching costs.

Require: A hyperparameter $C > 0$ and Algorithm 3.2.

1: For each dimension $i \in [1 : d]$, initialize a copy of Algorithm 3.2 as $\mathcal{A}_i$. It uses the hyperparameter C/d and our potential V_α, with $\alpha = 4\lambda G^{-1} + 2$.
2: **for** $t = 1, 2, \ldots$ **do**
3: For all i, query $\mathcal{A}_i$ and assign its prediction to $x_{t,i}$. Define a vector as $x_t = [x_{t,1}, \ldots, x_{t,d}] \in \mathbb{R}^d$.
4: Predict x_t and receive a loss gradient $g_t = [g_{t,1}, \ldots, g_{t,d}]$.
5: For all i, send $g_{t,i}$ to $\mathcal{A}_i$ as a new surrogate loss gradient.
6: **end for**

Theorem 3.4. *For all $C > 0$, Algorithm 3.4 guarantees ($\alpha = 4\lambda G^{-1} + 2$)*

$$\sum_{t=1}^{T} \langle g_t, x_t - u \rangle + \lambda \sum_{t=1}^{T-1} \|x_t - x_{t+1}\|_1 \leq G\sqrt{\alpha T} \left[C + \|u\|_1 \left(\sqrt{4 \log \left(1 + \frac{\|u\|_\infty d}{C} \right)} + 2 \right) \right],$$

for all $u \in \mathbb{R}^d$ and $T \in \mathbb{N}_+$.

As for L_2 norm switching costs, we can follow the polar decomposition technique (Algorithm 2.1), which uses our 1D unconstrained OLO algorithm as the base algorithm. The only required modification is that the base algorithm should account for an extra regularization term, as in Algorithm 3.1 and Theorem 3.1. Concretely, instead of bounding the augmented regret (3.1), we should bound

$$\sum_{t=1}^{T} g_t(x_t - u) + \lambda \sum_{t=1}^{T-1} |x_t - x_{t+1}| + \frac{\gamma}{\sqrt{t}} \sum_{t=1}^{T} |x_t| , \tag{3.8}$$

for any given weight γ.

To this end, we can consider a surrogate OCO problem with switching costs, where the loss functions are defined by $l_t(x) = g_t x + \gamma t^{-1/2} |x|$. Such a loss function is Lipschitz, therefore we can use the OCO-OLO reduction, and run our Algorithm 3.2 on its linearized surrogate. Details of this step are standard, therefore omitted.

Given a 1D algorithm with controlled (3.8), we can apply the polar decomposition trick surveyed in Section 1.4 on top of it. For clarity, such a procedure is restated as Algorithm 3.5 below. The resulting algorithm has its augmented regret bounded by

Algorithm 3.5 Extension to $\mathbb{R}^d$.

Require: A 1D algorithm with a bound on (3.8), denoted by $\mathcal{A}_r$.

1: Define $\mathcal{A}_B$ as OGD on B^d with learning rate $\eta_t = 1/(G\sqrt{t})$, initialized at the origin 0.
2: **for** $t = 1, 2, \ldots$ **do**
3: Obtain $y_t \in \mathbb{R}$ from $\mathcal{A}_r$ and $z_t \in \mathbb{R}^d$ from $\mathcal{A}_B$.
4: Predict $x_t = y_t z_t \in \mathbb{R}^d$, observe $g_t \in \mathbb{R}^d$.
5: Return $\langle g_t, z_t \rangle$ and g_t as the t-th loss gradient to $\mathcal{A}_r$ and $\mathcal{A}_B$, respectively.
6: **end for**

$$
\begin{aligned}
& \sum_{t=1}^{T} \langle g_t, x_t - u \rangle + \lambda \sum_{t=1}^{T-1} \|x_t - x_{t+1}\|_2 \\
= & \sum_{t=1}^{T} \langle g_t, y_t z_t - u \rangle + \lambda \sum_{t=1}^{T-1} \|y_t z_t - y_{t+1} z_{t+1}\|_2 \\
\leq & \sum_{t=1}^{T} \left(\langle g_t, z_t \rangle y_t - \langle g_t, z_t \rangle \|u\|_2 \right) + \|u\|_2 \sum_{t=1}^{T} \left\langle g_t, z_t - \frac{u}{\|u\|_2} \right\rangle \\
& \qquad + \lambda \sum_{t=1}^{T-1} |y_t - y_{t+1}| \, \|z_{t+1}\|_2 + \lambda \sum_{t=1}^{T-1} \|z_t - z_{t+1}\|_2 \, |y_t| \\
\leq & \sum_{t=1}^{T} \left(\langle g_t, z_t \rangle y_t - \langle g_t, z_t \rangle \|u\|_2 \right) + \lambda \sum_{t=1}^{T-1} |y_t - y_{t+1}| + \sum_{t=1}^{T-1} \frac{\lambda}{\sqrt{t}} y_t \\
& \qquad + \|u\|_2 \sum_{t=1}^{T} \left\langle g_t, z_t - \frac{u}{\|u\|_2} \right\rangle .
\end{aligned}
$$

On the RHS of the last inequality, the first three terms can be bounded via (3.8), and the last term can be controlled by the standard result of OGD.

3.5 Learning with expert advice

In this section, we extend our results so far on OCO to a related problem called *Learning from Expert Advice* (LEA) (Littlestone and Warmuth, 1994), still with switching costs.

LEA is another well-studied problem in online learning, with a history even longer than OCO. The original setting is that we are given d experts, i.e., decision making algorithms. In each round we choose to follow one of them (indexed by i_t) in the face of an adversarial environment, and then observe the loss (a real number) of all the experts, denoted by $g_{t,1}, \dots, g_{t,d}$. The goal is to asymptotically perform no worse than the best of the experts. In other words, we aim to show that the regret

$$\sum_{t=1}^{T} g_{t,i_t} - \sum_{t=1}^{T} g_{t,j}$$

is sublinear in T for all comparator expert index j.

Since the proposal of OCO/OLO, LEA is often formulated as their special case on the probability simplex. Concretely, we adopt the following definition.

Definition 8 (Learning with Expert Advice)**.** *LEA is the special case of OLO where*

- *the domain $\mathcal{X}$ is the probability simplex $\Delta(d)$; and*
- *the linear loss functions are G-Lipschitz with respect to the L_∞ norm. That is, the loss gradient g_t is in the L_∞ norm ball of radius G, centered at the origin.*

The relation of this OLO formulation to the original expert setting is that, any prediction given by the OLO algorithm is a probability vector in $\Delta(d)$, from which we can sample the index i_t, or equivalently, an individual expert to follow. In this way, the instantaneous loss $\langle g_t, x_t \rangle$ in OLO is equivalent to the *expected loss* of the expert we choose, and furthermore, the regret of OLO (with comparators being the unit directional vectors) is equivalent to the expected regret of our expert selection

procedure. The reader is referred to (Orabona, 2019, Section 6.7) for a detailed treatment.

For LEA, there is a classical minimax algorithm called *Exponentiated Gradient* (EG) (Littlestone and Warmuth, 1994). It is another instance of the *Online Mirror Descent* (OMD) framework which generalizes OGD, therefore, EG shares essentially a similar intuition as OGD. With an unknown T, EG guarantees

$$\sup_{Env;u\in\Delta(d)} \text{Regret}_T(Env, u) = O\left(G\sqrt{T\log d}\right).$$

The limitation is that, the parameter d can be large in certain situations (for example, exponential in some auxiliary variable), therefore the $\log d$ term in the bound is still not ideal. Through a more theoretical lens, the $\log d$ term measures the capacity of the comparator class just like the D factor in the standard $O(DG\sqrt{T})$ regret bound of OGD. Therefore, we may improve it using the philosophy of comparator adaptivity.

Comparator adaptivity in LEA Although LEA is a special case of OLO, the geometry of the domain motivates a different form of bounds from Section 1.4. The typical comparator adaptive regret bound in LEA has the form $O\left(G\sqrt{T\cdot \text{KL}(u||\pi)}\right)$, where $u, \pi \in \Delta(d)$ represent the comparator and a user-chosen prior. Such an idea was initiated in (Chaudhuri et al., 2009), and the analysis was improved and extended by a series of works (Chernov and Vovk, 2010; Luo and Schapire, 2015; Koolen and Van Erven, 2015; Chen et al., 2021; Negrea et al., 2021; Portella et al., 2022). Notably, a comparator adaptive LEA algorithm naturally induces a bound on the *ε-quantile regret* – the regret with respect to the ε-quantile best expert. Lower bounds were considered in (Negrea et al., 2021).

In this section, we aim to improve the state-of-the-art comparator adaptive LEA bounds, in a strictly generalized setting with switching costs. Just like how switching costs were introduced in OCO at the beginning of this chapter, we define the setting

as follows.

Definition 9 (LEA with switching costs)**.** *LEA with switching costs is the following variant of LEA: after the t-th round, besides suffering the instantaneous loss $\langle g_t, x_t \rangle$, the learning agent also suffers a switching cost $\lambda \|x_t - x_{t-1}\|_1$, where the weight λ is known to the agent. Without loss of generality, $x_0 = 0$. With the augmented regret defined as*

$$\mathrm{Regret}_T^{\lambda}(Env, u) := \sum_{t=1}^{T} \langle g_t, x_t - u \rangle + \lambda \sum_{t=1}^{T-1} \|x_{t+1} - x_t\|_1 ,$$

our goal is to upper bound

$$\mathrm{Regret}_T^{\lambda}(u) := \sup_{Env} \mathrm{Regret}_T^{\lambda}(Env, u),$$

for all $T \in \mathbb{N}_+$ and $u \in \Delta(d)$.

When $G = 1$, minimax algorithms such as the *Shrinking Dartboard* (Geulen et al., 2010) guarantee $\mathrm{Regret}_T^{\lambda}(u) = O(\sqrt{\lambda T \log d})$, which is similar to EG, but with the additional switching costs. Before our results, comparator adaptivity had not been established in LEA with switching costs.

3.5.1 Reduction of LEA to OLO

Our main contribution is a new reduction technique that converts comparator adaptive LEA to comparator adaptive OLO (on $\mathbb{R}^d$). Such a reduction problem has been considered in prior works on vanilla LEA (Luo and Schapire, 2015; Orabona and Pál, 2016), but the existing reduction technique is unsatisfactory for controlling the switching cost. Let us explain.

The existing technique has the following procedure. Given a 1D OLO algorithm that predicts on $\mathbb{R}_+$, independent copies are created for each coordinate (of $\mathbb{R}^d$) and updated using certain surrogate losses. A meta-algorithm queries the coordinate-wise predictions $\{w_{t,i}; i \in [1:d]\}$, collects them into a weight vector $w_t = [w_{t,1}, \ldots, w_{t,d}]$,

and finally predicts the scaled weight $x_t = w_t / \|w_t\|_1$ on $\Delta(d)$. Despite its success in the vanilla LEA problem, such an approach has a discontinuity problem when switching costs are incorporated – if two consecutive weights w_t and w_{t+1} are both close to the origin, then simply scaling them to $\Delta(d)$ can lead to a large switching cost, even when $\|w_t - w_{t+1}\|_1$ is small. This problem is exacerbated by the typical setting[4] of $w_1 = 0$, due to the associated analysis. A graphical demonstration is provided in Figure 3·1 (Left).

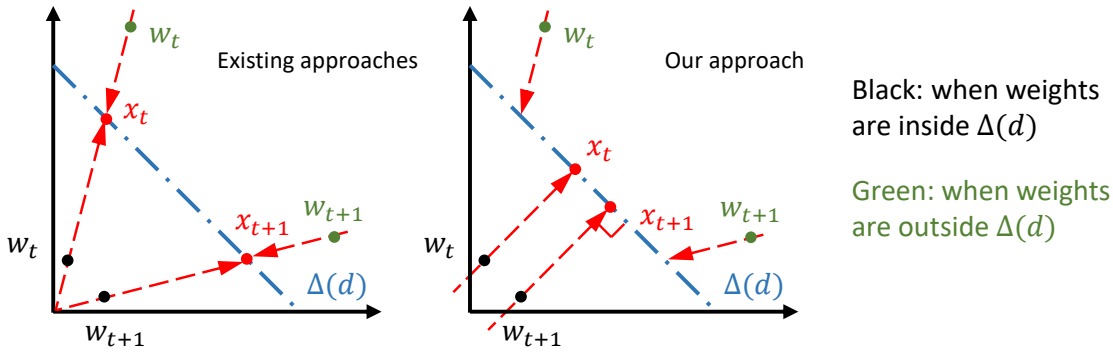


Figure 3·1: Switching costs in LEA-OLO reductions. Left: existing approaches. Right: ours, where the projection of w_t contains two cases. (i) $\|w_t\|_1 \geq 1$, shown in green; (ii) $\|w_t\|_1 < 1$, shown in black.

In contrast, our solution is based on *a unified view* of the LEA-OLO reduction and the constrained domain reduction surveyed as Algorithm 2.2 in Section 2.3. Starting without switching costs, we observe that the general Banach version of the latter (Cutkosky and Orabona, 2018) can be adopted to turn the unconstrained prediction $w_t \in \mathbb{R}^d_+$ to the constrained one $x_t \in \Delta(d)$, which generalizes the existing scaling procedure. Therefore, specialized techniques like (Luo and Schapire, 2015; Orabona and Pál, 2016) are not required for this task. Algorithmically, we set $x_t \in \arg\min_{x \in \Delta(d)} \|x - w_t\|_1$ as opposed to $x_t = w_t / \|w_t\|_1$. The major benefit is the non-uniqueness of the L_1 norm projection – if $\|w_t\| < 1$, then any $x_t \in \Delta(d)$ satisfying

[4]When $w_t = 0$, x_t can be arbitrary on $\Delta(d)$ by definition. However, as w_t changes continuously w.r.t. the observed information, it could hover around 0 at some point, thus experiencing the sketched problem.

$\{x_{t,i} \geq w_{t,i}; \forall i\}$ minimizes $\|x - w_t\|_1$ on $\Delta(d)$. This brings more flexibility to the algorithm design. Specifically, we adopt

1. the orthogonal projection $x_t = w_t + d^{-1}(1 - \|w_t\|_1)$ when $\|w_t\|_1 \leq 1$;
2. the scaling $x_t = w_t / \|w_t\|_1$ when $\|w_t\|_1 > 1$.

With the addition of switching costs, the orthogonal projection is better for controlling it, as shown in Figure 3·1 (Right). Concretely, we present the pseudocode as Algorithm 3.6, which is the first comparator adaptive algorithm for LEA with switching costs. It uses our second algorithm for OLO with switching costs as the base algorithm, whose predictions are aggregated and constrained to $\Delta(d)$ via the aforementioned L_1 norm projection. The performance guarantee is the following theorem.

Theorem 3.5. *For LEA with switching costs, given any prior π in the relative interior of $\Delta(d)$, Algorithm 3.6 guarantees*

$$\mathrm{Regret}_T^{\lambda}(u) = \left[\sqrt{\mathrm{TV}(u||\pi) \cdot \mathrm{KL}(u||\pi)} + 1\right] \cdot O\left(\sqrt{(\lambda G + G^2)T}\right),$$

for all $u \in \Delta(d)$ and $T \in \mathbb{N}_+$.

We emphasize two strengths of this bound.

1. Since it is comparator adaptive, such a bound only implicitly depends on d through the divergence term $\sqrt{\mathrm{TV} \cdot \mathrm{KL}}$. In favorable cases we may have a good prior π such that $\mathrm{TV}(u||\pi) \cdot \mathrm{KL}(u||\pi) = O(1)$; this will save us a $\sqrt{\log d}$ factor compared to minimax algorithms (with switching costs), such as *Follow the Lazy Leader* (Kalai and Vempala, 2005) and *Shrinking Dartboard* (Geulen et al., 2010).
2. Even without switching costs, we improve the $\sqrt{\mathrm{KL}}$ divergence term in existing comparator adaptive bounds (Chaudhuri et al., 2009; Luo and Schapire, 2015;

Algorithm 3.6 Converting OLO to LEA via the constrained domain reduction.

Require: A prior $\pi = [\pi_1, \ldots, \pi_d]$ in the relative interior of $\Delta(d)$, and Algorithm 3.3 (our improve algorithm for 1D OLO with switching costs, with an added constraint and a coordinate shift).

1: For each dimension $i \in [1:d]$, initialize a copy of Algorithm 3.3 as $\mathcal{A}_i$. We set $\tilde{\lambda} = 4\lambda$ and $\tilde{G} = 2G$ as the switching cost weight and the Lipschitz constant adopted internally by A_i. Moreover, A_i uses the domain $\mathcal{X} = \mathbb{R}_+$, the offset $x^* = \pi_i$, the hyperparameter π_i, and our potential V_α, where $\alpha = 4\tilde{\lambda}\tilde{G}^{-1} + 2$.

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

3: For all i, query $\mathcal{A}_i$ and assign its prediction to $w_{t,i}$. Define the weight vector as $w_t = [w_{t,1}, \ldots, w_{t,d}] \in \mathbb{R}_+^d$.

4: Compute the LEA prediction $x_t = [x_{t,1}, \ldots, x_{t,d}]$ from

$$x_{t,i} = \frac{w_{t,i} + \frac{1}{d}\max\{0, 1 - \|w_t\|_1\}}{\max\{\|w_t\|_1, 1\}}.$$

5: Predict x_t and receive a loss vector $g_t \in [-G, G]^d$.

6: For all i, compute

$$z_{t,i} = \begin{cases} g_{t,i} - \|g_t\|_\infty, & \text{if } \|w_t\|_1 < 1, \\ g_{t,i}, & \text{if } \|w_t\|_1 = 1, \\ g_{t,i} + \|g_t\|_\infty, & \text{if } \|w_t\|_1 > 1, \end{cases}$$

and return $z_{t,i}$ to $\mathcal{A}_i$ as a new surrogate loss.

7: **end for**

Orabona and Pál, 2016) to $\sqrt{\mathrm{TV} \cdot \mathrm{KL}}$. The latter is better since (i) TV is always less than 1, and (ii) there exist $p, q \in \Delta(d)$ such that $\mathrm{TV}(p||q) \cdot \mathrm{KL}(p||q) \le 1$ but $\mathrm{KL}(p||q) \ge \sqrt{\log d} - o(1)$ (see Example 3.1). In other words, compared to $\sqrt{\mathrm{KL}}$, the $\sqrt{\mathrm{TV} \cdot \mathrm{KL}}$ bound is never worse (up to constants), and can save at least a $(\log d)^{1/4}$ factor in certain cases. Generalizations of root KL to *f-divergences* have been considered in (Alquier, 2021; Negrea et al., 2021), but to our knowledge, no prior algorithm guarantees a better divergence term than root KL.

In summary, our result for the first time achieves comparator adaptivity in LEA with switching costs. Even in the absence of switching costs, we improve the state-of-the-art

comparator adaptive LEA regret bounds through a better divergence characterization than KL.

3.6 Application and experiment

To complement our theoretical results, we present applications to a portfolio selection problem with transaction costs. Online portfolio selection has been studied by multiple communities, resulting in a large amount of literature. Here we focus on an *unconstrained* setting, allowing both short selling (i.e., holding negative amount of assets) and margin trading (i.e., borrowing money to buy assets).

Setting We consider a market with d assets and discrete trading period $t \in \mathbb{N}_+$. In the t-th round, an algorithm chooses a portfolio vector $x_t = [x_{t,1}, \dots, x_{t,d}] \in \mathbb{R}^d$, where $x_{t,i}$ is the *number of shares* of the i-th asset that the algorithm suggests to hold. Compared to the previous round, we need to buy $x_{t,i} - x_{t-1,i}$ shares[5] (or sell, if negative), which requires paying a $\lambda |x_{t,i} - x_{t-1,i}|$ transaction cost. Then, the market reveals a number $g_{t,i} \in [-G, G]$, which represents the price change per share (of the i-th asset) in this round. This effectively increases the value of our portfolio by $\langle g_t, x_t \rangle$.

The considered performance metric is the increased amount of *wealth* on any time horizon $[1 : T] \subset \mathbb{N}_+$, and such wealth includes the total value of our portfolio *plus cash*. Our goal is to show that the performance of our algorithm is never much worse than that of any unconstrained *Buy-and-Hold* (BAH) strategy, which picks a portfolio vector $u \in \mathbb{R}^d$ at the beginning and holds that amount throughout the considered time horizon. That is, we aim to upper bound $\sum_{t=1}^{T} \langle -g_t, x_t - u \rangle + \lambda \sum_{t=1}^{T-1} \|x_t - x_{t+1}\|_1$ for all $u \in \mathbb{R}^d$ and $T \in \mathbb{N}_+$. This is exactly the setting of Algorithm 3.4 with flipped gradients, therefore the same theoretical result (Theorem 3.4) carries over.

[5]Without loss of generality, assume $x_0 = x_1$.

Synthetic market First, let us consider an experiment with synthetic data.

We test both 1D algorithms (Algorithm 3.1 with the constraint $[0, \bar{R}]$ removed and the betting fraction β_t allowed to be negative; and Algorithm 3.2) proposed in this chapter, extended to higher dimensions using the coordinate-wise construction (Algorithm 3.4). Both algorithms require a confidence parameter (ε in Algorithm 3.1; C in Algorithm 3.2), and we set them to 1 following the practice of comparator adaptive algorithms (Orabona and Pál, 2016; Chen et al., 2022; Zhang et al., 2022a).

Regarding the synthetic market, we let $G = 1$, $\lambda = 0.1$, and the market contains five assets with different return characteristics. Each $g_{t,i}$ is the summation of a i.i.d. noise, a periodic fluctuation and a constant trend. Specifically, we consider three different market return models. The first is

$$g_{t,1} = 0.4 \cdot \text{Uniform}[-1, 1] + 0.4 \sin[(t/500) \cdot \pi] + 0.2,$$

$$g_{t,2} = 0.5 \cdot \text{Uniform}[-1, 1] + 0.3 \sin[(t/500 + 1/2) \cdot \pi] + 0.2,$$

$$g_{t,3} = 0.6 \cdot \text{Uniform}[-1, 1] + 0.2 \sin[(t/500 + 1) \cdot \pi] + 0.2,$$

$$g_{t,4} = 0.7 \cdot \text{Uniform}[-1, 1] + 0.1 \sin[(t/500 + 3/2) \cdot \pi] + 0.2,$$

$$g_{t,5} = 0.8 \cdot \text{Uniform}[-1, 1] + 0.2.$$

The second model is

$$g_{t,1} = 0.2 \cdot \text{Uniform}[-1, 1] + 0.4 \sin[(t/500) \cdot \pi] + 0.4,$$

$$g_{t,2} = 0.3 \cdot \text{Uniform}[-1, 1] + 0.3 \sin[(t/500 + 1/2) \cdot \pi] + 0.4,$$

$$g_{t,3} = 0.4 \cdot \text{Uniform}[-1, 1] + 0.2 \sin[(t/500 + 1) \cdot \pi] + 0.4,$$

$$g_{t,4} = 0.5 \cdot \text{Uniform}[-1, 1] + 0.1 \sin[(t/500 + 3/2) \cdot \pi] + 0.4,$$

$$g_{t,5} = 0.55 \cdot \text{Uniform}[-1, 1] + 0.45.$$

The third model is the same as the second one, except we replace $g_{t,5}$ by

$$g_{t,5} = 0.5 \cdot \text{Uniform}[-1, 1] + 0.5.$$

For each market return model, we test both algorithms in 50 random trials, and the increased wealth $\sum_{\tau=1}^{t} \langle g_\tau, x_\tau \rangle - \lambda \sum_{\tau=1}^{t-1} \|x_\tau - x_{\tau+1}\|_1$ (mean $\pm$ std) is plotted in Figure 3·2, higher is better. In all three settings, our second algorithm (denoted as "ours") beats the first one (denoted as "baseline") by a considerable margin, due to being a lot less conservative.

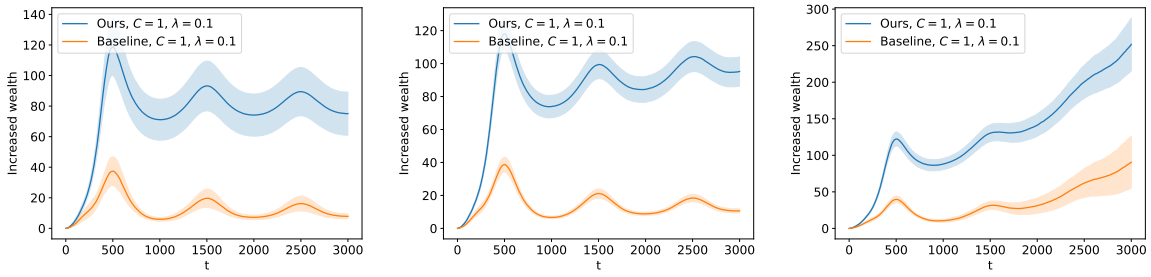


Figure 3·2: Synthetic market experiment with different market models. From left to right: the first, the second and the third market model.

Historical stock data Next, we present results on the historical US stock data. Eight stocks (Table 3.1) are considered on a time period of 5 years (1/1/2013 to 1/1/2018).

The algorithm trades once per day after the market closes, based on the closing price. We take the difference between the closing price on the $(t+1)$-th day and the closing price on the t-th day, and define it as the market vector g_t. The largest single day price change for any stock is less than \$15, therefore G is set in a posterior manner to 15. We consider a hypothetical broker that charges \$0.1 per share, therefore define $\lambda = 0.1$.

Same as the synthetic market experiment, we test both algorithms from this chapter.

Table 3.1: List of considered stocks

Company	Symbol
Apple Inc.	AAPL
Berkshire Hathaway Inc. Class B	BRK.B
Meta Platforms Inc.	FB
Johnson & Johnson	JNJ
JPMorgan Chase & Co.	JPM
Microsoft Corporation	MSFT
Pfizer Inc.	PFE
Exxon Mobil Corporation	XOM

The second algorithm is in its default parameter-free implementation ($C = 1$). However, setting the same confidence parameter is too conservative for the first algorithm (i.e., the "baseline"), which means the baseline hardly makes any investment, making the comparison less interesting. Therefore we set the confidence parameter as 10 for the baseline (denoted by ε in Algorithm 3.1), thus giving it an advantage at the beginning. In this way, the increased wealth of the two algorithms is roughly comparable.

We plot the results in Figure 3·3. Specifically, Figure 3·3 (Left) shows the increased wealth (in USD) over the considered time period. Figure 3·3 (Right) shows the cumulative amount of investment (in USD), which is the total net amount of cash the investor uses to buy stocks (i.e., increases when buying, and decreases when selling), plus the transaction costs paid to the broker.

From the plot we can see that the baseline is more aggressive at the beginning, due to a much larger C. Therefore, it slightly makes more profit during 2013-2014. When the market oscillates and declines in 2015 and 2016, the two algorithms perform roughly the same, while the baseline has a lower risk due to holding a smaller portfolio at the time. However, the major difference starts after mid-2016, when the market

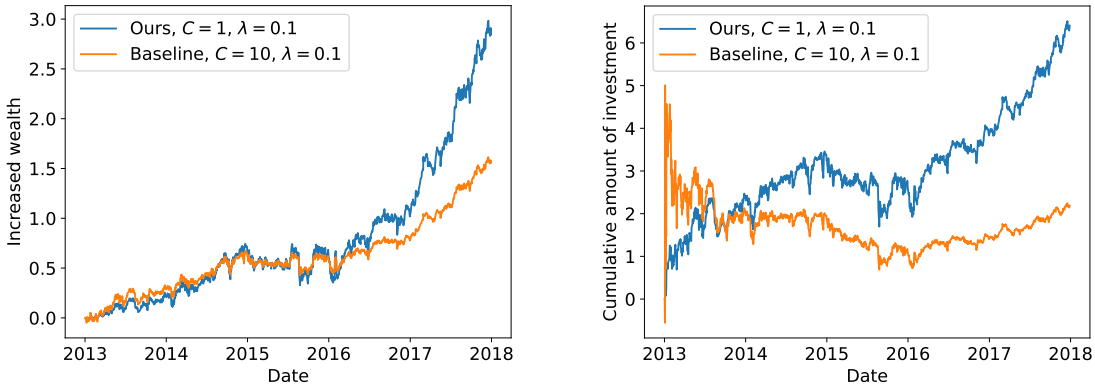


Figure 3·3: Experiment on historical US stock data. Left: the increased wealth of the two algorithms. Right: total amount of investment since the start of the experiment (1/1/2013), including the transaction costs paid to the broker.

grows rapidly. Our second, improved algorithm is able to identify this trend and quickly increase the amount of investment. This brings a lot more profit than the baseline, which hardly recovers its confidence from the declining market in the previous year. Such an advantage is partly due to the better control of switching costs, and partly due to a better risk-return tradeoff.

Our experiment also shows a limitation of the unconstrained investment setting. Throughout this five year period, the second algorithm invests a total amount of ∼\$6.5 (including the transaction costs), and makes a total profit of ∼\$3. However, in practice, one typically invests a lot more than this (let's say, \$10,000), and expect a similar rate of return. Our setting does not model such a budget explicitly; instead, it relies on the comparator adaptivity of the trading algorithms to increase the invested amount. Such a process can be slow, especially since we only consider trading once per day. Therefore, to use our algorithms in real trading situations, one has to tune the confidence parameter C to implicitly take his budget and tolerable risk into account. For example, using $C = 1000$ would result in investing \$6,500 throughout the five year period, and make a total profit of \$3,000. The connection of this approach to

rebalancing could be an interesting direction for future works.

3.7 Summary

This chapter investigates the design of comparator adaptive algorithms in the presence of switching costs. By carefully trading off these two opposite considerations, we present two OCO algorithms, with the second one achieving several forms of optimality. Notably, the key idea of this algorithm is not guessed, but derived from a continuous-time analysis. Through this result, we aim to demonstrate a key strength of the continuous-time PDE analysis – it makes the generalization of algorithmic structures much easier. Such an observation could open up exciting possibilities. For example,

- Does this approach apply to other variants of the online learning problem?
- Can we use it to generalize other forms of adaptivity?
- Continuous-time potentials have been extensively studied under the framework of *potential theory* (Doob, 1984). Can we borrow techniques from there to further improve the workflow of algorithm design?

3.8 Proofs

3.8.1 The first algorithm

Lemma 3.1

Proof of Lemma 3.1. For clarity, Equation (3.3) is copied here.

$$\text{Wealth}_t = (1 - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t})\text{Wealth}_{t-1} - \lambda|\beta_t\text{Wealth}_{t-1} - \beta_{t+1}\text{Wealth}_t|.$$

By definition, $|\lambda\beta_{t+1}| \leq 1/2$. The RHS of (3.3) is 1/2-Lipschitz with respect to Wealth_t, and the LHS is Wealth_t itself. Therefore, a solution exists and is unique.

To prove $\text{Wealth}_t > 0$, we use induction. $\text{Wealth}_0 = \varepsilon > 0$. Suppose $\text{Wealth}_{t-1} > 0$, then

$$\text{Wealth}_t \geq (1 - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t})\text{Wealth}_{t-1} - \lambda\beta_t\text{Wealth}_{t-1} - \lambda\beta_{t+1}\left|\text{Wealth}_t\right|.$$

Let $z = \lambda\beta_{t+1}\text{sign}(\text{Wealth}_t)$. Note that $|z| \leq 1/2$ and $|\tilde{g}_t + \gamma/\sqrt{t} + \lambda|\beta_t \leq 1/2$. Therefore,

$$\text{Wealth}_t \geq \frac{1 - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t} - \lambda\beta_t}{1+z}\text{Wealth}_{t-1} > 0. \qquad \square$$

Theorem 3.1 The proof is a lengthy one, with several auxiliary lemmas introduces in the end.

Proof of Theorem 3.1. We prove the two parts of Theorem 3.1 separately, starting from the second part.

Combining Lemma 3.4 and Lemma 3.5, for all $t \geq 2$,

$$|\tilde{x}_t - \tilde{x}_{t+1}| \leq \frac{6}{Ct} \cdot 4\bar{R}C\sqrt{t-1} \leq 24\bar{R}\frac{1}{\sqrt{t}}.$$

For $t = 1$, the same result can be verified. Therefore, for all $[a:b] \subset [1:T]$,

$$\begin{aligned}\sum_{t=a}^{b}|x_t - x_{t+1}| \leq 24\bar{R}\sum_{t=a}^{b}\frac{1}{\sqrt{t}} &\leq 24\bar{R}\int_{a-1}^{b}\frac{1}{\sqrt{x}}dx \\ &\leq 24\bar{R}\left(2\sqrt{b} - 2\sqrt{a-1}\right) \leq 48\bar{R}\sqrt{b-a+1}.\end{aligned}$$

The fourth inequality is due to $\sqrt{b} - \sqrt{a-1} \leq \sqrt{b-a+1}$.

Now consider the proof of the first part of the theorem. Due to the complexity, we proceed in steps.

Step 1 The overall strategy

The considered bound does not rely on the bounded domain, therefore the first step is to apply the reduction from constrained OLO to unconstrained OLO (Lemma 2.2)

and the contraction property of Euclidean projection to show that

$$\sum_{t=1}^{T}\left(g_t x_t - g_t u + \lambda\left|x_t - x_{t+1}\right| + \frac{\gamma}{\sqrt{t}}\left|x_t\right|\right) \leq \sum_{t=1}^{T}\left(\tilde{g}_t\tilde{x}_t - \tilde{g}_t u + \lambda\left|\tilde{x}_t - \tilde{x}_{t+1}\right| + \frac{\gamma}{\sqrt{t}}\left|\tilde{x}_t\right|\right). \quad (3.9)$$

Note that Wealth_{t-1} is positive due to Lemma 3.1, and $\beta_t \geq 0$ from our construction. Therefore, $\tilde{x}_t \geq 0$. From here, we can focus on bounding the RHS of (3.9) with $|\tilde{x}_t|$ replaced by $\tilde{x}_t$. Also note that $|\tilde{g}_t| \leq |g_t| \leq G$ from Lemma 2.2.

From (3.3), we can rewrite wealth as

$$\text{Wealth}_T = \varepsilon - \sum_{t=1}^{T}\left(\tilde{g}_t\tilde{x}_t + \lambda\left|\tilde{x}_t - \tilde{x}_{t+1}\right| + \frac{\gamma}{\sqrt{t}}\tilde{x}_t\right).$$

If we guarantee $\text{Wealth}_T \geq F(-\sum_{t=1}^{T}\tilde{g}_t)$ for an arbitrary function F, then

$$\begin{aligned}\sum_{t=1}^{T}\left(\tilde{g}_t\tilde{x}_t - \tilde{g}_t u + \lambda\left|\tilde{x}_t - \tilde{x}_{t+1}\right| + \frac{\gamma}{\sqrt{t}}\tilde{x}_t\right) &= \varepsilon + \left\langle -\sum_{t=1}^{T}\tilde{g}_t, u\right\rangle - \text{Wealth}_T \\ &\leq \varepsilon + \left\langle -\sum_{t=1}^{T}\tilde{g}_t, u\right\rangle - F\left(-\sum_{t=1}^{T}\tilde{g}_t\right) \\ &\leq \varepsilon + \sup_{X\in\mathbb{R}}\left(\langle X, u\rangle - F\left(X\right)\right) = \varepsilon + F^*(u),\end{aligned}$$

where F^* is the Fenchel conjugate of F. Therefore, our goal is to find such an lower bound for Wealth_T, and then take its Fenchel conjugate.

Step 2 Recursion on the wealth update

Now consider (3.3). There are two cases: (i) $\beta_t\text{Wealth}_{t-1} \geq \beta_{t+1}\text{Wealth}_t$; (ii) $\beta_t\text{Wealth}_{t-1} < \beta_{t+1}\text{Wealth}_t$. If $\beta_t\text{Wealth}_{t-1} \geq \beta_{t+1}\text{Wealth}_t$, then

$$(1-\lambda\beta_{t+1})\text{Wealth}_t = (1 - \tilde{g}_t\beta_t - \lambda\beta_t - \gamma\beta_t/\sqrt{t})\text{Wealth}_{t-1},$$

$$\log\text{Wealth}_t = \log\text{Wealth}_{t-1} + \log[1 - \beta_t(\tilde{g}_t + \lambda + \gamma/\sqrt{t})] - \log(1-\lambda\beta_{t+1}).$$

Note that $\beta_t|\tilde{g}_t + \lambda + \gamma/\sqrt{t}| \leq 1/2$ and $\lambda\beta_{t+1} < 1$. Applying $\log(1-x) \geq -x - x^2$ for

all $x \leq 1/2$ and $\log(1+x) \leq x$ for all $x > 1$, we have

$$\begin{aligned}\log \text{Wealth}_t &\geq \log \text{Wealth}_{t-1} - \beta_t(\tilde{g}_t + \lambda + \gamma/\sqrt{t}) - \beta_t^2(\tilde{g}_t + \lambda + \gamma/\sqrt{t})^2 + \lambda\beta_{t+1} \\ &\geq \log \text{Wealth}_{t-1} - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t} - C^2\beta_t^2 + \lambda(\beta_{t+1} - \beta_t).\end{aligned}$$

Similarly, if $\beta_t \text{Wealth}_{t-1} < \beta_{t+1}\text{Wealth}_t$, then

$$\log \text{Wealth}_t \geq \log \text{Wealth}_{t-1} - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t} - C^2\beta_t^2 + \lambda(\beta_t - \beta_{t+1}).$$

Therefore, combining both cases, we have

$$\log \text{Wealth}_t \geq \log \text{Wealth}_{t-1} - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t} - C^2\beta_t^2 + \lambda|\beta_t - \beta_{t+1}|,$$

and summed over $[1:T]$,

$$\log \text{Wealth}_T \geq \log \varepsilon - \sum_{t=1}^{T} \tilde{g}_t\beta_t - C^2\sum_{t=1}^{T}\beta_t^2 - \gamma\sum_{t=1}^{T}\frac{\beta_t}{\sqrt{t}} - \lambda\sum_{t=1}^{T}|\beta_t - \beta_{t+1}|. \quad (3.10)$$

Step 3 Bounding the sums on the RHS of (3.10)

We start from the first two sums on the RHS of (3.10). β_t is the output of Follow the Leader (FTL) on the strongly convex losses $\psi_t(\beta) = \tilde{g}_t\beta + C^2\beta^2 + I\{0 \leq \beta \leq 1/(C\sqrt{2t})\}(\beta)$, where $I\{0 \leq \beta \leq 1/(C\sqrt{2t})\}(\beta)$ is a convex function of β that equals 0 when $0 \leq \beta \leq 1/(C\sqrt{2t})$ and infinity otherwise. Note that ψ_t is $2C^2$-strongly convex, therefore a standard result shows that the regret of this FTL problem is logarithmic in T. Concretely, from Corollary 7.17 of (Orabona, 2019),

$$\sum_{t=1}^{T}\left(\tilde{g}_t\beta_t + C^2\beta_t^2\right) - \min_{0\leq u\leq 1/(C\sqrt{2T})}\sum_{t=1}^{T}\left(\tilde{g}_t u + C^2u^2\right) \leq \frac{G^2}{4C^2}\left(1 + \log T\right).$$

Moreover, taking $u = 1/(C\sqrt{2T})$,

$$\min_{0\leq u\leq 1/(C\sqrt{2T})}\sum_{t=1}^{T}\left(\tilde{g}_t u + C^2u^2\right) \leq \frac{\sum_{t=1}^{T}\tilde{g}_t}{C\sqrt{2T}} + \frac{1}{2}.$$

As for the other sums in (3.10),

$$\sum_{t=1}^{T} \frac{\beta_t}{\sqrt{t}} = \frac{1}{\sqrt{2}C} \sum_{t=1}^{T} \frac{1}{t} \leq \frac{1}{\sqrt{2}C}(1 + \log T).$$

Applying Lemma 3.3,

$$\sum_{t=1}^{T} |\beta_t - \beta_{t+1}| \leq \frac{2}{C} \sum_{t=1}^{T} \frac{1}{t} \leq \frac{2}{C}(1 + \log T).$$

Plugging the above into (3.10),

$$\log \text{Wealth}_T \geq \log \varepsilon - \frac{\sum_{t=1}^{T} \tilde{g}_t}{C\sqrt{2T}} - 2(1 + \log T) - \frac{1}{2},$$

$$\text{Wealth}_T \geq \frac{\varepsilon}{\exp(5/2) \cdot T^2} \exp\left(- \frac{\sum_{t=1}^{T} \tilde{g}_t}{C\sqrt{2T}} \right).$$

Step 4 Taking Fenchel conjugate

From the Fechel conjugate table, if $f(x) = a \exp(bx)$ with $a, b > 0$, then for all $\theta \geq 0$,

$$f^*(\theta) = \frac{\theta}{b} \left(\log \frac{\theta}{ab} - 1 \right).$$

Applying this result on

$$F(x) = \frac{\varepsilon}{\exp(5/2) \cdot T^2} \exp\left(\frac{x}{C\sqrt{2T}} \right),$$

for all $u \geq 0$ we have

$$F^*(u) = uC\sqrt{2T} \left(\frac{3}{2} + \log \frac{\sqrt{2} u C T^{5/2}}{\varepsilon} \right).$$

Combining the above with Step 1 completes the proof. □

The above proof of Theorem 3.1 relies on three auxiliary lemmas. The first lemma states that the betting fraction β_t changes slowly.

Lemma 3.3. *For all $t \geq 1$, $|\beta_{t+1} - \beta_t| \leq 2/(Ct)$.*

Proof of Lemma 3.3. The result for $t = 1$ trivially holds. We only consider $t \geq 2$.

Since the Euclidean projection to a closed convex set is contractive, we have

$$\left|\Pi_{\mathcal{B}_t}(\hat{\beta}_t) - \Pi_{\mathcal{B}_t}(\hat{\beta}_{t+1})\right| \leq \left|\hat{\beta}_t - \hat{\beta}_{t+1}\right| = \left|\frac{\tilde{g}_t + 2C^2\hat{\beta}_t}{2C^2 t}\right| \leq \frac{G}{C^2 t}.$$

Moreover,

$$\left|\Pi_{\mathcal{B}_t}(\hat{\beta}_{t+1}) - \Pi_{\mathcal{B}_{t+1}}(\hat{\beta}_{t+1})\right| \leq \left|\frac{1}{\sqrt{2}C\sqrt{t-1}} - \frac{1}{\sqrt{2}C\sqrt{t}}\right| \leq \frac{1}{2\sqrt{2}C\sqrt{t}(t-1)} \leq \frac{1}{Ct}.$$

Applying the triangle inequality yields the result. □

The second auxiliary lemma quantifies the movement of Algorithm 3.1 using Wealth_t. By doing this, bounding the switching cost (Part 2 of Theorem 3.1) reduces to bounding the growth of Wealth_t.

Lemma 3.4. *For all $t \geq 1$,*

$$|\tilde{x}_t - \tilde{x}_{t+1}| \leq \frac{6}{Ct}\text{Wealth}_{t-1}.$$

Proof of Lemma 3.4. Assume $t > 1$ for the rest of this proof; the case of $t = 1$ can be verified similarly. Starting from (3.3), some simple algebra yields

$$\begin{aligned}
\tilde{x}_{t+1} - \tilde{x}_t &= \beta_{t+1}\text{Wealth}_t - \beta_t\text{Wealth}_{t-1} \\
&= \left(\beta_{t+1} - \beta_t - \beta_{t+1}\tilde{g}_t\beta_t - \beta_{t+1}\beta_t\frac{\gamma}{\sqrt{t}}\right)\text{Wealth}_{t-1} \\
&\qquad - \lambda\beta_{t+1}\left|\beta_{t+1}\text{Wealth}_t - \beta_t\text{Wealth}_{t-1}\right|.
\end{aligned}$$

From Lemma 3.1, $\text{Wealth}_{t-1} > 0$, therefore,

$$\begin{aligned}
&(1 - \lambda\beta_{t+1})\left|\beta_{t+1}\text{Wealth}_t - \beta_t\text{Wealth}_{t-1}\right| \\
\leq\ &\left|\beta_{t+1} - \beta_t - \beta_{t+1}\tilde{g}_t\beta_t - \beta_{t+1}\beta_t\frac{\gamma}{\sqrt{t}}\right|\text{Wealth}_{t-1}.
\end{aligned}$$

Note that $1 - \lambda\beta_{t+1} \geq 1/2$.

$$\begin{aligned}|\beta_{t+1}\text{Wealth}_t - \beta_t\text{Wealth}_{t-1}| &\leq 2\left|\beta_{t+1} - \beta_t - \beta_{t+1}\tilde{g}_t\beta_t - \beta_{t+1}\beta_t\frac{\gamma}{\sqrt{t}}\right|\text{Wealth}_{t-1} \\ &\leq 2\left|\beta_{t+1} - \beta_t\right|\text{Wealth}_{t-1} + 2\beta_t\beta_{t+1}\left|\tilde{g}_t + \frac{\gamma}{\sqrt{t}}\right|\text{Wealth}_{t-1}.\end{aligned}$$

Applying Lemma 3.3 and the definition of β_t and β_{t+1},

$$\|\tilde{x}_t - \tilde{x}_{t+1}\| \leq \left(\frac{4}{Ct} + \frac{2C}{2C^2\sqrt{t(t-1)}}\right)\text{Wealth}_{t-1} \leq \frac{6}{Ct}\text{Wealth}_{t-1}. \qquad \square$$

Following the reasoning above, the third auxiliary lemma bounds the growth rate of Wealth_t, which could be of special interest. The key idea is that, the surrogate loss (Line 3 of Algorithm 3.1) incentivizes the *unconstrained prediction* $\tilde{x}_t$ to be bounded. Equivalently, the betting amount in the coin-betting algorithm is bounded, and hence the wealth cannot grow too fast.

Notably, our proof makes a novel use of the black-box reduction from unconstrained OLO to constrained OLO (Algorithm 2.2): we *do not use it as a black-box*, but rather analyze its impact on the unconstrained algorithm.

Lemma 3.5. *For all $t \geq 1$, $\text{Wealth}_t \leq 4\bar{R}C\sqrt{t}$.*

Proof of Lemma 3.5. Note that from Lemma 3.1, $\text{Wealth}_t \geq 0$. Additionally from our definition of β_t, we have $\beta_t, x_t, \tilde{x}_t \geq 0$.

We prove this lemma in three steps. First, we show a weaker result, $\text{Wealth}_t \leq G\bar{R}(t+1)$. Using this result, we then prove that $\tilde{x}_t \leq 2\sqrt{2}\bar{R}$. In other words, even though $\tilde{x}_t$ is the output of a coin-betting-based OLO algorithm that works in the unbounded domain, *it is actually bounded* due to the effect of the surrogate losses. Finally, we revisit wealth and show that $\text{Wealth}_t \leq 4\bar{R}C\sqrt{t}$.

Step 1 Prove that for all $t \geq 0$, $\text{Wealth}_t \leq G\bar{R}(t+1)$.

Consider the two cases in the definition of $\tilde{g}_t$. If $g_t\tilde{x}_t \geq g_t x_t$, then $\tilde{g}_t = g_t$, and

$$\begin{aligned}\text{Wealth}_t &= \text{Wealth}_{t-1} - \tilde{g}_t\tilde{x}_t - \lambda|\tilde{x}_t - \tilde{x}_{t+1}| - \frac{\gamma}{\sqrt{t}}|\tilde{x}_t| \\ &\leq \text{Wealth}_{t-1} - g_t x_t \leq \text{Wealth}_{t-1} + |g_t|\bar{R}.\end{aligned}$$

If $g_t\tilde{x}_t < g_t x_t$, then $\tilde{g}_t = 0$ and $\text{Wealth}_t \leq \text{Wealth}_{t-1}$. An induction and $\varepsilon \leq G\bar{R}$ yield the result.

Step 2 Prove that for all $t \geq 1$, $\tilde{x}_t \leq 2\sqrt{2}\bar{R}$.

This holds trivially for $t = 1$. We use induction: suppose this result holds for t, and we need to show $\tilde{x}_{t+1} \leq 2\sqrt{2}\bar{R}$. There are two cases: (1) $\tilde{x}_t \notin \mathcal{V}_{1d}$; (2) $\tilde{x}_t \in \mathcal{V}_{1d}$. Note that the first case is only possible when $t > 1$.

- **Case (1.1)** $\tilde{x}_t \notin \mathcal{V}_{1d}$, $g_t\tilde{x}_t \geq g_t x_t$.

  In this case, $\tilde{g}_t = g_t \geq 0$ and $g_t x_t \geq 0$. It follows,

  $$\text{Wealth}_t \leq \text{Wealth}_{t-1} - g_t x_t \leq \text{Wealth}_{t-1}.$$

  Next we consider the three cases of β_t.

  (i) First, note that $\beta_t \neq 0$; otherwise $\tilde{x}_t = \beta_t \text{Wealth}_{t-1} = 0 \in \mathcal{V}_{1d}$.

  (ii) If $\beta_t = \hat{\beta}_t = -\sum_{i=1}^{t-1}\tilde{g}_i/[2C^2(t-1)]$, then

  $$\beta_{t+1} \leq \left|\hat{\beta}_{t+1}\right| = \frac{1}{2C^2 t}\left|-\sum_{i=1}^{t}\tilde{g}_i\right| = \frac{|2C^2(t-1)\beta_t - g_t|}{2C^2 t} \leq \max\left\{\frac{t-1}{t}\beta_t, \frac{g_t}{2C^2 t}\right\}.$$

  The last inequality is due to $\beta_t, g_t \geq 0$. Therefore,

  $$\begin{aligned}\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t &\leq \max\left\{\beta_t \text{Wealth}_{t-1}, G\text{Wealth}_{t-1}/(2C^2 t)\right\} \\ &\leq \max\{2\sqrt{2}\bar{R}, G^2\bar{R}/(2C^2)\} \leq 2\sqrt{2}\bar{R},\end{aligned}$$

  where we use the result from Step 1.

(iii) If $\beta_t = 1/(C\sqrt{2(t-1)})$, then

$$\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \le \frac{1}{C\sqrt{2t}}\text{Wealth}_{t-1} \le \frac{1}{C\sqrt{2(t-1)}}\text{Wealth}_{t-1} = \beta_t\text{Wealth}_{t-1} \le 2\sqrt{2}\bar{R}.$$

- **Case (1.2)** $\tilde{x}_t \notin \mathcal{V}_{1d}$, $g_t\tilde{x}_t < g_t x_t$.

  In this case, $\tilde{g}_t = 0$ and $\text{Wealth}_t \le \text{Wealth}_{t-1}$. Same as Case (1.1), $\beta_t \neq 0$, leading to $\hat{\beta}_t \ge 0$ and $\beta_t = \min\{\hat{\beta}_t, 1/(C\sqrt{2(t-1)})\}$. Also note that

$$\left|\hat{\beta}_{t+1}\right| = \frac{1}{2C^2t}\left|-\sum_{i=1}^{t}\tilde{g}_i\right| = \frac{1}{2C^2t}\left|-\sum_{i=1}^{t-1}\tilde{g}_i\right| \le \frac{1}{2C^2(t-1)}\left|-\sum_{i=1}^{t-1}\tilde{g}_i\right| = \left|\hat{\beta}_t\right|.$$

  Therefore,

$$\beta_{t+1} \le \min\left\{\left|\hat{\beta}_{t+1}\right|, \frac{1}{C\sqrt{2t}}\right\} \le \min\left\{\left|\hat{\beta}_t\right|, \frac{1}{C\sqrt{2(t-1)}}\right\} = \beta_t,$$

  and $\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \le \beta_t\text{Wealth}_{t-1} \le \tilde{x}_t \le 2\sqrt{2}\bar{R}$.

- **Case (2)** $\tilde{x}_t \in \mathcal{V}_{1d}$.

  In this case, $\tilde{x}_t = x_t$ and $\tilde{g}_t = g_t$. $\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \le (1 - g_t\beta_t)\beta_{t+1}\text{Wealth}_{t-1}$.

  If $t = 1$, then $\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \le \sqrt{2}G\bar{R}/C \le \sqrt{2}\bar{R}$, where we use $\text{Wealth}_1 \le 2G\bar{R}$ from Step 1 and $\beta_2 \le 1/(\sqrt{2}C)$.

  If $t > 1$, we consider the three cases of β_t as follows. (For the rest of the discussion assume $t > 1$.)

  (i) If $\beta_t = 0$, then from Lemma 3.3 we have $\beta_{t+1} \le 2/(Ct)$, and $\tilde{x}_{t+1} \le (1 - g_t\beta_t)\beta_{t+1}\text{Wealth}_{t-1} = \beta_{t+1}\text{Wealth}_{t-1} \le 2G\bar{R}/C \le 2\bar{R}$.

  (ii) If $\beta_t = \hat{\beta}_t = -\sum_{i=1}^{t-1}\tilde{g}_i/[2C^2(t-1)]$, then

$$\beta_{t+1} \le \left|\hat{\beta}_{t+1}\right| = \frac{1}{2C^2t}\left|-\sum_{i=1}^{t}\tilde{g}_i\right| = \frac{|2C^2(t-1)\beta_t - g_t|}{2C^2t} \le \frac{t-1}{t}\beta_t + \frac{G}{2C^2t}.$$

  Note that since $\tilde{x}_t \in \mathcal{V}_{1d}$, we have $\beta_t\text{Wealth}_{t-1} \le \bar{R}$. Using $\tilde{x}_{t+1} \le (1 -$

$g_t\beta_t)\beta_{t+1}\text{Wealth}_{t-1}$ and $|g_t\beta_t| \leq 1/2$ we have

$$\tilde{x}_{t+1} \leq \frac{3}{2}\left(\frac{t-1}{t}\beta_t\text{Wealth}_{t-1} + \frac{G}{2C^2t}\text{Wealth}_{t-1}\right) \leq \frac{3}{2}\left(1 + \frac{G^2}{2C^2}\right)\bar{R} \leq 2\sqrt{2}\bar{R}.$$

(iii) If $\beta_t = 1/(C\sqrt{2(t-1)})$, then

$$\beta_{t+1} \leq 1/(C\sqrt{2t}) \leq 1/(C\sqrt{2(t-1)}) = \beta_t,$$

$$\tilde{x}_{t+1} \leq (1 - g_t\beta_t)\beta_{t+1}\text{Wealth}_{t-1} \leq 2\beta_t\text{Wealth}_{t-1} \leq 2\bar{R}.$$

Step 3 Prove that for all $t \geq 1$, $\text{Wealth}_t \leq 4\bar{R}C\sqrt{t}$.

Considering β_{t+1}, there are three cases: (1) $\beta_{t+1} = 1/(C\sqrt{2t})$; (2) $\beta_{t+1} = \hat{\beta}_{t+1}$; and (3) $\beta_{t+1} = 0$. For the first case, this result follows from $\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \leq 2\sqrt{2}\bar{R}$. Now consider the second case.

$$\begin{aligned}
\log \text{Wealth}_t &\leq \log\varepsilon + \sum_{i=1}^{t}\log(1 - \tilde{g}_i\beta_i) \\
&\leq \log\varepsilon - \sum_{i=1}^{t}\tilde{g}_i\beta_i \\
&= \log\varepsilon - \sum_{i=1}^{t}\left(\tilde{g}_i\beta_i + C^2\beta_i^2\right) + C^2\sum_{i=1}^{t}\beta_i^2.
\end{aligned}$$

β_t is the output of *Follow the Leader* (FTL) on the strongly convex losses $\psi_t(\beta) = \tilde{g}_t\beta + C^2\beta^2 + I\{0 \leq \beta \leq 1/(C\sqrt{2t})\}(\beta)$, where $I\{0 \leq \beta \leq 1/(C\sqrt{2t})\}(\beta)$ is a convex function of β that equals 0 when $0 \leq \beta \leq 1/(C\sqrt{2t})$ and infinity otherwise. Therefore we can use standard FTL results to show that the regret is non-negative.

Let $F_t(\beta) = \sum_{i=1}^{t-1}\psi_i(\beta)$, then $\beta_t \in \arg\min F_t(\beta)$. From Lemma 7.1 of (Orabona, 2019), for any $u \in \mathbb{R}$,

$$\sum_{i=1}^{t}\left[\psi_i(\beta_i) - \psi_i(u)\right] = \sum_{i=1}^{t}\left[F_i(\beta_i) - F_{i+1}(\beta_{i+1}) + \psi_i(\beta_i)\right] + F_{t+1}(\beta_{t+1}) - F_{t+1}(u).$$

Note that if $u = \beta_{t+1}$, we have RHS ≥ 0. Therefore,

$$\begin{aligned}\log \text{Wealth}_t &\leq \log \varepsilon - \min_{0 \leq \beta \leq 1/(C\sqrt{2t})} \sum_{i=1}^{t} \left(\tilde{g}_i \beta + C^2 \beta^2\right) + C^2 \sum_{i=1}^{t} \beta_i^2 \\ &\leq \log \varepsilon - \min_{\beta \in \mathbb{R}} \sum_{i=1}^{t} \left(\tilde{g}_i \beta + C^2 \beta^2\right) + C^2 \sum_{i=1}^{t} \beta_i^2 \\ &\leq \log \varepsilon + \frac{\left(\sum_{i=1}^{t} \tilde{g}_i\right)^2}{4C^2 t} + \frac{1}{2} \sum_{\tau=1}^{t-1} \tau^{-1}.\end{aligned}$$

The last term is bounded by $(1 + \log t)/2$. From the assumption of the second case, $|\sum_{i=1}^{t} \tilde{g}_i| < C\sqrt{2t}$. Combining everything we have $\log \text{Wealth}_t \leq 1 + \log \varepsilon + (\log t)/2$ and $\text{Wealth}_t \leq e\varepsilon\sqrt{t} \leq e\bar{R}C\sqrt{t}$.

Finally consider the third case. Same as the above, we have

$$\log \text{Wealth}_t \leq \log \varepsilon - \min_{0 \leq \beta \leq 1/(C\sqrt{2t})} \sum_{i=1}^{t} \left(\tilde{g}_i \beta + C^2 \beta^2\right) + C^2 \sum_{i=1}^{t} \beta_i^2.$$

Since $\beta_{t+1} = 0$, we have $\sum_{i=1}^{t} \tilde{g}_i \geq 0$. Therefore,

$$\log \text{Wealth}_t \leq \log \varepsilon + C^2 \sum_{i=1}^{t} \beta_i^2 \leq \log \varepsilon + \frac{1}{2}(1 + \log t),$$

and $\text{Wealth}_t \leq \sqrt{e}\bar{R}C\sqrt{t}$. □

3.8.2 The second algorithm

Theorem 3.2

Proof of Theorem 3.2. Combining Lemma 2.5, 3.9 and 3.2, we have

$$\sum_{t=1}^{T} \left(g_t x_t + \lambda \left|x_t - x_{t+1}\right|\right) \leq -G \cdot V_\alpha(T, S_T).$$

Consider $V_\alpha(T, S_T)$ as a function of S_T; let us write $V^*_{\alpha,T}(\cdot)$ as its Fenchel conjugate.

Then, in any adversarial environment Env, the augmented regret can be bounded as

$$\begin{aligned}\text{Regret}_T^{\lambda}(Env, u) &= \sum_{t=1}^{T} g_t(x_t - u) + \lambda \sum_{t=1}^{T-1} |x_t - x_{t+1}| \\ &\leq G \cdot uS_T + \sum_{t=1}^{T} \left(g_t x_t + \lambda |x_t - x_{t+1}|\right) \\ &\leq G \left[uS_T - V_\alpha(T, S_T)\right] \\ &\leq G \cdot V_{\alpha,T}^*(u).\end{aligned}$$

The last step is to bound $V_{\alpha,T}^*(u)$, which also follows from a standard proof strategy.

$$V_{\alpha,T}^*(u) = \sup_{S \in \mathbb{R}} uS - V_\alpha(T, S).$$

It is clear that the supremum is uniquely achieved; let S^* be the maximizing argument. Then,

$$u = \nabla_S V_\alpha(T, S^*) = C \int_0^{S^*/\sqrt{4\alpha T}} \exp\left(x^2\right) dx.$$

If we define $\text{erfi}(z) = \int_0^z \exp(x^2) dx$ (note that it a scaled version of the conventional *imaginary error function*), then $S^* = \sqrt{4\alpha T} \cdot \text{erfi}^{-1}\left(uC^{-1}\right)$.

$$V_{\alpha,T}^*(u) = uS^* - V_\alpha(T, S^*) \leq uS^* - V_\alpha(T, 0) = C\sqrt{\alpha T} + |u| \sqrt{4\alpha T} \cdot \text{erfi}^{-1}\left(uC^{-1}\right).$$

Finally, as shown in (Zhang et al., 2022b, Theorem 4), $\text{erfi}^{-1}(x) \leq 1 + \sqrt{\log(1+x)}$. Combining the above completes the proof. □

The proof relies on the following lemmas. We start by proving a few basic properties of Algorithm 3.2 and the potential function V_α (3.4).

Lemma 3.6. *If the potential $V(t, S)$ is even and convex in S, then $\bar{\nabla}_S V(t, S)$ is odd and monotonically increasing in S.*

Proof of Lemma 3.6. $\bar{\nabla}_S V(t,S)$ is odd due to the simple relation

$$\begin{aligned}\bar{\nabla}_S V(t,-S) &= \frac{1}{2}\left[V(t,-S+1) - V(t,-S-1)\right] \\ &= \frac{1}{2}\left[V(t,S-1) - V(t,S+1)\right] \\ &= -\bar{\nabla}_S V(t,S).\end{aligned}$$

As for the monotonicity, it is equivalent to showing for all $\delta \geq 0$,

$$V(t,S+1+\delta) - V(S-1+\delta) \geq V(t,S+1) - V(S-1).$$

This follows from the convexity of $V(t,\cdot)$, as

$$V(t,S+1) \leq \frac{2}{2+\delta} V(t,S+1+\delta) + \frac{\delta}{2+\delta} V(t,S-1),$$

$$V(t,S-1+\delta) \leq \frac{\delta}{2+\delta} V(t,S+1+\delta) + \frac{2}{2+\delta} V(t,S-1). \qquad \square$$

For the potential function V_α, we compute its continuous partial derivatives. The proof is straightforward calculation, therefore omitted.

Lemma 3.7. *For any $\alpha > 0$, V_α defined in (3.4) is even and convex. Moreover,*

$$\nabla_S V_\alpha(t,S) = C \int_0^{S/\sqrt{4\alpha t}} \exp\left(x^2\right) dx,$$

$$\nabla_{SS} V_\alpha(t,S) = \frac{C}{2\sqrt{\alpha t}} \exp\left(\frac{S^2}{4\alpha t}\right),$$

$$\nabla_{SSS} V_\alpha(t,S) = \frac{CS}{4(\alpha t)^{3/2}} \exp\left(\frac{S^2}{4\alpha t}\right),$$

$$\nabla_t V_\alpha(t,S) = -\frac{C\sqrt{\alpha}}{2\sqrt{t}} \exp\left(\frac{S^2}{4\alpha t}\right).$$

Based on the above, the discrete derivative $\bar{\nabla}_S V_\alpha$ has the following properties.

Lemma 3.8. *For all $\alpha > 0$, $t \geq 0$ and $S \geq 0$,*

1. *$\bar{\nabla}_S V_\alpha(t,S)$ as a function of t is decreasing and convex;*

2. $\bar{\nabla}_S V_\alpha(t,S)$ as a function of S is convex.

Proof of Lemma 3.8. Considering the first part of the lemma,

$$\begin{aligned}\nabla_t \left[\bar{\nabla}_S V_\alpha(t,S)\right] &= \frac{1}{2}\left[\nabla_t V_\alpha(t,S+1) - \nabla_t V_\alpha(t,S-1)\right] \\ &= -\frac{C\sqrt{\alpha}}{4\sqrt{t}} \exp\left(\frac{S^2+1}{4\alpha t}\right) \sinh\left(\frac{S}{2\alpha t}\right),\end{aligned}$$

which, when $S \geq 0$, is negative and increasing in t. Therefore, $\bar{\nabla}_S V_\alpha(t,S)$ as a function of t is decreasing and convex. Similarly,

$$\nabla_S \left[\bar{\nabla}_S V_\alpha(t,S)\right] = \frac{1}{2}\left[\nabla_S V_\alpha(t,S+1) - \nabla_S V_\alpha(t,S-1)\right] = \frac{C}{2} \int_{(S-1)/\sqrt{4\alpha t}}^{(S+1)/\sqrt{4\alpha t}} \exp\left(x^2\right) dx,$$

which is increasing in S. Therefore, $\bar{\nabla}_S V_\alpha(t,S)$ as a function of S is convex. □

Now, we are ready to prove two key lemmas for Theorem 3.2.

Lemma 3.9. *For all $\alpha > 0$, consider Algorithm 3.2 induced by the potential V_α. For all $t \in \mathbb{N}_+$,*

$$|x_t - x_{t+1}| \leq \bar{\nabla}_S V_\alpha(t, S_{t-1}+1) - \bar{\nabla}_S V_\alpha(t, S_{t-1}-1).$$

Proof of Lemma 3.9. First, since $\bar{\nabla}_S V_\alpha(t,S)$ is monotonic in S due to Lemma 3.6, we have

$$\begin{aligned}|x_t - x_{t+1}| &= \left|\bar{\nabla}_S V_\alpha(t, S_{t-1}) - \bar{\nabla}_S V_\alpha(t+1, S_t)\right| \\ &\leq \max_{c=\pm 1} \left|\bar{\nabla}_S V_\alpha(t, S_{t-1}) - \bar{\nabla}_S V_\alpha(t+1, S_{t-1}+c)\right|.\end{aligned}$$

For clarity, from the RHS we define

$$f(t,S) := \max_{c=\pm 1} \left|\bar{\nabla}_S V_\alpha(t,S) - \bar{\nabla}_S V_\alpha(t+1, S+c)\right|.$$

It is even in S, as

$$\begin{aligned}f(t,-S) &= \max_{c=\pm 1} \left|\bar{\nabla}_S V_\alpha(t,-S) - \bar{\nabla}_S V_\alpha(t+1,-S+c)\right| \\ &= \max_{c=\pm 1} \left|-\bar{\nabla}_S V_\alpha(t,S) + \bar{\nabla}_S V_\alpha(t+1,S-c)\right| \qquad \text{(Lemma 3.6)} \\ &= \max_{c=\pm 1} \left|\bar{\nabla}_S V_\alpha(t,S) - \bar{\nabla}_S V_\alpha(t+1,S-c)\right| = f(t,S).\end{aligned}$$

Therefore, we can restrict the rest of the proof to $S \geq 0$, and the remaining task is to upper bound $f(t, S)$ for all $0 \leq S \leq t-1$.

From Lemma 3.6 and 3.8,

$$\bar{\nabla}_S V_\alpha(t+1, S-1) \leq \bar{\nabla}_S V_\alpha(t+1, S) \leq \bar{\nabla}_S V_\alpha(t, S),$$

$$\bar{\nabla}_S V_\alpha(t+1, S-1) \leq \bar{\nabla}_S V_\alpha(t+1, S+1).$$

Therefore, if $\bar{\nabla}_S V_\alpha(t+1, S-1) \leq \bar{\nabla}_S V_\alpha(t, S) \leq \bar{\nabla}_S V_\alpha(t+1, S+1)$, then

$$\begin{aligned} & f(t, S) \\ = \; & \max\left\{\left|\bar{\nabla}_S V_\alpha(t, S) - \bar{\nabla}_S V_\alpha(t+1, S-1)\right|, \left|\bar{\nabla}_S V_\alpha(t, S) - \bar{\nabla}_S V_\alpha(t+1, S+1)\right|\right\} \\ \leq \; & \bar{\nabla}_S V_\alpha(t+1, S+1) - \bar{\nabla}_S V_\alpha(t+1, S-1). \end{aligned}$$

On the other hand, if $\bar{\nabla}_S V_\alpha(t+1, S+1) \leq \bar{\nabla}_S V_\alpha(t, S)$, then

$$f(t, S) = \bar{\nabla}_S V_\alpha(t, S) - \bar{\nabla}_S V_\alpha(t+1, S-1).$$

Combining the above,

$$\begin{aligned} & f(t, S) \leq \\ & \max\left\{\bar{\nabla}_S V_\alpha(t, S) - \bar{\nabla}_S V_\alpha(t+1, S-1), \bar{\nabla}_S V_\alpha(t+1, S+1) - \bar{\nabla}_S V_\alpha(t+1, S-1)\right\}. \end{aligned}$$

Our goal next is to upper bound $f(t, S)$ by $\bar{\nabla}_S V_\alpha(t, S+1) - \bar{\nabla}_S V_\alpha(t, S-1)$, which can be divided into two cases.

Case 1 We aim to show that

$$\bar{\nabla}_S V_\alpha(t, S) - \bar{\nabla}_S V_\alpha(t+1, S-1) \leq \bar{\nabla}_S V_\alpha(t, S+1) - \bar{\nabla}_S V_\alpha(t, S-1),$$

which is equivalent to

$$\bar{\nabla}_S V_\alpha(t, S-1) - \bar{\nabla}_S V_\alpha(t+1, S-1) \leq \bar{\nabla}_S V_\alpha(t, S+1) - \bar{\nabla}_S V_\alpha(t, S). \tag{3.11}$$

Note that this trivially holds when $0 \leq S < 1$: due to Lemma 3.8, the RHS is always positive; however, the LHS is negative due to $\bar{\nabla}_S V_\alpha(t, S-1)$ being increasing in t (Lemma 3.6 and 3.8 Part 1). Therefore, we only need to show (3.11) for all $S \geq 1$.

To this end, with $S \geq 1$, we apply the convexity of $\bar{\nabla}_S V_\alpha$ from Lemma 3.8:

$$\bar{\nabla}_S V_\alpha(t, S+1) - \bar{\nabla}_S V_\alpha(t, S) \geq \nabla_S \left[\bar{\nabla}_S V_\alpha(t, S)\right],$$

$$\bar{\nabla}_S V_\alpha(t, S-1) - \bar{\nabla}_S V_\alpha(t+1, S-1) \leq -\nabla_t \left[\bar{\nabla}_S V_\alpha(t, S-1)\right].$$

Consequently, it suffices to show that

$$-\nabla_t \left[\bar{\nabla}_S V_\alpha(t, S-1)\right] \leq \nabla_S \left[\bar{\nabla}_S V_\alpha(t, S)\right].$$

Now it is time to invoke the specific form of V_α. We may reuse $\nabla_S \left[\bar{\nabla}_S V_\alpha(t, S)\right]$ and $\nabla_t \left[\bar{\nabla}_S V_\alpha(t, S)\right]$ calculated from the proof of Lemma 3.8.

$$\nabla_S \left[\bar{\nabla}_S V_\alpha(t, S)\right] = \frac{C}{2} \int_{(S-1)/\sqrt{4\alpha t}}^{(S+1)/\sqrt{4\alpha t}} \exp\left(x^2\right) dx \geq \frac{C}{2\sqrt{\alpha t}} \exp\left(\frac{S^2}{4\alpha t}\right),$$

and for all $1 \leq S \leq t-1$,

$$\begin{aligned}
-\nabla_t \left[\bar{\nabla}_S V_\alpha(t, S-1)\right] &= \frac{C\sqrt{\alpha}}{4\sqrt{t}} \exp\left(\frac{(S-1)^2+1}{4\alpha t}\right) \sinh\left(\frac{S-1}{2\alpha t}\right) \\
&= \frac{C\sqrt{\alpha}}{8\sqrt{t}} \exp\left(\frac{S^2}{4\alpha t}\right) \left[1 - \exp\left(\frac{-S+1}{\alpha t}\right)\right] \\
&\leq \frac{C\sqrt{\alpha}}{8\sqrt{t}} \exp\left(\frac{S^2}{4\alpha t}\right) \left[1 - \exp\left(-\frac{1}{\alpha}\right)\right] \qquad (S-1 \leq t) \\
&\leq \frac{C}{8\sqrt{\alpha t}} \exp\left(\frac{S^2}{4\alpha t}\right). \qquad (\exp(x) \geq x+1)
\end{aligned}$$

Therefore, $-\nabla_t \left[\bar{\nabla}_S V_\alpha(t, S-1)\right] \leq \nabla_S \left[\bar{\nabla}_S V_\alpha(t, S)\right]$, which proves (3.11) and concludes Case 1.

Case 2 We aim to show that

$$\bar{\nabla}_S V_\alpha(t+1, S+1) - \bar{\nabla}_S V_\alpha(t+1, S-1) \leq \bar{\nabla}_S V_\alpha(t, S+1) - \bar{\nabla}_S V_\alpha(t, S-1).$$

This is straightforward, as

$$\begin{aligned}
& \nabla_t \left[\bar{\nabla}_S V_\alpha(t, S+1) - \bar{\nabla}_S V_\alpha(t, S-1)\right] \\
= \; & \frac{1}{2}\left[\nabla_t V_\alpha(t, S+2) + \nabla_t V_\alpha(t, S-2) - 2\nabla_t V_\alpha(t, S)\right] \\
= \; & -\frac{C\sqrt{\alpha}}{4\sqrt{t}}\left[\exp\left(\frac{(S+2)^2}{4\alpha t}\right) + \exp\left(\frac{(S-2)^2}{4\alpha t}\right) - 2\exp\left(\frac{S^2}{4\alpha t}\right)\right] \\
\leq \; & 0. \qquad \text{(convexity)}
\end{aligned}$$

Combining the two cases, we can upper bound $f(t, S)$ by $\bar{\nabla}_S V_\alpha(t, S+1) - \bar{\nabla}_S V_\alpha(t, S-1)$, which completes the proof. □

Lemma 3.2

Proof of Lemma 3.2. We restate the definition of Δ_t for easier reference.

$$\Delta_t = \bar{\nabla}_t V_\alpha(t, S_{t-1}) + \frac{1}{2}\bar{\nabla}_{SS} V_\alpha(t, S_{t-1}) + G^{-1}\lambda\left[\bar{\nabla}_S V_\alpha(t, S_{t-1}+1) - \bar{\nabla}_S V_\alpha(t, S_{t-1}-1)\right].$$

Let us define a function $g(t, S)$ as

$$\begin{aligned}
g(t, S) := \frac{1}{2}V_\alpha(t, S+1) + \frac{1}{2}V_\alpha(t, S-1) - V_\alpha(t-1, S) \\
+ \frac{\lambda}{2G}\left[V_\alpha(t, S+2) + V_\alpha(t, S-2) - 2V_\alpha(t, S)\right],
\end{aligned}$$

then from the definition of discrete derivatives, $\Delta_t = g(t, S_{t-1})$. Also note that $g(t, S)$ is even in S, so we can only focus on positive values of S. The rest of the proof will show $g(t, S) \leq 0$ for all $t \in \mathbb{N}_+$ and $S \geq 0$.

Let us start from the special case, $t = 1$. S can only take the value 0, therefore $g(1, S) = g(1, 0)$. We now present a general result that upper bounds $g(t, 0)$ for all

$t \in \mathbb{N}_+$:

$$
\begin{aligned}
& g(t,0) \\
=& V_\alpha(t,1) - V_\alpha(t-1,0) + G^{-1}\lambda V_\alpha(t,2) - G^{-1}\lambda V_\alpha(t,0) \\
=& C\sqrt{\alpha t}\left[2\int_0^{1/\sqrt{4\alpha t}}\left(\int_0^u \exp(x^2)dx\right)du + \frac{2\lambda}{G}\int_0^{1/\sqrt{\alpha t}}\left(\int_0^u \exp(x^2)dx\right)du + \sqrt{\frac{t-1}{t}} - 1\right] \\
\le& C\sqrt{\alpha t}\left[\frac{1}{\sqrt{4\alpha t}}\int_0^{1/\sqrt{4\alpha t}} \exp(x^2)dx + \frac{2\lambda}{G}\cdot\frac{1}{2}\cdot\frac{1}{\sqrt{\alpha t}}\int_0^{1/\sqrt{\alpha t}} \exp(x^2)dx + \sqrt{\frac{t-1}{t}} - 1\right] \\
& \qquad\qquad (\mathrm{erfi}(x) \text{ is increasing and convex on } \mathbb{R}_+) \\
\le& C\sqrt{\alpha t}\left[\frac{1}{4\alpha t}\exp\left(\frac{1}{4\alpha t}\right) + \frac{\lambda}{G\alpha t}\exp\left(\frac{1}{\alpha t}\right) + \sqrt{\frac{t-1}{t}} - 1\right].
\end{aligned}
$$

Since $\sqrt{1+x} \le 1 + x/2$ for all $x \ge -1$, we have $\sqrt{(t-1)/t} - 1 \le -1/(2t)$. Therefore, if $\alpha \ge 4\lambda G^{-1} + 2$, then

$$
\begin{aligned}
g(t,0) &\le C\sqrt{\alpha t}\left[\frac{\lambda G^{-1} + (1/4)}{\alpha t}\exp\left(\frac{1}{\alpha t}\right) - \frac{1}{2t}\right] \\
&\le \frac{C\sqrt{\alpha}}{\sqrt{t}}\left[\frac{\lambda G^{-1} + (1/4)}{\alpha}\exp\left(\frac{1}{2}\right) - \frac{1}{2}\right] \le 0. \quad (3.12)
\end{aligned}
$$

As its special case, we have $g(1,0) \le 0$, which concludes the proof of the special case ($t = 1$).

Next, we prove $g(t,S) \le 0$ for general t, i.e., $t \ge 2$. Our overall strategy is to show that for all $0 \le S \le t-1$, $g(t,S) \le g(t,0)$, and then using the argument above we have $g(t,0) \le 0$. Concretely, let us calculate the first and second order derivatives of

$g(t,S)$, using Lemma 3.7.

$$\begin{aligned}
&\nabla_S g(t,S) \\
&\quad = \frac{C}{2}\left[\int_0^{(S+1)/\sqrt{4\alpha t}} \exp(x^2)dx + \int_0^{(S-1)/\sqrt{4\alpha t}} \exp(x^2)dx - 2\int_0^{S/\sqrt{4\alpha(t-1)}} \exp(x^2)dx\right] \\
&\qquad + \frac{\lambda C}{2G}\left[\int_0^{(S+2)/\sqrt{4\alpha t}} \exp(x^2)dx + \int_0^{(S-2)/\sqrt{4\alpha t}} \exp(x^2)dx - 2\int_0^{S/\sqrt{4\alpha t}} \exp(x^2)dx\right],
\end{aligned}$$

$$\begin{aligned}
&\nabla_{SS} g(t,S) \\
&= \frac{C}{4\sqrt{\alpha t}}\left[\frac{\lambda}{G}\exp\left(\frac{(S+2)^2}{4\alpha t}\right) + \exp\left(\frac{(S+1)^2}{4\alpha t}\right) - \frac{2\lambda}{G}\exp\left(\frac{S^2}{4\alpha t}\right)\right. \\
&\quad \left. + \exp\left(\frac{(S-1)^2}{4\alpha t}\right) + \frac{\lambda}{G}\exp\left(\frac{(S-2)^2}{4\alpha t}\right)\right] - \frac{C}{2\sqrt{\alpha(t-1)}}\exp\left(\frac{S^2}{4\alpha(t-1)}\right) \\
&= \frac{C}{2\sqrt{\alpha t}}\exp\left(\frac{S^2}{4\alpha t}\right)\left[\frac{\lambda}{G}\exp\left(\frac{1}{\alpha t}\right)\cosh\left(\frac{S}{\alpha t}\right) + \exp\left(\frac{1}{4\alpha t}\right)\cosh\left(\frac{S}{2\alpha t}\right)\right. \\
&\qquad \left. - \frac{\lambda}{G} - \sqrt{\frac{t}{t-1}}\exp\left(\frac{S^2}{4\alpha t(t-1)}\right)\right]. \qquad (3.13)
\end{aligned}$$

Notice that $\nabla_S g(t,0) = 0$. To proceed, we aim to prove $\nabla_{SS} g(t,S) \le 0$ for all $S \ge 0$, which then shows $g(t,S) \le g(t,0)$. To this end, we will show the sum inside the bracket in (3.13) is negative. Denote it as $h(t,S)$, and more specifically,

$$\begin{aligned}
h(t,S) := \frac{\lambda}{G}\exp\left(\frac{1}{\alpha t}\right)\cosh\left(\frac{S}{\alpha t}\right) + \exp\left(\frac{1}{4\alpha t}\right)\cosh\left(\frac{S}{2\alpha t}\right) \\
- \frac{\lambda}{G} - \sqrt{\frac{t}{t-1}}\exp\left(\frac{S^2}{4\alpha t(t-1)}\right).
\end{aligned}$$

The rest of the proof is divided into two steps: we first prove (i) $h(t,0) \le 0$; and then prove (ii) $\nabla_S h(t,S) \le 0$ for all $S \ge 0$.

Step 1: prove $h(t,0) \le 0$. From the definition of $h(t,S)$,

$$h(t,0) = \frac{\lambda}{G}\exp\left(\frac{1}{\alpha t}\right) + \exp\left(\frac{1}{4\alpha t}\right) - \frac{\lambda}{G} - \sqrt{\frac{t}{t-1}}.$$

Letting $x = 1/t$, then to prove $h(t,0) \le 0$ for all $t \ge 2$, it suffices to prove

$$\psi(x) := \frac{\lambda}{G}\exp\left(\frac{x}{\alpha}\right) + \exp\left(\frac{x}{4\alpha}\right) - \frac{\lambda}{G} - \sqrt{\frac{1}{1-x}} \le 0,$$

on the range $x \in (0, 1/2]$. $\psi(0) = 0$, and

$$\nabla_x \psi(x) = \frac{\lambda}{\alpha G}\exp\left(\frac{x}{\alpha}\right) + \frac{1}{4\alpha}\exp\left(\frac{x}{4\alpha}\right) - \frac{1}{2}(1-x)^{-3/2} \le \frac{4\lambda G^{-1}+1}{4\alpha}\exp\left(\frac{1}{2\alpha}\right) - \frac{1}{2},$$

which is negative when $\alpha \ge 4\lambda G^{-1} + 2$. Therefore, $h(t,0) \le 0$ for all $t \ge 2$.

Step 2: prove $\nabla_S h(t,S) \le 0$. Taking the derivative of $h(t,S)$,

$$\begin{aligned}\nabla_S h(t,S) &= \frac{\lambda}{\alpha t G}\exp\left(\frac{1}{\alpha t}\right)\sinh\left(\frac{S}{\alpha t}\right) + \frac{1}{2\alpha t}\exp\left(\frac{1}{4\alpha t}\right)\sinh\left(\frac{S}{2\alpha t}\right) \\ &\quad - \sqrt{\frac{t}{t-1}}\cdot\frac{S}{2\alpha t(t-1)}\exp\left(\frac{S^2}{4\alpha t(t-1)}\right) \\ &\le \left(\frac{\lambda}{\alpha t G} + \frac{1}{2\alpha t}\right)\exp\left(\frac{1}{\alpha t}\right)\sinh\left(\frac{S}{\alpha t}\right) - \frac{S}{2\alpha t^2}\sqrt{\frac{t}{t-1}}.\end{aligned}$$

Note that for all x, $\exp(-x) \ge 1 - x$, therefore for all $0 \le x < 1$, $\exp(x/2) \le \sqrt{1/(1-x)}$. Assigning x to $1/t$ which is less than 1, we have for all $\alpha \ge 2$,

$$\exp\left(\frac{1}{\alpha t}\right) \le \exp\left(\frac{1}{2t}\right) \le \sqrt{\frac{t}{t-1}}.$$

Moreover, for all $0 \le x \le 1$, $\sinh(x) \le 2x$. Therefore,

$$\begin{aligned}\nabla_S h(t,S) &\le \sqrt{\frac{t}{t-1}}\left[\frac{\lambda G^{-1} + (1/2)}{\alpha t}\sinh\left(\frac{S}{\alpha t}\right) - \frac{S}{2\alpha t^2}\right] \\ &\le \frac{S}{\alpha^2 t^2}\sqrt{\frac{t}{t-1}}\left[2\lambda G^{-1} + 1 - \frac{\alpha}{2}\right].\end{aligned}$$

When $\alpha \ge 4\lambda G^{-1} + 2$, $\nabla_S h(t,S) \le 0$ for all $t \ge 2$ and $S \ge 0$.

Concluding the above two steps, we have shown $h(t,S) \leq 0$. Plugging it back into (3.13), we have $\nabla_{SS} g(t,S) \leq 0$, which shows that for all $t \geq 2$ and $S \geq 0$, $g(t,S) \leq g(t,0)$. Finally, $g(t,0) \leq 0$ following (3.12). □

Theorem 3.3

Proof of Theorem 3.3. The first part of the theorem directly follows from (Cutkosky, 2020, Theorem 2) and the contraction property of 1D projections. As for the second part, we divide the proof into five steps.

Step 1: the "concentration" of S_t Without loss of generality, assume $S_{t-1} \geq 0$. Considering the prediction $\tilde{x}_t = \bar{\nabla}_S V_\alpha(t, S_{t-1})$ of the unconstrained base algorithm, there are two cases.

- **Case 1: $\tilde{x}_t \leq D$.** Due to convexity,

$$\begin{aligned}\tilde{x}_t &= \bar{\nabla}_S V_\alpha(t, S_{t-1}) \\ &= C\sqrt{\alpha t} \int_{(S_{t-1}-1)/\sqrt{4\alpha t}}^{(S_{t-1}+1)/\sqrt{4\alpha t}} \left(\int_0^u \exp(x^2) dx \right) du \geq C \int_0^{S_{t-1}/\sqrt{4\alpha t}} \exp(x^2) dx.\end{aligned}$$

  Similar to the proof of Theorem 3.2, if we define $\operatorname{erfi}(z) = \int_0^z \exp(x^2) dx$, then $S_{t-1} \leq \sqrt{4\alpha t} \cdot \operatorname{erfi}^{-1}(DC^{-1})$. As for the next round, $|S_t| \leq S_{t-1} + |g_t| / G \leq \sqrt{4\alpha t} \cdot \operatorname{erfi}^{-1}(DC^{-1}) + 1$.

- **Case 2: $\tilde{x}_t > D$.** In this case, since $x^* \in \mathcal{X}$, we have $\tilde{x}_t + x^*$ larger than the maximum element of $\mathcal{X}$, leading to $\tilde{x}_t + x^* > x_t$. Due to the definition of the surrogate loss, $\tilde{g}_t \geq 0$. Therefore, $|S_t| \leq \max\{|S_{t-1}|, |\tilde{g}_t / G|\} \leq \max\{|S_{t-1}|, 1\}$.

Combining the two cases and their analogous arguments for $S_{t-1} \leq 0$, we can see that for all t, $|S_t| \leq \max\left\{\sqrt{4\alpha t} \cdot \operatorname{erfi}^{-1}(DC^{-1}) + 1, |S_{t-1}|, 1\right\}$. By induction, we obtain for all t,

$$|S_t| \leq \sqrt{4\alpha t} \cdot \operatorname{erfi}^{-1}(DC^{-1}) + 1.$$

Step 2: bounding the switching cost using S_t Still, assume $S_{t-1} \geq 0$ without loss of generality. From Lemma 3.9,

$$\begin{aligned}
& |x_t - x_{t+1}| \\
\leq\ & \bar{\nabla}_S V_\alpha(t, S_{t-1}+1) - \bar{\nabla}_S V_\alpha(t, S_{t-1}-1) \\
=\ & C\sqrt{\alpha t}\left[\int_{S_{t-1}/\sqrt{4\alpha t}}^{(S_{t-1}+2)/\sqrt{4\alpha t}} \left(\int_0^u \exp(x^2)dx\right) du - \int_{(S_{t-1}-2)/\sqrt{4\alpha t}}^{S_{t-1}/\sqrt{4\alpha t}} \left(\int_0^u \exp(x^2)dx\right) du\right] \\
\leq\ & C\sqrt{\alpha t}\left[\frac{2}{\sqrt{4\alpha t}} \int_0^{(S_{t-1}+2)/\sqrt{4\alpha t}} \exp(x^2)dx - \frac{2}{\sqrt{4\alpha t}} \int_0^{(S_{t-1}-2)/\sqrt{4\alpha t}} \exp(x^2)dx\right] \\
=\ & C \int_{(S_{t-1}-2)/\sqrt{4\alpha t}}^{(S_{t-1}+2)/\sqrt{4\alpha t}} \exp(x^2)dx \leq \frac{2C}{\sqrt{\alpha t}} \exp\left(\frac{(S_{t-1}+2)^2}{4\alpha t}\right).
\end{aligned}$$

Step 3: plug in the concentration of S_t Next, we use the upper bound on S_{t-1} to show that $|x_t - x_{t+1}| = O(Ct^{-1/2}\exp[(\text{erfi}^{-1}(DC^{-1}))^2])$. To this end, we discuss two cases regarding how the "concentration" bound (i.e., $O(\sqrt{t})$) compares to the trivial bound (i.e., $S_t \leq t$).

- **Case 1: $\sqrt{4\alpha t} \cdot \text{erfi}^{-1}(DC^{-1}) \geq t$.** In this case, note that $S_{t-1} + 1 \leq t$ and $\alpha \geq 2$,

$$\begin{aligned}
|x_t - x_{t+1}| &\leq \frac{2C}{\sqrt{\alpha t}} \exp\left(\frac{(S_{t-1}+2)^2}{4\alpha t}\right) \\
&= \frac{2C}{\sqrt{\alpha t}} \exp\left(\frac{S_{t-1}^2}{4\alpha t}\right) \exp\left(\frac{4S_{t-1}+4}{4\alpha t}\right) \leq \frac{2\sqrt{e}C}{\sqrt{\alpha t}} \exp\left(\frac{S_{t-1}^2}{4\alpha t}\right).
\end{aligned}$$

  Since $\sqrt{4\alpha t} \cdot \text{erfi}^{-1}(DC^{-1}) \geq t$, we have $t \leq 4\alpha(\text{erfi}^{-1}(DC^{-1}))^2$. Therefore,

$$\exp\left(\frac{S_{t-1}^2}{4\alpha t}\right) \leq \exp\left(\frac{t}{4\alpha}\right) \leq \exp\left[\left(\text{erfi}^{-1}(DC^{-1})\right)^2\right],$$

$$|x_t - x_{t+1}| \leq \frac{2\sqrt{e}C}{\sqrt{\alpha t}} \exp\left[\left(\text{erfi}^{-1}(DC^{-1})\right)^2\right].$$

- **Case 2: $\sqrt{4\alpha t} \cdot \text{erfi}^{-1}(DC^{-1}) < t$.** Plugging the $O(\sqrt{t})$ bound for S_{t-1} into

$|x_t - x_{t+1}|$,

$$\begin{aligned}|x_t - x_{t+1}| &\leq \frac{2C}{\sqrt{\alpha t}} \exp\left(\frac{(\sqrt{4\alpha(t-1)} \cdot \text{erfi}^{-1}(DC^{-1}) + 3)^2}{4\alpha t}\right) \\ &\leq \frac{2C}{\sqrt{\alpha t}} \exp\left[\left(\text{erfi}^{-1}(DC^{-1})\right)^2\right] \exp\left(\frac{6t+9}{4\alpha t}\right) \\ &\leq \frac{2e^2 C}{\sqrt{\alpha t}} \exp\left[\left(\text{erfi}^{-1}(DC^{-1})\right)^2\right].\end{aligned}$$

Combining the above, we have

$$|x_t - x_{t+1}| \leq \frac{2e^2 C}{\sqrt{\alpha t}} \exp\left[\left(\text{erfi}^{-1}(DC^{-1})\right)^2\right].$$

Step 4: upper-bound $\exp[(\text{erfi}^{-1}(x))^2]$**.** Let us consider the related $\int_0^x \text{erfi}(u)du$. Using integration by parts,

$$\begin{aligned}\int_0^x \text{erfi}(u)du &= u \cdot \text{erfi}(u)\Big|_{u=0}^{x} - \int_0^x u \exp(u^2)du \\ &= x \cdot \text{erfi}(x) - \frac{1}{2}\exp(x^2) + \frac{1}{2}.\end{aligned}$$

Plugging in $x = \text{erfi}^{-1}(DC^{-1})$, we have

$$\begin{aligned}\exp\left[\left(\text{erfi}^{-1}(DC^{-1})\right)^2\right] &= 2DC^{-1} \cdot \text{erfi}^{-1}(DC^{-1}) + 1 - 2\int_0^{\text{erfi}^{-1}(DC^{-1})} \text{erfi}(u)du \\ &\leq 2DC^{-1} \cdot \text{erfi}^{-1}(DC^{-1}) + 1.\end{aligned}$$

Then, as we did in Theorem 3.2, we plug in $\text{erfi}^{-1}(x) \leq 1 + \sqrt{\log(1+x)}$ and obtain

$$|x_t - x_{t+1}| \leq \frac{11}{\sqrt{t}}\left\{2D\left[1 + \sqrt{\log(1 + DC^{-1})}\right] + C\right\}.$$

Step 5: final bits. Note that

$$\sum_{t=T_1}^{T_2-1} \frac{1}{\sqrt{t}} \leq \int_{T_1-1}^{T_2-1} \frac{1}{\sqrt{x}}dx \leq 2\sqrt{T_2 - 1} - 2\sqrt{T_1 - 1} \leq 2\sqrt{T_2 - T_1}.$$

Combining it with our bound on $|x_t - x_{t+1}|$ completes the proof. □

3.8.3 The expert problem

Theorem 3.5 The proof relies on an auxiliary lemma (Lemma 3.10), which is presented at the end.

Proof of Theorem 3.5. We divide the proof into three steps.

Step 1 The first step is to show that (i) for all $u \in \Delta(d)$, $\langle g_t, x_t - u\rangle \le \langle z_t, w_t - u\rangle$; and (ii) $\|x_t - x_{t+1}\|_1 \le O(\|w_t - w_{t+1}\|_1)$. In this way, we can translate the LEA problem to a d-dimensional OLO problem with the loss vector z_t, despite not achieving the root KL bound yet.

To prove the goal (i), we consider two cases, $\|w_t\|_1 < 1$ and $\|w_t\|_1 > 1$ (the case of $\|w_t\|_1 = 1$ trivially holds). If $\|w_t\|_1 < 1$, we have $x_t = w_t + d^{-1}(1 - \|w_t\|_1)$ and $z_t = g_t - \|g_t\|_\infty$.

$$\langle g_t, x_t - u\rangle = \langle g_t, w_t - u\rangle + (1 - \|w_t\|_1)\left(\sum_i g_{t,i}/d\right),$$

$$\langle z_t, w_t - u\rangle = \langle g_t, w_t - u\rangle + (1 - \|w_t\|_1)\|g_t\|_\infty,$$

therefore $\langle g_t, x_t - u\rangle \le \langle z_t, w_t - u\rangle$. As for the case of $\|w_t\|_1 > 1$, similarly, $x_t = w_t/\|w_t\|_1$, $z_t = g_t + \|g_t\|_\infty$, $\langle g_t, x_t - u\rangle = \langle g_t, w_t/\|w_t\|_1 - u\rangle$, and $\langle z_t, w_t - u\rangle = \langle g_t, w_t - u\rangle + \|g_t\|_\infty(\|w_t\|_1 - 1)$.

$$\langle g_t, x_t - u\rangle - \langle z_t, w_t - u\rangle = -(\langle g_t, x_t\rangle + \|g_t\|_\infty)(\|w_t\|_1 - 1) \le 0.$$

Now consider the goal (ii). To avoid cluttered notations, define $a_t = w_t + d^{-1}\max\{0, 1 - \|w_t\|_1\}$ and $A_t = \max\{\|w_t\|_1, 1\}$. Note that $A_t = \|a_t\|_1$.

$$\begin{aligned}
\|x_t - x_{t+1}\|_1 &= \left\|\frac{a_t}{A_t} - \frac{a_{t+1}}{A_{t+1}}\right\|_1 \\
&= \left\|\frac{(a_t - a_{t+1})A_{t+1} + a_{t+1}(A_{t+1} - A_t)}{A_t A_{t+1}}\right\|_1 \\
&\le \frac{1}{A_t}\|a_t - a_{t+1}\|_1 + \frac{1}{A_t}(A_{t+1} - A_t) \le 2\|a_t - a_{t+1}\|_1.
\end{aligned}$$

$$
\begin{aligned}
\|a_t - a_{t+1}\|_1 &= \left\| w_t + d^{-1}\max\{0, 1 - \|w_t\|_1\} - w_{t+1} - d^{-1}\max\{0, 1 - \|w_{t+1}\|_1\} \right\|_1 \\
&\le \|w_t - w_{t+1}\|_1 + |\max\{0, 1 - \|w_t\|_1\} - \max\{0, 1 - \|w_{t+1}\|_1\}| \\
&\le \|w_t - w_{t+1}\|_1 + |\|w_t\|_1 - \|w_{t+1}\|_1| \le 2\|w_t - w_{t+1}\|_1 .
\end{aligned}
$$

Therefore, $\|x_t - x_{t+1}\|_1 \le 4\|w_t - w_{t+1}\|_1$.

Step 2 The second step is to add up the regret bound for each coordinates. Consider the i-th coordinate. Note that $|z_{t,i}| \le 2G$. Using Theorem 3.3, for all $u_{1d} \in \mathbb{R}_+$,

$$
\begin{aligned}
& \sum_{t=1}^{T} z_{t,i}(w_{t,i} - u_{1d}) + \tilde{\lambda}\sum_{t=1}^{T-1} |w_{t,i} - w_{t+1,i}| \\
\le\ & \sqrt{(4\tilde{\lambda}\tilde{G} + 2\tilde{G}^2)T}\left[\pi_i + |u_{1d} - \pi_i|\left(\sqrt{4\log\left(1 + \frac{|u_{1d} - \pi_i|}{\pi_i}\right)} + 2\right)\right] \\
=\ & \sqrt{(32\lambda G + 8G^2)T}\left[\pi_i + |u_{1d} - \pi_i|\left(\sqrt{4\log\left(1 + \frac{|u_{1d} - \pi_i|}{\pi_i}\right)} + 2\right)\right].
\end{aligned}
$$

Then, by summing up all the coordinates, for all $u \in \Delta(d)$,

$$
\begin{aligned}
& \sum_{t=1}^{T} \langle g_t, x_t - u\rangle + \lambda\sum_{t=1}^{T-1}\|x_t - x_{t+1}\|_1 \\
\le\ & \sum_{t=1}^{T} \langle z_t, w_t - u\rangle + 4\lambda\sum_{t=1}^{T-1}\|w_t - w_{t+1}\|_1 \\
=\ & \sum_{i=1}^{d}\left[\sum_{t=1}^{T} z_{t,i}(w_{t,i} - u_i) + \tilde{\lambda}\sum_{t=1}^{T-1}|w_{t,i} - w_{t+1,i}|\right] \\
\le\ & \sqrt{(32\lambda G + 8G^2)T}\left[1 + 2\|u - \pi\|_1 + 2\sum_{i=1}^{d}|u_i - \pi_i|\sqrt{\log\left(1 + \frac{|u_i - \pi_i|}{\pi_i}\right)}\right] \\
\le\ & \sqrt{(32\lambda G + 8G^2)T} \\
& \qquad \times\left[1 + 2\|u - \pi\|_1 + 2\sqrt{\|u - \pi\|_1}\sqrt{\sum_{i=1}^{d}|u_i - \pi_i|\log\left(1 + \frac{|u_i - \pi_i|}{\pi_i}\right)}\right].
\end{aligned}
$$

(Cauchy-Schwarz)

Observe that since u and π both belong to $\Delta(d)$, $\|u-\pi\|_1 \le 2$. If we define a function f as

$$f := |x-1|\log(1+|x-1|),$$

then using the standard definition of *f-divergence*

$$D_f(u||\pi) := \sum_{i=1}^{d} \pi_i f\left(\frac{u_i}{\pi_i}\right),$$

we have

$$\mathrm{Regret}_T^{\lambda}(u) = \left[\sqrt{\mathrm{TV}(u||\pi)\cdot D_f(u||\pi)}+1\right]\cdot O\left(\sqrt{(\lambda G+G^2)T}\right).$$

Step 3 The last step is to upper bound $D_f(u||\pi)$ by $\mathrm{KL}(u||\pi)$. To this end, notice that $\mathrm{KL}(u||\pi) = D_g(u||\pi)$, where

$$g(x) := 1-x+x\log x.$$

By Lemma 3.10, $f(x) \le 2g(x)$ for all $x \ge 0$, therefore $D_f(u||\pi) \le 2D_g(u||\pi) = 2\mathrm{KL}(u||\pi)$. □

Lemma 3.10. *For all $x \ge 0$,*

$$|x-1|\log(1+|x-1|) \le 2(1-x+x\log x).$$

Proof of Lemma 3.10. Define LHS − RHS as $h(x)$. Clearly, $h(1)=0$. When $x>1$,

$$h'(x) = 1-\log x - x^{-1}.$$

It equals 0 when $x=1$, and $h''(x) = (1-x)/x^2$ which is negative for all $x>1$. Therefore, $h(x)\le 0$ for all $x \ge 1$.

As for the case of $x<1$,

$$h'(x) = -\log(2-x) - \frac{1-x}{2-x} - 2\log x,$$

$$h''(x) = -\frac{x^2-x+2}{(x-2)^2 x} < 0,$$

therefore $h(x)\le 0$ for all $0 \le x \le 1$. □

Regarding Theorem 3.5, to justify the improvement of $\sqrt{\mathrm{TV} \cdot \mathrm{KL}}$ over $\sqrt{\mathrm{KL}}$, here is an example.

Example 3.1. *For all $d \geq 3$, define $p, q \in \Delta(d)$ from*

$$p_1 = \frac{1}{\sqrt{\log d}}, \quad q_1 = \frac{1}{d\sqrt{\log d}},$$

and

$$p_i = \frac{1 - p_1}{d - 1}, \quad q_i = \frac{1 - q_1}{d - 1}, \quad \forall i \in [2 : d].$$

Then,

$$\mathrm{TV}(p||q) = \frac{1}{2}\left[|p_1 - q_1| + (d-1)\left|\frac{1-p_1}{d-1} - \frac{1-q_1}{d-1}\right|\right] = |p_1 - q_1| = \frac{d-1}{d\sqrt{\log d}},$$

$$\begin{aligned}
\mathrm{KL}(p||q) &= p_1 \log \frac{p_1}{q_1} + (d-1) \cdot \frac{1-p_1}{d-1} \log \frac{1-p_1}{1-q_1} \\
&= \sqrt{\log d} + \left(1 - \frac{1}{\sqrt{\log d}}\right) \log \left(1 - \frac{d-1}{d\sqrt{\log d} - 1}\right) \\
&\geq \sqrt{\log d} + \log \left(1 - \frac{d}{d\sqrt{\log d} - 1}\right) = \sqrt{\log d} - o(1).
\end{aligned}$$

Since we also have

$$\mathrm{KL}(p||q) = \sqrt{\log d} + (1 - p_1) \log \frac{1 - p_1}{1 - q_1} \leq \sqrt{d},$$

we can combine the above and obtain $\mathrm{TV}(p||q) \cdot \mathrm{KL}(p||q) \leq 1$ and $\mathrm{KL}(p||q) \geq \sqrt{\log d} - o(1)$. If our comparator u and prior π take the value of p and q respectively, then even without switching costs, Theorem 3.5 saves a $(\log d)^{1/4}$ factor from the existing comparator adaptive bounds, e.g., (Orabona, 2019, Section 9.6).

Part II

Temporal Representation

Chapter 4

Unconstrained Dynamic Regret

Moving to the second part of the dissertation, we shift our attention from temporal continuity to temporal representation. The overarching idea is that the use of time-dependent features can bring some surprising benefits to adaptive online learning. This is based on (Zhang et al., 2023).

Concretely, we will study the dynamic regret (1.1) in OCO, rather than the static regret considered in previous chapters. Furthermore, the domain $\mathcal{X}$ is unconstrained, i.e., $\mathcal{X} = \mathbb{R}^d$. This will unify two types of adaptivity discussed in Section 1.4 (Type 1 and Type 3), under a generalized notion of comparator adaptivity.

Section 4.1 motivates the problem and summarizes our contributions. Section 4.2 surveys existing results. Section 4.3 introduces the general algorithmic framework achieving a sparsity adaptive dynamic regret bound. Section 4.4 specialized this general framework using the Haar wavelet features, resulting in quantitative improvements over the state of the art. An application to fine-tuning time series forecasters, including experiments, are presented in Section 4.5. Section 4.6 concludes this chapter and discusses future directions. All the proofs are deferred to Section 4.7.

Setting The setting of this chapter is in some sense the most challenging one considered this dissertation. We study the OCO problem (Definition 1) with the domain $\mathcal{X} = \mathbb{R}^d$, and the loss functions are G-Lipschitz with respect to the Euclidean norm. The time horizon T is in general unknown unless specified otherwise. The

performance metric is the dynamic regret (1.1), copied as

$$\text{Regret}_T(Env, u_{1:T}) := \sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u_t).$$

Our goal is to upperbound its supremum over Env, i.e.,

$$\text{Regret}_T(u_{1:T}) := \sup_{Env} \left[\sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u_t) \right], \tag{4.1}$$

by a function of $u_{1:T}$. Specifically, there are no restrictions on the comparator at all: for all t, $u_t \in \mathbb{R}^d$. Therefore, this objective is called the *unconstrained dynamic regret*.

Notation Our results will depend on suitable statistics to quantify the regularity of a comparator sequence $u_{1:T}$. Several such statistics are defined throughout this chapter, which are summarized in Table 4.1. This is mainly for the purpose of reference. These statistics will be motivated and introduced carefully as we proceed.

Name	Notation	Definition
Maximum range	M	$\max_t \|u_t\|$
Comparator average	$\bar{u}$	$\frac{1}{T}\sum_{t=1}^{T} u_t$
Path length	P	$\sum_{t=1}^{T-1} \|u_{t+1} - u_t\|_2$
Norm sum	S	$\sum_{t=1}^{T} \|u_t\|_2$
First order variability	$\bar{S}$	$\sum_{t=1}^{T} \|u_t - \bar{u}\|_2$
Energy	E	$\sum_{t=1}^{T} \|u_t\|_2^2$
Second order variability	$\bar{E}$	$\sum_{t=1}^{T} \|u_t - \bar{u}\|_2^2$
Number of switches	K	$\sum_{t=1}^{T-1} \mathbf{1}[u_{t+1} \neq u_t]$
Sparsity on a dictionary $\mathcal{H}$	$\text{Sparsity}_{\mathcal{H}}$	$\frac{(\sum_{n=1}^{N} \|z^{(n)}\|_2)^2}{\sum_{n=1}^{N} \|z^{(n)}\|_2^2}$

Table 4.1: List of comparator statistics.

Furthermore, the types of norms will be important throughout our analysis. There-

fore, unlike Chapter 2, we will write the Euclidean norm as $\|\cdot\|_2$, rather than the suppressed notation $\|\cdot\|$.

4.1 Motivation and contribution

Our setting above deviates from the most standard setting of OCO in two ways: the domain $\mathbb{R}^d$ is unbounded, and the comparator is allowed to be time-varying. Taking a closer look at their analysis, one could see that these two problem structures actually share a common theme, despite being studied mostly separately. In either the unconstrained static setting (McMahan and Orabona, 2014; Orabona and Pál, 2016; Cutkosky and Orabona, 2018) or the bounded dynamic setting (Zinkevich, 2003; Hall and Willett, 2015; Zhang et al., 2018a), the standard form of minimax optimality (Abernethy et al., 2008a, 2009; Rakhlin and Sridharan, 2014b) becomes vacuous, as it is impossible to guarantee that $\sup_{u_{1:T}} \text{Regret}_T(u_{1:T})$ is sublinear in T. Circumventing this issue in either cases relies on comparator adaptivity[1] – instead of only depending on T, any appropriate regret upper bound, denoted by $\text{Bound}_T(u_{1:T})$, should also depend on the comparator $u_{1:T}$ through a certain complexity measure. Intuitively, despite the intractability of hard comparators, nonvacuous bounds can be established against "easy ones". A total loss bound then follows from the oracle inequality

$$\sum_{t=1}^{T} l_t(x_t) \leq \inf_{u_{1:T}} \left[\sum_{t=1}^{T} l_t(u_t) + \text{Bound}_T(u_{1:T}) \right]. \tag{4.2}$$

A crucial observation is that the complexity of $u_{1:T}$ is not uniquely defined: one could imagine bounding $\text{Regret}_T(u_{1:T})$ by many different non-comparable functions of $u_{1:T}$. Essentially, this complexity measure serves as a Bayesian prior (Section 1.3): choosing it amounts to assigning different priorities to different comparators $u_{1:T} \in \mathbb{R}^{d \times T}$. The associated algorithm guarantees lower $\text{Bound}_T(u_{1:T})$ against comparators

[1] Here it should be interpreted as a more general, dynamic version of the Type 1 discussed in Section 1.4.

with higher priority, and due to (4.2), the total loss of our algorithm is low if some of these high priority comparators actually achieve low loss $\sum_{t=1}^{T} l_t(u_t)$. Such a Bayesian reasoning highlights the importance of versatility in this workflow: in order to place an arbitrary application-dependent prior, we need a versatile algorithmic framework that adapts to a wide range of complexity measures. This leads to the limitations of existing results, discussed next.

To our knowledge, (Jacobsen and Cutkosky, 2022) is the only existing work that considers our setting. Two unconstrained dynamic regret bounds are presented based on three statistics of the comparator sequence, the *maximum range* $M := \max_t \|u_t\|_2$, the *norm sum* $S := \sum_{t=1}^{T} \|u_t\|_2$ and the *path length* $P := \sum_{t=1}^{T-1} \|u_{t+1} - u_t\|_2$. First, using a 1D unconstrained static algorithm as a simple range scaler, the paper achieves (Jacobsen and Cutkosky, 2022, Lemma 10)

$$\mathrm{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\sqrt{(M+P)MT}\right), \tag{4.3}$$

Then, by developing a customized mirror descent approach, most of the effort is devoted to improving MT to S (Jacobsen and Cutkosky, 2022, Theorem 4), i.e., adapting to the magnitude of individual u_t.

$$\mathrm{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\sqrt{(M+P)S}\right). \tag{4.4}$$

Despite the strengths of these results and their nontrivial analysis, a shared limitation is that both bounds depend explicitly on the path length P. Intuitively, it means that good performance is only guaranteed in almost static environments: in the typical situation of $S = \Theta(T)$, these bounds are only sublinear when $P = o(T)$, which rules out important persistent dynamics such as periodicity. Moreover, even the second bound still depends on $\sqrt{MS}$ instead of a finer characterization of each individual

u_t's magnitude. That is, the mission of improving M is not fully accomplished yet.[2]

The goal of this chapter is to extend comparator adaptivity to a wider range of complexity measures. For almost static environments in particular, quantitative benefits will be obtained from specific instances of this general approach.

Result and contribution Our contributions are twofold.

1. First, we present an algorithmic framework achieving a new type of unconstrained dynamic regret bounds. It is based on a conversion to vector-output *Online Linear Regression* (OLR): given a dictionary $\mathcal{H}$ of orthogonal feature vectors in the *sequence space* $\mathbb{R}^{dT}$, we use an unconstrained static OCO algorithm to linearly aggregate these feature vectors, which are themselves time-varying prediction sequences. Such a procedure guarantees

$$\mathrm{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\sqrt{E \cdot \mathrm{Sparsity}_{\mathcal{H}}}\right), \tag{4.5}$$

   where $E = \sum_{t=1}^{T} \|u_t\|_2^2$ is the *energy* of the comparator $u_{1:T}$, and $\mathrm{Sparsity}_{\mathcal{H}}$ measures the sparsity of $u_{1:T}$ on the dictionary $\mathcal{H}$.[3] Both E and $\mathrm{Sparsity}_{\mathcal{H}}$ are unknown beforehand.

   Compared to (Jacobsen and Cutkosky, 2022), the main advantage of this framework is its versatility. Prior knowledge on the transform domain can be incorporated by picking $\mathcal{H}$, and favorable algorithmic properties can be conveniently inherited from static online learning.

[2]The significance of this issue could be seen through an analogy to (static D-bounded domain) gradient adaptive OCO: although there are algorithms achieving the already adaptive $O\left(D\sqrt{G\sum_{t=1}^{T}\|g_t\|_2}\right)$ static regret bound, the hallmark of gradient adaptivity is the so-called "second-order bound" $O\left(D\sqrt{\sum_{t=1}^{T}\|g_t\|_2^2}\right)$, popularized by AdaGrad Duchi et al. (2011). In a rough but related sense, we aim to achieve "second order comparator adaptivity", which is only manifested in the less studied dynamic setting.

[3]For conciseness, we omit $u_{1:T}$ in the notation. Throughout this chapter, the regularity parameters on the RHS of the regret bound generally depend on $u_{1:T}$.

Algorithm	P-dependent bound	K-switching regret	Example 4.1	Example 4.2
ADER (Zhang et al., 2018a)	$\tilde{O}\left(\sqrt{(D+P)DT}\right)$	$\tilde{O}\left(D\sqrt{(1+K)T}\right)$	N/A	$\tilde{O}(T^{3/4})$
(Jacobsen and Cutkosky, 2022, Algorithm 6)	$\tilde{O}\left(\sqrt{(M+P)MT}\right)$	$\tilde{O}\left(M\sqrt{(1+K)T}\right)$	$\tilde{O}(T)$	$\tilde{O}(T^{3/4})$
(Jacobsen and Cutkosky, 2022, Algorithm 2)	$\tilde{O}\left(\sqrt{(M+P)S}\right)$	$\tilde{O}\left(\sqrt{(1+K)MS}\right)$	$\tilde{O}(T^{3/4})$	$\tilde{O}(T^{3/4})$
Ours (Haar OLR)	$\tilde{O}\left(\|\bar{u}\|_2\sqrt{T}+\sqrt{P\bar{S}}\right)$	$\tilde{O}\left(\|\bar{u}\|_2\sqrt{T}+\sqrt{K\bar{E}}\right)$	$\tilde{O}(\sqrt{T})$	$\tilde{O}(\sqrt{T})$

Table 4.2: Comparison in almost static environments. Each row improves the previous row (omitting logarithmic factors). The ADER algorithm requires a D-bounded domain, while the other three algorithms are unconstrained. The rates in the two examples refer to the minimum of the P and K dependent bounds.

2. Our second contribution is quantitative: although (Jacobsen and Cutkosky, 2022) is specifically crafted to handle almost static environments, we show that equipped with a *Haar wavelet* dictionary, our framework actually guarantees better bounds (Table 4.2) in this setting, which is a surprising finding to us.

   - With the *comparator average* $\bar{u} := \sum_{t=1}^{T} u_t/T$ and the *first order variability* $\bar{S} := \sum_{t=1}^{T} \|u_t - \bar{u}\|_2$, our Haar wavelet algorithm guarantees

   $$\text{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\|\bar{u}\|_2\sqrt{T}+\sqrt{P\bar{S}}\right).$$

   It improves Eq.(4.4) by (i) a better dependence on the comparator magnitude ($\sqrt{MS} \to \|\bar{u}\|_2\sqrt{T}$); and (ii) decoupling the bias $\bar{u}$ from the characterization of variability ($\sqrt{PS} \to \sqrt{P\bar{S}}$).

   - With the *number of switches* $K := \sum_{t=1}^{T-1} \mathbf{1}[u_{t+1} \neq u_t]$ and the *second order variability* $\bar{E} := \sum_{t=1}^{T} \|u_t - \bar{u}\|_2^2$, the same Haar wavelet algorithm guarantees an anytime *unconstrained switching regret bound*

   $$\text{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\|\bar{u}\|_2\sqrt{T}+\sqrt{K\bar{E}}\right),$$

   which improves the existing $\tilde{O}\left(\sqrt{(1+K)MS}\right)$ bound resulting from Eq.(4.4) and $P = O(KM)$.

Due to the local property of wavelets, our algorithm runs in $O(d \log T)$ time per round, matching that of the baselines. As for the regret, our bounds are never worse than the baselines, and in two examples corresponding to $\|\bar{u}\|_2 \ll M$ and $\bar{S} \ll S$, they reduce to clearly improved rates in T. Furthermore, our analysis follows from the generic regret bound (4.5) and the wavelet approximation theory, providing an intriguing connection between disparate fields.

4.2 Related work

Tackling unconstrained dynamic regret requires addressing the connection between unconstrained OCO and dynamic OCO. Although they both embody the idea of comparator adaptivity, unified studies have been scarce. For related works on unconstrained OCO, the reader is referred to Section 2.2. Here we survey related works on the dynamic aspects of online learning.

Dynamic OCO Comparing against dynamic sequences is a classical research topic. It is clear that one cannot go beyond linear regret in the worst case, therefore various notions of complexity should be introduced.

- The closest topic to ours is the *universal dynamic regret*, where the regret bound adapts to the complexity of an arbitrary $u_{1:T}$ on a bounded domain with L_p-diameter D. In the most common framework, the complexity measure is an $L_{p,q}$ norm of the difference sequence $\{u_{t+1} - u_t\}$, such as the $L_{p,1}$ norm, i.e., the path length $P = \sum_{t=1}^{T-1} \|u_{t+1} - u_t\|_p$ (Herbster and Warmuth, 2001). Omitting the dependence on the dimension d (thus also the choice of p), the optimal bound under convex Lipschitz losses is $\tilde{O}\left(\sqrt{(D+P)DT}\right)$ (Zinkevich, 2003; Hall and Willett, 2015; Jadbabaie et al., 2015; Gyorgy and Szepesvári, 2016; Zhang et al., 2018a), while the accelerated rate[4] $\tilde{O}(P^{2/3}T^{1/3} \vee 1)$ can be achieved with strong

[4]Further omitting the dependence on the diameter D.

convexity (Baby and Wang, 2021, 2022). These bounds subsume results in *switching (or shifting) regret*, as P is dominated by D times the number of switches in $u_{1:T}$.

A notable exception is the *dynamic model* framework from (Hall and Willett, 2015; Zhang et al., 2018a). Still considering a bounded domain, it takes a collection of dynamic models as input, which are mappings from the domain to itself. Then, the complexity of a comparator $u_{1:T}$ is measured by how well it can be reconstructed by the best dynamic model in hindsight. Essentially, the use of temporal representations is similar to the dictionary in our framework. The key difference is that instead of using the best feature (or the best convex combination of the features) to measure the comparator, we use linear combinations of the features – this allows handling unconstrained domains through subspace modeling.

- Besides the universal dynamic regret, there are other notions of dynamic regret that do not induce oracle inequalities like (4.2), including (i) the *restricted dynamic regret* (Yang et al., 2016; Zhang et al., 2017; Baby and Wang, 2019, 2020; Baby et al., 2021), which depends on the complexity of certain *offline optimal* comparators;[5] and (ii) regret bounds that depend on the *functional variation* $\sum_{t=1}^{T-1} \max_x |l_t(x) - l_{t+1}(x)|$ (Besbes et al., 2015; Chen et al., 2019). They are not as relevant to our purpose, due to being vacuous on unbounded domains under the linear losses – this is an important setting in our investigation.

Unconstrained (universal) dynamic regret To our knowledge, (Jacobsen and Cutkosky, 2022) is the only work studying the universal dynamic regret without a

[5]Notably, (Baby and Wang, 2019, 2020) creatively employed wavelet techniques to detect change-points of the environment, which, to the best of our knowledge, is the only existing use of wavelets in the online learning literature.

bounded domain, whose contributions have been summarized in Section 4.1. Here we survey some negative results in the literature, which will be revisited later on.

- The restricted dynamic regret is a special case of the universal dynamic regret, therefore lower bounds for the former apply to the latter as well. For convex Lipschitz losses (Yang et al., 2016) and strongly convex losses (Baby and Wang, 2019), any algorithm should suffer the dynamic regret of $\Omega(P)$ and $\Omega(P^2)$, respectively.

- For dynamic OCO on bounded domains, a recurring analysis goes through the notion of *strong adaptivity* (Daniely et al., 2015), i.e., Type 4 in Section 1.4: one first achieves low static regret bounds on every subinterval of the time horizon $[1:T]$, and then assembles these local bounds appropriately to bound the global dynamic regret (Zhang et al., 2018b; Cutkosky, 2020; Baby and Wang, 2021, 2022). Following this route in the unconstrained setting appears to be challenging, as (Jacobsen and Cutkosky, 2022, Section 4) showed that (a natural form of) strong adaptivity cannot be achieved there.

Besides these "core" related works, the following topics are also relevant.

Online regression Our sparse coding framework converts unconstrained dynamic OCO to a special form of online regression. The standard setting of the latter (Rakhlin and Sridharan, 2014a) considers a repeated game as well: in each round, we observe a covariate $x_t \in \mathbb{R}^d$, make a prediction $\hat{y}_t \in \mathbb{R}$ (which depends on x_t), and then observe a label $y_t \in \mathbb{R}$. The performance metric is the minimax regret under the square loss

$$\mathrm{Regret}_T(\mathcal{F}) = \sum_{t=1}^{T} (\hat{y}_t - y_t)^2 - \inf_{f \in \mathcal{F}} \sum_{t=1}^{T} (f(x_t) - y_t)^2.$$

Roughly, the problem is of a nonparametric type if the complexity of the function class $\mathcal{F}$ is not fixed a priori, but grows with T (i.e., the amount of data).

Overall, such an online regression problem is highly general, as static OCO is recovered if x_t is time-invariant. The setting we utilize is a variant with (i) vector output; (ii) general convex losses; (iii) x_t specified by the dictionary, possibly being sparse itself (e.g., wavelets); and (iv) the function class $\mathcal{F}$ being linear, but unbounded.

Existing works on online nonparametric regression (Rakhlin and Sridharan, 2014a; Gaillard and Gerchinovitz, 2015) have established the relation of this problem to certain path length characterizations of dynamic regret. However, the generality of this setting makes the analysis challenging, and especially, algorithms can be computationally expensive. With a bounded domain assumption (on predictions $\hat{y}_t$), a recent breakthrough (Baby and Wang, 2021) simultaneously achieved several notions of optimality for path-length-dependent bounds, with efficient computation. Readers are referred to (Baby and Wang, 2021, Appendix A) for a thorough discussion of this line of works.

For the special case of OLR with square losses, the celebrated VAW forecaster (Azoury and Warmuth, 2001; Vovk, 2001) guarantees $O(N \log T)$ regret against any unbounded coefficient vector $\hat{u} \in \mathbb{R}^N$, where N is the dimension of the feature space. Such a fast rate becomes vacuous in the nonparametric regime (when $N > T$) (Gerchinovitz and Yu, 2014), therefore Gerchinovitz (2013) proposed a *sparsity regret bound* $\tilde{O}(\|\hat{u}\|_0)$ and an accompanying inefficient algorithm as its high dimensional generalization. Efficient computation was addressed by (Gaillard and Wintenberger, 2018), but the obtained result only applies to bounded $\hat{u}$. In a rough sense, such sparsity regret bounds are the square loss analogues of the L_1-norm parameter-free bounds in OLO (Orabona, 2019, Chapter 9). They are also closely related to *sparsity oracle inequalities* in statistics, as reviewed by (Gerchinovitz, 2013).

Parametric time series models For time series forecasting, most prior works are devoted to parametric strategies with strong inductive bias, such as the ARMA model,

state space models, and more recent deep learning models. Online learning has been applied to such models as well (Anava et al., 2013, 2015b; Anava and Mannor, 2016; Kuznetsov and Mohri, 2016; Hazan et al., 2018), leading to forecasting guarantees under mild statistical assumptions. When convexity is present, some of these problems could be reframed as special cases of our OLR problem, with a constant-size dictionary that does not grow with T; for example, learning the *autoregressive* model corresponds to defining the features as the fixed-length observation history. Also, Section 4.5 shows that given a parametric time series forecaster (possibly without performance guarantees), our algorithm can be applied on top of it, in order to provably correct its nonstationary bias.

Other sparsity topics in OL Finally, we review other sparsity-related topics in online learning, which do not fit into the scope of this chapter. (Langford et al., 2009; Xiao, 2009; Duchi et al., 2010; Shalev-Shwartz and Tewari, 2011) considered using online learning to solve batch L_1 regularized problems. The goal is to achieve sparse predictions instead of sparsity adaptive regret bounds. (Kale, 2014; Foster et al., 2016; Kale et al., 2017) studied *online sparse regression*, where only a subset of features are available in each round. The challenge is to handle bandit feedback in OLR.

4.3 General sparse coding framework

This section presents our sparse coding framework, achieving the generic sparsity adaptive regret bound (4.5). The key idea is to view online learning on the sequence space $\mathbb{R}^{dT}$, rather than the default domain $\mathbb{R}^d$. Despite its central role in signal processing (e.g., the Fourier transform), such a view is (in our opinion) under-explored by the online learning community.[6] Along this line, Section 4.3.1 converts our setting

[6]Possibly due to the emphasis on the static regret: the sequence $u_{1:T}$ collapses into a time-invariant u, which is contained in $\mathbb{R}^d$.

into a variant of OLR. Our main result and related discussions are presented in Section 4.3.2.

4.3.1 Setting

To begin with, we adopt the conversion from OCO to OLO from Section 1.1. Then, consider the length T sequences of predictions $x_{1:T}$, gradients $g_{1:T}$ and comparators $u_{1:T}$. Let us flatten everything and treat them as dT dimensional vectors, concatenating per-round quantities in $\mathbb{R}^d$. These are called *signals*. The comparator statistics could be more succinctly represented using vector notations, e.g., the energy $E = \sum_{t=1}^{T} \|u_t\|_2^2 = \|u_{1:T}\|_2^2$.

Our framework requires a *dictionary* matrix $\mathcal{H} \in \mathbb{R}^{dT \times N}$, possibly revealed online, whose columns are N nonzero *feature vectors*. We write $\mathcal{H}$ in an equivalent block form as $[h_{t,n}]_{1 \le t \le T, 1 \le n \le N}$, where each block $h_{t,n} \in \mathbb{R}^{d \times 1}$. The accompanied linear transform $u = \mathcal{H}\hat{u}$ relates a signal $u \in \mathbb{R}^{dT}$ to a coefficient vector $\hat{u} \in \mathbb{R}^N$ (if it exists). Adopting the convention in signal processing, we will call $\mathbb{R}^{dT}$ the *time domain*, and $\mathbb{R}^N$ the *transform domain*. In general, symbols without hat refer to time domain quantities, while their transform domain counterparts are denoted with hat.

Summarizing the above, we consider the following concise interaction protocol.[7] Despite its parametric appearance, our main focus is on the nonparametric regime, where the dictionary size N scales with the amount of data T.

Vector-output OLR with linear losses In the t-th round, our algorithm observes a d-by-N feature matrix $\mathcal{H}_t := [h_{t,n}]_{1 \le n \le N}$, linearly combines its columns into a prediction $x_t \in \mathbb{R}^d$, receives a loss gradient $g_t \in \mathbb{R}^d$, and then suffers the linear

[7] Rigorously, this setting is not exactly "regression", since the loss function is the relaxed linear loss. A more accurate name could be "Online Linear Decision". We use "OLR" as it is arguably easier to interpret.

loss $\langle g_t, x_t \rangle$. We assume that[8] $\|h_{t,n}\|_2 \leq 1$, $\sum_{t=1}^{T} \|h_{t,n}\|_2^2 \geq 1$ and $\|g_t\|_2 \leq G$. The performance metric is the unconstrained dynamic regret defined in (4.1).

4.3.2 Main result

In a nutshell, our strategy is to apply an unconstrained static OLO algorithm on the transform domain, and in a coordinate-wise fashion. This is remarkably simple, but also contains a few twists. To make it concrete, let us start with a single feature vector.

Size 1 dictionary Consider an index $n \in [1 : N]$, which is associated to the feature $h_{1:T,n} := [h_{1,n}, \ldots, h_{T,n}] \in \mathbb{R}^{dT}$. We suppress the index n and write it as $h_{1:T} = [h_1, \ldots, h_T]$. For any comparator $u_{1:T} \in \mathrm{span}(h_{1:T})$, there exists $\hat{u} \in \mathbb{R}$ such that $u_{1:T} = h_{1:T}\hat{u}$. The cumulative loss of $u_{1:T}$ can be rewritten as

$$\langle g_{1:T}, u_{1:T} \rangle = \langle g_{1:T}, h_{1:T} \rangle \, \hat{u} = \sum_{t=1}^{T} \langle g_t, h_t \rangle \, \hat{u},$$

which is the loss of the coefficient $\hat{u}$ in a 1D OLO problem with surrogate loss gradients $\langle g_t, h_t \rangle$. Essentially, to compete with a one dimensional comparator subspace $\mathrm{span}(h_{1:T})$, it suffices to run a 1D static regret algorithm that competes with $\hat{u} \in \mathbb{R}$. Such a procedure is presented as Algorithm 4.1.

It remains to choose the static algorithm $\mathcal{A}$. Our results from Chapter 2 can be applied, but to demonstrate its versatility a bit more, we adopt the FreeGrad algorithm (Mhammedi and Koolen, 2020) as an example, which simultaneously achieves static comparator adaptivity and second order gradient adaptivity (Duchi et al., 2011), i.e., Type 2 in Section 1.4. Its pseudocode and static regret bound are presented as follows. The algorithm enjoys another favorable property called *scale-freeness*: the

[8]The assumptions on the features are mild: an important special case is $\max_t \|h_{t,n}\|_2 = 1$, as in the Haar wavelet dictionary. We impose these assumptions to apply unconstrained static algorithms verbatim.

Algorithm 4.1 Sparse coding with size 1 dictionary.

Require: An algorithm $\mathcal{A}$ for static 1D unconstrained OLO with G-Lipschitz losses; and a nonzero feature vector $h_{1:T} \subset \mathbb{R}^{dT}$.

1: **for** $t = 1, 2, \ldots,$ **do**
2: Receive $h_t \in \mathbb{R}^d$.
3: If h_t is nonzero, query $\mathcal{A}$ for its output, and assign it to $\hat{x}_t \in \mathbb{R}$; otherwise, $\hat{x}_t$ is arbitrary.
4: Predict $x_t = \hat{x}_t h_t \in \mathbb{R}^d$, and receive the loss gradient $g_t \in \mathbb{R}^d$.
5: If h_t is nonzero, compute $\hat{g}_t = \langle g_t, h_t \rangle$ and send it to $\mathcal{A}$ as its surrogate loss gradient.
6: **end for**

predictions are invariant to any positive scaling of the loss gradients and the Lipschitz constant G.

Algorithm 4.2 FreeGrad (Mhammedi and Koolen, 2020, Definition 4): scale-free and gradient adaptive unconstrained static OLO.

Require: A hyperparameter $\varepsilon > 0$; dimension d; Lipschitz constant $\hat{G}$.

1: Initialize a gradient sum counter $s = 0 \in \mathbb{R}^d$ and a variance counter $v = \hat{G}^2$.
2: **for** $t = 1, 2, \ldots$ **do**
3: Predict

$$\hat{x}_t = -\varepsilon s \cdot \frac{(2v + \hat{G} \|s\|_2)\hat{G}^2}{2(v + \hat{G} \|s\|_2)^2 \sqrt{v}} \cdot \exp\left(\frac{\|s\|_2^2}{2v + 2\hat{G} \|s\|_2}\right).$$

4: Observe the loss gradient $\hat{g}_t$.
5: Update $s \leftarrow s + \hat{g}_t$, and $v \leftarrow v + \hat{g}_t^2$.
6: **end for**

Lemma 4.1 (Theorem 20 of (Mhammedi and Koolen, 2020))**.** *With any hyperparameter $\varepsilon > 0$, for all $T \in \mathbb{N}_+$ and $\hat{u} \in \mathbb{R}$, Algorithm 4.2 guarantees*

$$\sum_{t=1}^{T} \langle \hat{g}_t, \hat{x}_t - \hat{u} \rangle \leq \varepsilon \hat{G} + \left[2 \|\hat{u}\|_2 \sqrt{V_T \log_+\left(\frac{2 \|\hat{u}\|_2 V_T}{\varepsilon \hat{G}^2}\right)}\right] \vee \left[4 \|\hat{u}\|_2 \hat{G} \log\left(\frac{4 \|\hat{u}\|_2 \sqrt{V_T}}{\varepsilon \hat{G}}\right)\right],$$

where

$$V_T = \hat{G}^2 + \sum_{t=1}^{T} \|\hat{g}_t\|_2^2 .$$

In combination, Algorithm 4.1 has the following guarantee.

Lemma 4.2. *Let $\varepsilon > 0$ be an arbitrary hyperparameter for Algorithm 4.2. Applying its 1D version as the static subroutine, for all $T \in \mathbb{N}_+$ and $u_{1:T} \in \operatorname{span}(h_{1:T})$, against any adversary $\mathcal{E}$, Algorithm 4.1 guarantees*

$$\sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u_t) \leq \varepsilon G + \left(G \frac{\|u_{1:T}\|_2}{\|h_{1:T}\|_2} + \sqrt{\sum_{t=1}^{T} \langle g_t, u_t \rangle^2} \right) \cdot \operatorname{polylog}\left(\max_t \|u_t\|_2 , T, \varepsilon^{-1} \right).$$

General dictionary With the one dimensional learner above, let us turn to the general setting with N features. We run N copies of Algorithm 4.1 in parallel, aggregate their predictions, and the regret bound sums Lemma 4.2, similar to (Cutkosky, 2019b) in the static setting. The pseudocode is presented as Algorithm 4.3, and the regret bound is Theorem 4.1.

Algorithm 4.3 Sparse coding with general dictionary.

Require: A dictionary $\mathcal{H} = [h_{t,n}]$, where $h_{t,n} \in \mathbb{R}^d$; and a hyperparameter $\varepsilon > 0$.

1: For all $n \in [1 : N]$, initialize a copy of Algorithm 4.1 as $\mathcal{A}_n$. It runs the 1D version of Algorithm 4.2 as a subroutine, with hyperparameter ε/N.
2: **for** $t = 1, 2, \ldots,$ **do**
3: Receive $\mathcal{H}_t = [h_{t,n}]_{1 \leq n \leq N}$. For all n, send $h_{t,n}$ to $\mathcal{A}_n$, and query its prediction $w_{t,n}$.
4: Predict $x_t = \sum_{n=1}^{N} w_{t,n}$.
5: Receive loss gradient g_t, and send it to $\mathcal{A}_1, \ldots, \mathcal{A}_N$ as loss gradients.
6: **end for**

Theorem 4.1. *Consider any collection of signals $z^{(n)} \in \operatorname{span}(h_{1:T,n})$, $\forall n$. We define its reconstruction error (for the comparator $u_{1:T}$) as $z^{(0)} = u_{1:T} - \sum_{n=1}^{N} z^{(n)} \in \mathbb{R}^{dT}$. Then,*

for all $T \in \mathbb{N}_+$ and $u_{1:T} \in \mathbb{R}^{dT}$, against any adversary $\mathcal{E}$, Algorithm 4.3 guarantees

$$\sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u_t) \leq -\sum_{t=1}^{T} \left\langle g_t, z_t^{(0)} \right\rangle + \varepsilon G + \left(G \sum_{n=1}^{N} \frac{\left\| z^{(n)} \right\|_2}{\left\| h_{1:T,n} \right\|_2} + \sum_{n=1}^{N} \sqrt{\sum_{t=1}^{T} \left\langle g_t, z_t^{(n)} \right\rangle^2} \right) \cdot \operatorname{polylog}\left(\max_{t,n} \left\| z_t^{(n)} \right\|_2, T, N, \varepsilon^{-1} \right),$$

where $z_t^{(n)} \in \mathbb{R}^d$ is the t-th round component of the sequence $z^{(n)} \in \mathbb{R}^{dT}$.

To interpret this very general result, let us consider a few concrete settings.

- *Static regret.* If the size $N = d$ and the dictionary $\mathcal{H}_t = I_d$, then for any static comparator ($u_t = u \in \mathbb{R}^d$), we can let $z^{(n)}$ be the projection of the sequence $u_{1:T}$ onto $\operatorname{span}(h_{1:T,n})$. This leaves zero reconstruction error, i.e., $u_{1:T} = \sum_{n=1}^{N} z^{(n)}$. Theorem 4.1 reduces to

$$\operatorname{Regret}_T(u_{1:T}) \leq \varepsilon G + \|u\|_1 G\sqrt{T} \cdot \operatorname{polylog}\left(\|u\|_\infty, T, d, \varepsilon^{-1} \right), \tag{4.6}$$

  which recovers a standard $\tilde{O}(\|u\|_1 \sqrt{T})$ bound in coordinate-wise unconstrained static OLO (Orabona, 2019, Section 9.3).

- *Orthogonal dictionary.* Entering the dynamic realm, we now consider the situation where feature vectors are orthogonal (standard in signal processing), and the comparator $u_{1:T} \in \operatorname{span}(\mathcal{H})$. Same as the static setting, we are free to define $z^{(n)}$ as the projection

$$z^{(n)} = \langle h_{1:T,n}, u_{1:T} \rangle \frac{h_{1:T,n}}{\|h_{1:T,n}\|_2^2}.$$

  Due to orthogonality, the projection preserves the energy of the time domain signal, i.e, $E = \|u_{1:T}\|_2^2 = \sum_{n=1}^{N} \left\| z^{(n)} \right\|_2^2$. By further defining $\text{Sparsity}_{\mathcal{H}} := (\sum_{n=1}^{N} \left\| z^{(n)} \right\|_2)^2 / \sum_{n=1}^{N} \left\| z^{(n)} \right\|_2^2$ (arbitrary when the denominator is zero), Theo-

rem 4.1 reduces to

$$\text{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\sqrt{E \cdot \text{Sparsity}_{\mathcal{H}}}\right). \tag{4.7}$$

Note that as the squared L_1/L_2 ratio, $\text{Sparsity}_{\mathcal{H}}$ is a classical sparsity measure (Hurley and Rickard, 2009) of the decomposed signals $\{z^{(n)}\}_{1\leq n\leq N}$: if there are only $N_0 \leq N$ nonzero vectors within this collection, then $\text{Sparsity}_{\mathcal{H}} \leq N_0$ due to the Cauchy-Schwarz inequality. Therefore, the generic sparsity adaptive bound (4.7) depends on (i) the energy of the comparator $u_{1:T}$; and (ii) the sparsity of its representation, without knowing either condition beforehand. The easier the comparator is (low energy, and sparse on $\mathcal{H}$), the lower the bound becomes.

- *Overparameterization.* So far we have only considered $N \leq dT$, where feature vectors can be orthogonal. However, a key idea in signal processing is to use redundant features ($N \gg dT$) to obtain sparser representations. Theorem 4.1 implies a *feature selection* property in this context: since it applies to *any* decomposition of $u_{1:T}$, as long as $u_{1:T}$ can be represented by a subset $\tilde{\mathcal{H}}$ of orthogonal features within $\mathcal{H}$, the regret bound adapts to $\text{Sparsity}_{\tilde{\mathcal{H}}}$, the sparsity of $u_{1:T}$ measured on $\tilde{\mathcal{H}}$. That is, we are theoretically justified to assemble smaller dictionary into a larger one – the regret bound adapts to the quality of the optimal (comparator-dependent) sub-dictionary $\tilde{\mathcal{H}}$.

How to choose the dictionary $\mathcal{H}$? In practice, we may use prior knowledge on the dynamics of the environment. For example, if the environment is periodic, such as the weather or the traffic, then a good choice could be the Fourier dictionary. Similarly, wavelet dictionaries are useful for piecewise regular environments. Another possibility is to learn the dictionary from offline datasets, which is also called *representation learning*. Overall, such prior knowledge is not required to be correct – our algorithm can take any dictionary as input, and the regret bound naturally adapts to its quality.

The established connection between adaptivity and signal structures is a key benefit of our framework.

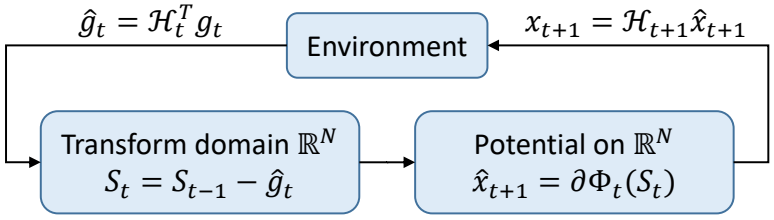


Figure 4·1: Update from the dual space.

A view from the dual space Besides the primal space analysis so far, our algorithm has an equivalent interpretation on the dual space, which leads to a possibly interesting intuition. Typically, in the t-th round, the dual space maintains a summary S_{t-1} of the past observations $g_{1:t-1}$ (called a sufficient statistic), and then passes it through a potential function Φ_t to generate predictions (Cesa-Bianchi and Lugosi, 2006; Foster et al., 2018; Orabona, 2019). While most existing algorithms store the sum of past gradients $\sum_{i=1}^{t-1} g_i$ to handle the static regret, our algorithm stores an N-dimensional transform of the entire sequence $g_{1:t-1}$, illustrated in Figure 4·1. In this way, the *dynamics* of the environment are "memorized".

Power law For a more specific discussion, let us consider an empirically justified setup. In signal processing, the study of sparsity has been partially motivated by the *power law* (Price, 2021): under the standard Fourier or wavelet transforms, the n-th largest transform domain coefficient of many real signals can have magnitude roughly proportional to $n^{-\alpha}$, where $\alpha \in (0.5, 1)$. We also observe this phenomenon from a weather dataset, with details presented in Section 4.5. Figure 4·2 plots the sorted Fourier coefficients of an actual temperature sequence, on a log-log scale. A fitted dashed line is shown in orange, with (negative) slope $\alpha = 0.68$.

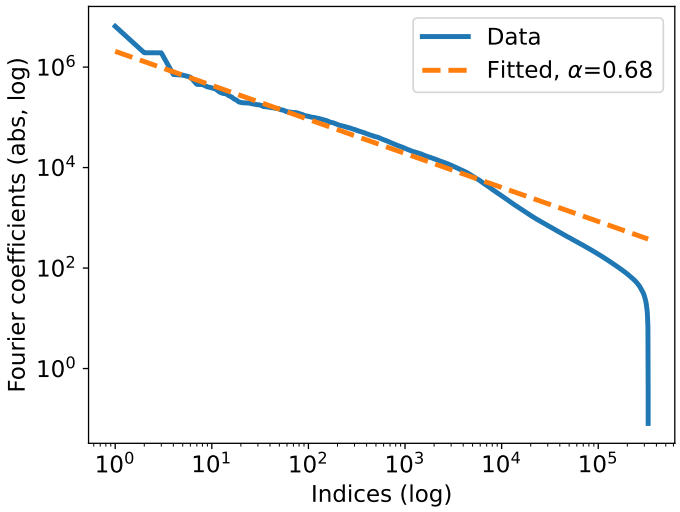


Figure 4·2: An illustration of the power law.

When the power law holds, our bound (4.7) has a more interpretable form. Assuming $d = 1$ and $N = T$,

$$\text{Sparsity}_{\mathcal{H}} = \frac{(\sum_{n=1}^{T} n^{-\alpha})^2}{\sum_{n=1}^{T} n^{-2\alpha}} = O\left(T^{2-2\alpha}\right).$$

In a typical setting of $E = \Theta(T)$, we obtain a sublinear $\tilde{O}(T^{1.5-\alpha})$ dynamic regret bound.

4.4 The Haar OLR algorithm

This section presents our quantitative contributions: despite its generality, our sparse coding framework can improve existing unconstrained dynamic regret bounds (Jacobsen and Cutkosky, 2022). The key workhorse is the ability of wavelet bases to sparsely represent smooth signals. Section 4.4.1 introduces the necessary background, while concrete bounds and proof sketches are presented in Section 4.4.2.

4.4.1 Haar wavelet

Wavelet is a fundamental topic in signal processing, with long lasting impact throughout modern data science. Roughly speaking, the motivation is that a signal can simultaneously exhibit nonstationarity at different time scales, such as slow drifts and fast jumps, therefore to faithfully represent it, we should apply feature vectors with different resolutions. We will only use the simplest Haar wavelets, which is already sufficient. Readers are referred to (Mallat, 2008; Johnstone, 2019) for a thorough introduction to this topic.

Specifically, we start from the 1D setting ($d = 1$) with a dyadic horizon ($T = 2^m$, for some $m \in \mathbb{N}_+$). The Haar wavelet dictionary consists of T (unnormalized) orthogonal feature vectors, indexed by a *scale* parameter $j \in [1 : \log_2 T]$ and a *location* parameter $l \in [1 : 2^{-j}T]$. Given a (j, l) pair, define a feature $h^{(j,l)} = [h_1^{(j,l)}, \ldots, h_T^{(j,l)}] \in \mathbb{R}^T$ entry-wise as

$$h_t^{(j,l)} = \begin{cases} 1, & t \in [2^j(l-1) + 1 : 2^j(l-1) + 2^{j-1}]; \\ -1, & t \in [2^j(l-1) + 2^{j-1} + 1 : 2^j l]; \\ 0, & \text{else.} \end{cases}$$

It means that $h^{(j,l)}$ is only nonzero on a length-2^j interval, while changing its sign once in the middle of this interval. Collecting all the (j, l) pairs yield $T - 1$ features; then, we incorporate an extra all-one feature $h^* = [1, \ldots, 1]$ to complete this size T dictionary.

The defined features can be assembled into the columns of a matrix Haar_m. To help with the intuition, Haar_2 with $T = 4$ is presented in (4.8). The columns from the left to the right are h^*, $h^{(2,1)}$, $h^{(1,1)}$ and $h^{(1,2)}$. Observe that they are orthogonal, and the norm assumption from Section 4.3.1 is satisfied. Therefore, our sparsity adaptive

regret bound (4.7) is applicable.

$$\mathrm{Haar}_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix}. \tag{4.8}$$

Given this 1D Haar wavelet dictionary, we apply a minor variant of Algorithm 4.3 to prevent the dimension d from appearing in the regret bound. When $d = 1$, the algorithm is exactly Algorithm 4.3, where intuitions are most clearly demonstrated. Then, the doubling trick is adopted to relax the knowledge of T. The pseudocode for the single block setting and the doubling trick is presented as Algorithm 4.4 and 4.5.

Algorithm 4.4 Haar OLR with known time horizon.

Require: A time horizon $T = 2^m$; the $T \times T$ Haar dictionary matrix Haar_m; and a hyperparameter $\varepsilon > 0$ (default is 1).

Let $N = T$. For all $n \in [1 : N]$, initialize a copy of the d dimensional version of Algorithm 4.2 as $\mathcal{A}_n$, with hyperparameter ε/N.
for $t = 1, 2, \ldots,$ **do**
  Receive the t-th row of Haar_m, and index it as $[h_{t,1}, \ldots, h_{t,N}]$; note that $h_{t,n} \in \mathbb{R}$.
  for $n = 1, 2, \ldots, N$ **do**
    If $h_{t,n} \neq 0$, query $\mathcal{A}_n$ for its output, and assign it to $\hat{x}_{t,n} \in \mathbb{R}^d$; otherwise, $\hat{x}_{t,n}$ is arbitrary.
    Define $w_{t,n} = h_{t,n}\hat{x}_{t,n} \in \mathbb{R}^d$.
  end for
  Predict $x_t = \sum_{n=1}^{N} w_{t,n} \in \mathbb{R}^d$, receive loss gradient $g_t \in \mathbb{R}^d$.
  for $n = 1, 2, \ldots, N$ **do**
    If $h_{t,n} \neq 0$, compute $\hat{g}_{t,n} = h_{t,n} g_t$ and send it to $\mathcal{A}_n$ as its surrogate loss gradient.
  end for
end for

Algorithm 4.5 Anytime Haar OLR (Algorithm 4.4 with doubling trick).

for $m = 1, 2, \ldots,$ **do**
  Run Algorithm 4.4 for 2^m rounds, which uses the matrix Haar_m. The hyperparameter is set to 1.
end for

Remark 4.1. *It is equivalent to view Algorithm 4.4 as operating on a $dT \times dT$ "master" dictionary matrix $\mathcal{H}$, defined block-wise as the following: for all $(i, j) \in [1 : T]^2$, the (i, j)-th block of $\mathcal{H}$ is the product of the (i, j)-th entry of* Haar_m *(which is a scalar) and the d-dimensional identity matrix I_d. That is, $\mathcal{H}$ is a block matrix; each block is a diagonal matrix with equal diagonal entries determined by* Haar_m*. Roughly, the algorithm measures distances in $\mathbb{R}^d$ by the L_2 norm, while measuring $\mathbb{R}^T$ by the L_1 norm.*

Computation An appealing property is that most Haar wavelet features are supported on short local intervals. Despite $N = T$, there are only $\log_2 T$ active features in each round. Therefore, the runtime of our algorithm is $O(d \log T)$ per round, matching that of all the baselines we compare to. This local property holds for compactly supported wavelets, most notably the *Daubechies family* (Daubechies, 1988; Cohen et al., 1993). The latter can represent more general, piecewise polynomial signals.

4.4.2 Main result

For almost static environments, our Haar OLR algorithm guarantees the following bounds, by relating comparator smoothness to the sparsity of its Haar wavelet representation. Different from (Jacobsen and Cutkosky, 2022) which only contains P-dependent bounds, we also provide a K-switching regret bound, in order to avoid using $P = O(KM)$.[9] Interestingly, the proofs of the following two bounds are quite different: the first uses *exact sparsity*, while the second uses *approximate sparsity*.

Theorem 4.2 (Switching regret)**.** *For all $T \in \mathbb{N}_+$ and $u_{1:T} \in \mathbb{R}^{dT}$, Algorithm 4.5 guarantees*

$$\mathrm{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\|\bar{u}\|_2 \sqrt{T} + \sqrt{K\bar{E}}\right). \tag{4.9}$$

Theorem 4.3 (Path length bound)**.** *For all $T \in \mathbb{N}_+$ and $u_{1:T} \in \mathbb{R}^{dT}$, Algorithm 4.5 guarantees*

$$\mathrm{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\|\bar{u}\|_2 \sqrt{T} + \sqrt{P\bar{S}}\right). \tag{4.10}$$

[9] Recall that one of our motivations is to remove M from the existing bounds.

It can be verified that for all comparators $u_{1:T}$, our bounds are at least as good as prior works. The optimality is a more subtle issue, as one should compare *upper bound functions* (of $u_{1:T}$) to *lower bound functions* in a global manner, rather than comparing the exponents of T in minimax online learning. Nonetheless, we present two examples of $u_{1:T}$, where the improvement can be more clearly seen through better exponents.

Example 4.1 (Tracking outliers)**.** *Consider the situation where $u_{1:T}$ has a locally outlying scale: we set all the instantaneous comparators u_t to 1, except $k \leq \sqrt{T}$ consecutive members which are set to $\sqrt{T}$. Crucially, $|\bar{u}| = O(1)$ and $\bar{S} = O(k\sqrt{T})$, while $M = \sqrt{T}$ and $S = \Theta(T)$. Both our bounds, i.e., (4.9) and (4.10), are $\tilde{O}(\sqrt{kT})$, while the fine baseline (4.4) is $\tilde{O}(T^{3/4})$, and the coarse baseline (4.3) is $\tilde{O}(T)$. The largest gain is observed when k is a constant, i.e., the comparator is subject to a short but large perturbation.*

Example 4.2 (Persistent oscillation)**.** *Consider the situation where $\bar{u} = 1$, and all the instantaneous comparators oscillate around $\bar{u}$: $u_t = \bar{u} + \alpha_t/\sqrt{T}$. $\alpha_t = 1$ or -1, and it only switches sign for k times. Notice that $\bar{S} = \sqrt{T}$, while $S = \Theta(T)$. All the baselines are $\tilde{O}\left(\sqrt{T} + k^{1/2}T^{1/4}\right)$, while both our bounds are $\tilde{O}(\sqrt{T})$. The largest gain is observed when $k = T - 1$, i.e., the comparator switches in every round.*

In summary, we show that existing bounds are suboptimal, while the optimality of our results remains to be studied. It highlights the importance of *comparator energy* and *variability* in the pursuit of better algorithms, which have not received enough attention in the literature. Next, we briefly sketch the proofs of these bounds.

Proof sketch The switching regret bound mostly follows from a very simple observation: if a sequence is constant throughout the support of a Haar wavelet feature, then its transform domain coefficient for this feature is zero. As features on the same scale j do not overlap, a K-switching comparator can only induce K nonzero coefficients on the j-th scale. There are at most $K \log_2 T$ nonzero coefficients in total, therefore

$\text{Sparsity}_{\mathcal{H}} = \tilde{O}(K)$. The bound (4.9) is obtained by applying this argument after taking out the average of $u_{1:T}$.

As for the path length bound, the idea is to consider the *reconstructed* sequences, using transform domain coefficients on a single scale j. These are usually called *detail sequences* in the wavelet literature (Mallat, 2008). Each detail sequence has a relatively simple structure, whose path length and variability can be associated to the magnitude of its transform domain coefficients. Moreover, as these detail sequences are certain "locally averaged" and "globally centered" versions of the actual comparator $u_{1:T}$, their regularities are dominated by the regularity of $u_{1:T}$ itself. In combination, this yields a relation between $P\bar{S}$ and the coefficients' L_1 norm, i.e., $\sum_{n=1}^{N} \left\| z^{(n)} \right\|_2$ in Theorem 4.1, from which the bound is established.

Compared to the analysis of (Jacobsen and Cutkosky, 2022), the key advantage of our analysis is the decoupling of function approximation from the generic sparsity-based regret bound. The former is algorithm-independent, while the latter can be conveniently combined with advances in static online learning. With the help of approximation theory (e.g., Fourier features, wavelets, and possibly deep learning further down the line), intuitions are arguably clearer in this way, and solutions could be more precise (compared to analyses that "mix" function approximation with regret minimization).

MRA in online learning On a broader scope, wavelets embody the idea of *Multi-Resolution Analysis* (MRA), which is reminiscent of the classical *geometric covering* (GC) construction in adaptive online learning (Daniely et al., 2015). Such a construction starts from a class of *GC time intervals*, which are equivalent to the support of Haar wavelet features. On each GC interval, a static online learning algorithm is defined (corresponding to using an all-one feature, cf., Section 4.3.2); and then, the outputs of these "local" algorithms are aggregated by a *sleeping expert* algorithm on

top (Luo and Schapire, 2015; Jun et al., 2017). Algorithmically, our innovation is introducing sign changes in the features, accompanied by a different, additive way to aggregate base algorithms. For tackling nonstationarity, both approaches have their own strengths: the GC construction can produce strongly adaptive guarantees (Type 4 in Section 1.4) on subintervals of the time horizon, while our algorithm does not need a bounded domain. Their possible connections are intriguing.

Lipschitz vs strongly convex losses Finally, we comment on the choice of loss functions in unconstrained dynamic OCO. Besides the Lipschitz assumption we impose, a fruitful line of works by Baby and Wang (Baby and Wang, 2019, 2020, 2021; Baby et al., 2021; Baby and Wang, 2022) considered an alternative setting with strong convexity, motivated by the prevalence of the square loss in statistics. Their focus is primarily on bounded domains, as (Baby and Wang, 2019) showed that evaluated under the square loss, a lower bound for the unconstrained dynamic regret is $\Omega(P^2)$. A sublinear regret bound here requires $P = o(\sqrt{T})$, rather than $P = o(T)$ with Lipschitz losses – that is, the environment is required to be "more static" than the typical requirement in the Lipschitz setting.

Essentially, such a behavior is due to the large penalty that the square loss imposes on outliers. An adversary in online learning can deliberately pick the loss functions such that some of the learning agent's predictions are large outliers with "huge" (square) losses, while the offline optimal comparator sequence suffers zero losses. Using the Lipschitz losses instead may offer an advantage on unbounded domains, due to being more tolerant to these outliers. Furthermore, Lipschitz losses do not necessarily have minimizers – this is useful for decision problems (as opposed to estimation), where a ground truth may not exist.[10]

[10]An example is financial investment without budget constraints: doubling the invested amount also doubles the return.

4.5 Application and experiment

This section presents an application of our framework to time series forecasting. Roughly speaking, we aim to address the following question:

Given a *black box* forecaster, can we make it provably robust against (structured) nonstationarity?

Along the way, our objective is to show that

- Simultaneously handling unconstrained domains and dynamic comparators in online learning brings downstream benefits in time series forecasting.
- Our sparse coding framework can enhance empirically developed forecasting strategies.

Setting Let us consider the following forecasting problem, which resembles the OCO game. The difference is that, here, we further assume access to a black box forecaster $\mathcal{A}$. In each (the t-th) round,

1. The black box forecaster $\mathcal{A}$ produces a prediction $a_t \in \mathbb{R}^d$ based on the observed history ($z_{1:t-1}$ and $l_{1:t-1}$).
2. After observing a_t, we make a prediction $x_t \in \mathbb{R}^d$.
3. The environment reveals a true value $z_t \in \mathbb{R}^d$ and a convex loss function $l_t : \mathbb{R}^d \to \mathbb{R}$. l_t is G-Lipschitz with respect to $\|\cdot\|_2$, and z_t is one of its minimizers satisfying $l_t(z_t) = 0$.

Our goal is to achieve low total loss $\sum_{t=1}^{T} l_t(x_t)$. Since trivially picking $x_t = a_t$ already achieves a total loss of $\sum_{t=1}^{T} l_t(a_t)$, our goal is to improve it in certain situations, by designing a more sophisticated prediction rule based on a_t.

Intuition In the above setting, $\mathcal{A}$ can be *any* algorithm that predicts $z_{1:T}$ in a reasonable, but non-robust manner. Taking the weather forecasting for example, there are a few notable cases.

- $\mathcal{A}$ is a simulator of the governing meteorological equations, which uses the online observations $z_{1:t-1}$ as boundary conditions.
- $\mathcal{A}$ is an autoregressive model, which predicts a linear combination of the past observations. The coefficients are determined by statistical modeling.
- $\mathcal{A}$ is a large deep learning model trained on offline datasets (e.g., the weather history at geographically similar locations).

Even though such forecasters typically lack performance guarantees, their predictions can be used to construct time-varying Bayesian priors (see our discussion in the Introduction): given a_t, we will apply a *fine-tuning* adjustment δ_t to predict $x_t = a_t + \delta_t$. Intuitively, the total loss is low if a_t is close to the true value z_t, i.e., when the prior is good.

Reduction to unconstrained dynamic regret Concretely, if $x_t = a_t + \delta_t$, then due to convexity, for all subgradients $g_t \in \partial l_t(x_t)$ we have $l_t(x_t) - l_t(z_t) \le \langle g_t, \delta_t \rangle - \langle g_t, z_t - a_t \rangle$. The RHS is the instantaneous regret of δ_t in an OLO problem with loss gradient g_t and comparator $z_t - a_t$. Applying our unconstrained dynamic OLO algorithm, the total loss in forecasting can be bounded as

$$\sum_{t=1}^{T} l_t(x_t) \le \mathrm{Regret}_T(z_{1:T} - a_{1:T}).$$

That is, the total loss bound adapts to the complexity of the *error sequence* $z_{1:T} - a_{1:T}$ (of the given black box forecaster). This contains $a_{1:T} = 0$ as a special case, where no side information is assumed.

Let us compare this bound to the baseline $\sum_{t=1}^{T} l_t(a_t)$, which corresponds to trivially picking $x_t = a_t$.

- If $z_{1:T} = a_{1:T}$, i.e., the black box $\mathcal{A}$ is perfect, then the baseline loss is $\sum_{t=1}^{T} l_t(a_t) = 0$. In this case, due to Theorem 4.1, our general sparse coding framework guarantees $\sum_{t=1}^{T} l_t(x_t) \leq \varepsilon G$, where $\varepsilon > 0$ is an arbitrary hyper-parameter. That is, our algorithm is worse than the baseline by at most a constant.

- If $z_{1:T} \neq a_{1:T}$, then in general, the baseline loss $\sum_{t=1}^{T} l_t(a_t)$ is linear in T. In contrast, our algorithm could guarantee a sublinear $\text{Regret}_T(z_{1:T} - a_{1:T})$, thus also a sublinear total loss, when the error sequence $z_{1:T} - a_{1:T}$ is structurally simple (e.g., sparse under a transform, or low path length) with respect to our prior knowledge.

In summary, the idea is that by sacrificing at most a constant loss when $\mathcal{A}$ is perfect ($z_{1:T} = a_{1:T}$), we could robustify $\mathcal{A}$ against certain structured unseen environments, improving the linear total loss to a sublinear rate.

Importance of unconstrained domain The above application critically relies on the ability of our algorithm to handle unconstrained domains. To demonstrate this, suppose we instead use the bounded domain algorithm from (Zhang et al., 2018a) to pick the fine-tuning adjustment δ_t. Then, the above analysis only holds if an upper bound D of the maximum error $\max_t \|z_t - a_t\|_2$ is known a priori – this is a stringent requirement in practice. Furthermore, when $z_{1:T} = a_{1:T}$, such an alternative approach only guarantees $\sum_{t=1}^{T} l_t(x_t) \leq \tilde{O}(D\sqrt{T})$, which is considerably worse than the baseline 0. In other words, the alternative fine-tuning strategy could ruin the black box forecaster $\mathcal{A}$, when the latter performs well.

For the rest of this section, we present experiments for this time series application. Section 4.5.1 demonstrates the power law phenomenon, which shows that both the time series $z_{1:T}$ and the error sequence $z_{1:T} - a_{1:T}$ could exhibit exploitable structures. This implies good performance guarantees using our theoretical framework. Section 4.5.2 goes one step further by actually testing the fine-tuning performance of our algorithm.

4.5.1 Power law phenomenon

This subsection further verifies the power law phenomenon discussed in Section 4.3.2, with both wavelet and Fourier dictionaries. The goal is to present concrete examples where signal structures can be exploited by our framework, generating more interpretable, sublinear regret bounds.

Wavelet dictionary We first verify the power law on the Haar wavelet dictionary. Intuitively it is suitable when the dynamics of the environment exhibits switching behavior. To this end, consider the following stochastic time series model

$$z_t = z_{t-1}\beta_t + \zeta_t, \tag{4.11}$$

where $\{\beta_t\}$ and $\{\zeta_t\}$ are iid random variables satisfying $\zeta_t \sim \text{Uniform}(-q, q)$ and

$$\beta_t = \begin{cases} -1, & \text{w.p.} \quad p, \\ 1, & \text{w.p.} \quad 1-p. \end{cases}$$

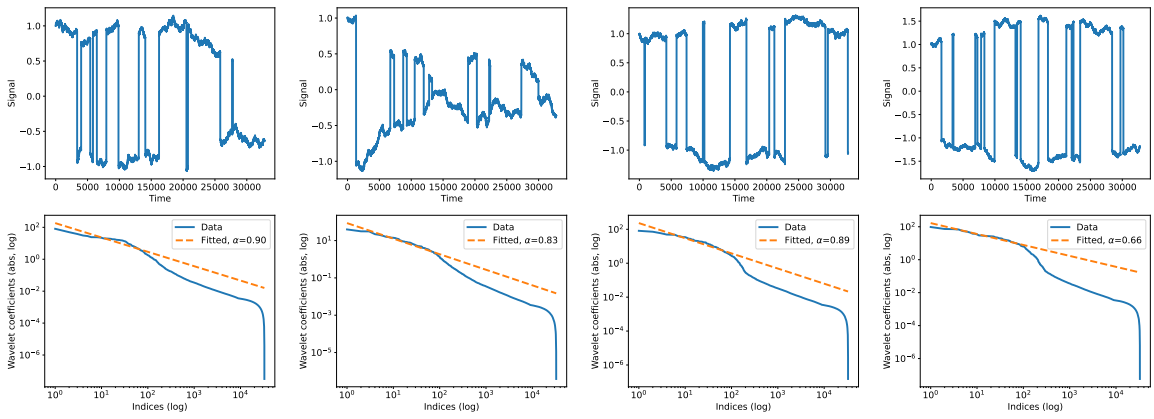
Picking $T = 2^{15} = 32768$, $p = 0.0005$ and $q = 0.005$, we generate four sample paths of $z_{1:T}$ using four *arbitrary* random seeds (2020, 2021, 2022 and 2023), and the obtained time domain signals are plotted in the first row of Figure 4·3. As the switching probability p is chosen to be low enough, all the sample paths exhibit a small amount of sharp switches, corrupted by the noise term ζ_t. According to our

intuition from signal processing, the Haar wavelet transform of these signals is sparse.

Now let us verify this intuition. We take the Haar wavelet transform of these signals, sort the transform domain coefficients and plot the results on log-log scales – these are shown as the solid blue lines in the second row of Figure 4·3. Using the largest 100 transform domain coefficients on each plot, we fit a liner model using least square, which is shown as the dashed orange line. The slope of each line is $-\alpha$, where α is displayed in the legend. It can be seen that for all four sample paths, the fitted α is within $(0.5, 1)$, thus justifying the power law phenomenon (Price, 2021). Given α, the regret of our Haar wavelet algorithm is $\tilde{O}(T^{1.5-\alpha})$, as shown in Section 4.3.2.

As for the implication in time series forecasting, let us consider forecasting $z_{1:T}$ with $a_{1:T} = 0$, i.e., without the external forecaster $\mathcal{A}$. Given the power law, the total forecasting loss of our fine-tuning approach is $\sum_{t=1}^{T} l_t(x_t) \leq \tilde{O}(T^{1.5-\alpha})$.

We also remark that although only four sample paths are demonstrated, we observe the power law phenomenon on all random seeds we tried in the experiment.



(a) Seed= 2020. **(b)** Seed= 2021. **(c)** Seed= 2022. **(d)** Seed= 2023.

Figure 4·3: Verifying the power law on the Haar Wavelet dictionary. First row: time domain signals. Second row: sorted transform domain coefficients on a log-log plot. The dashed orange line is the best linear fit on the log-log plot, using the largest 100 transform domain coefficients. From left to right: four arbitrary random seeds.

Fourier dictionary Next, we verify the power law on the Fourier dictionary. Here we use the Jena weather forecasting dataset,[11] which records the weather data at a German city, Jena, every 10 minutes. We take the data from Jan 1st, 2010 till July 1st, 2022, consisting of $T = 656956$ time steps. Two different modalities, namely the temperature and the humidity, are considered. The actual temperature and humidity sequences are plotted in Figure 4·4.

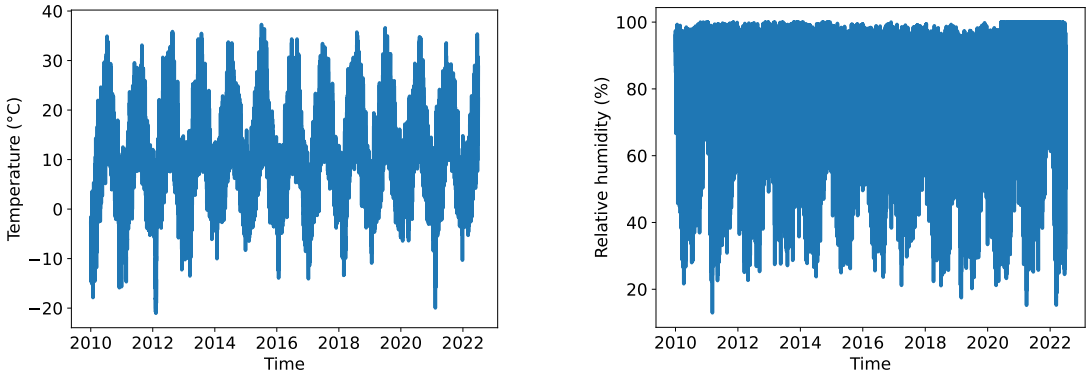


Figure 4·4: Time domain behavior of the weather data.

For the sequence of temperature $z_{1:T}$, we perform its *Discrete Fourier Transform* (DFT), which returns T complex number as the frequency domain coefficients. We discard the second half of the coefficients due to symmetry, since the input of the transform is real. For the remaining coefficients, we take their absolute values, sort them and plot the result on a log-log plot. Similar to the wavelet experiment, we also fit a linear model using the largest 100 transform domain coefficients. These are shown as Figure 4·5 (Left), which exhibit the power law phenomenon.

Furthermore, we perform the same procedure on the temperature difference sequence $\{z_{t+1} - z_t\}$, where the t-th entry is the change of temperature from the t-th time step to the $t+1$-th time step. The result is shown as Figure 4·5 (Right). Although the tail is heavier, we can still observe similar power-law phenomenon for large transform

[11] Available at `https://www.bgc-jena.mpg.de/wetter/`.

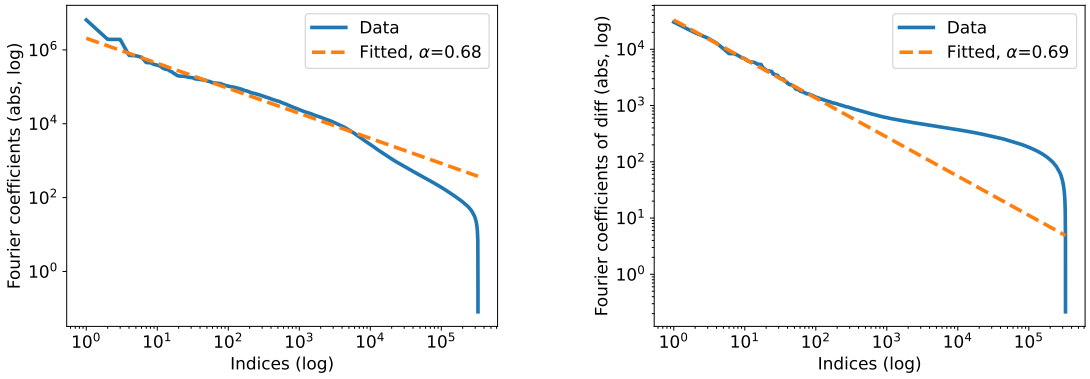


Figure 4·5: Verifying the power law on the Fourier dictionary. Left: the DFT of the temperature sequence. Right: the DFT of the temperature difference sequence.

domain coefficients.

Now, let us discuss again the implication of the observed power law in time series forecasting. First, consider forecasting $z_{1:T}$ without $\mathcal{A}$. Given the power law of $z_{1:T}$ itself, the Fourier version of our forecaster guarantees sublinear total loss. Next, consider forecasting $z_{1:T}$ with $\mathcal{A}$ being the *zeroth-order hold* forecaster, i.e, $a_t = z_{t-1}$. The power law of the difference sequence $\{z_{t+1} - z_t\}$ implies good forecasting performance of our framework in this context.

Parallel results on the humidity sequence are reported in Figure 4·6, with a similar qualitative behavior. It illustrates the prevalence of the power law across different types of the data.

4.5.2 Fine-tuning forecaster

Finally, we test the performance of our forecasting framework on the synthetic switching data and the actual temperature sequence. For the first case, our framework is equipped with the Haar wavelet dictionary. The Fourier dictionary is adopted in the second case. In both cases, we compare our algorithm against the baseline from (Jacobsen and Cutkosky, 2022). More specifically, we take our online learning algorithm

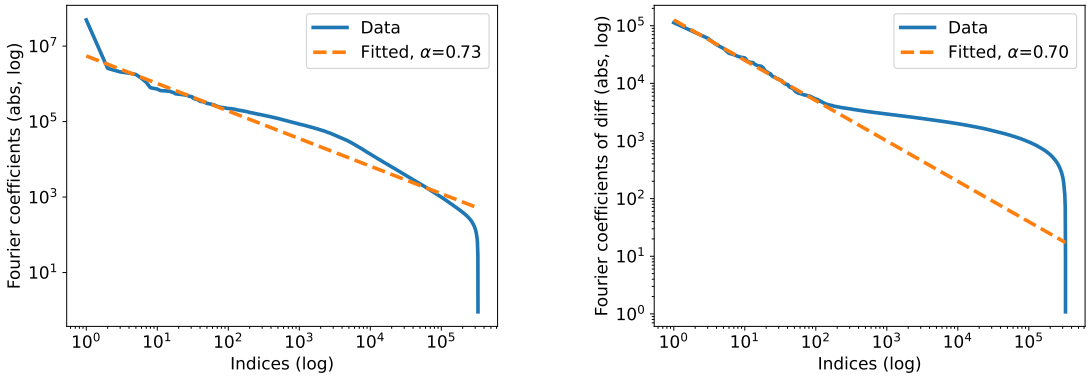


Figure 4·6: Verifying the power law on the Fourier dictionary. Left: the DFT of the humidity sequence. Right: the DFT of the humidity difference sequence.

(Algorithm 4.3) and the algorithm from (Jacobsen and Cutkosky, 2022), plug them both into the time series forecasting workflow introduced at the beginning of this section, and then compare their total forecasting loss.

Concretely, let us start from the wavelet dictionary.

Wavelet dictionary In this case, consider the setting without the external forecaster $\mathcal{A}$. We run both online learning algorithms (our Algorithm 4.3 and the baseline (Jacobsen and Cutkosky, 2022, Algorithm 2)), and use their outputs directly as the time series predictions. Our Algorithm 4.3 is equipped with the Haar wavelet dictionary defined in Section 4.4.1. The configurations of the time series model are the same as the previous subsection, with $T = 2^{15} = 32768$, $p = 0.0005$ and $q = 0.005$. The loss functions l_t are the absolute loss.

Both algorithms require a confidence hyperparameter ε, and we set it to 1. Since the time series data (4.11) is random, we run both algorithms on 10 random seeds, and calculate their total loss. Our algorithm achieves a total loss of 44048, which is considerably lower than the baseline's total loss 62465. This is consistent with the theoretical results developed so far.

Fourier dictionary Next, we turn to the task of temperature forecasting. The data is reported in the previous subsection. We take its first $T = 50000$ entries, and assign it to the true time series $z_{1:T}$; the loss functions l_t are the absolute loss. The black box forecaster $\mathcal{A}$ is assigned to the zeroth-order hold forecaster, i.e., $a_t = z_{t-1}$.

For our framework, we need to specify the dictionary. Although using the entire DFT matrix could lead to low regret guarantees (as demonstrated by the power law), this is computationally challenging. Instead, we exploit the fact that the weather is naturally periodic, with the period of one day. Picking the base frequency ω accordingly, we define features indexed by k (the harmonic order) as

$$h_{t,2k-1} = \cos(k\omega t),$$

$$h_{t,2k} = \sin(k\omega t).$$

By specifying the maximum order K, we obtain $2K$ features $\{h_{t,2k-1}, h_{t,2k}\}_{k\in[1:K]}$ from this construction. An all-one feature is further added, making the dictionary size $N = 2K + 1$.

Again, we set the confidence hyperparameter $\varepsilon = 1$ for our algorithm. The total loss as a function of the dictionary size N is plotted in Figure 4·7. Notably, the case of $N = 0$ is equivalent to trivially following the advice of the given forecaster $\mathcal{A}$: $x_t = a_t = z_{t-1}$. It can be seen that our fine-tuning framework ($N > 0$) actually results in better performance, due to exploiting the structures in the error sequence $z_{1:T} - a_{1:T}$.

We also test the fine-tuning performance of the algorithm from (Jacobsen and Cutkosky, 2022). Same as the above, we set $\varepsilon = 1$. The total loss achieved by this alternative algorithm is 8238, which is around the same as $\mathcal{A}$ itself, and significantly higher than the total loss of our algorithm with moderate amount of features ($N > 5$). This fits the intuition so far: the environment contains persistent dynamics, which the

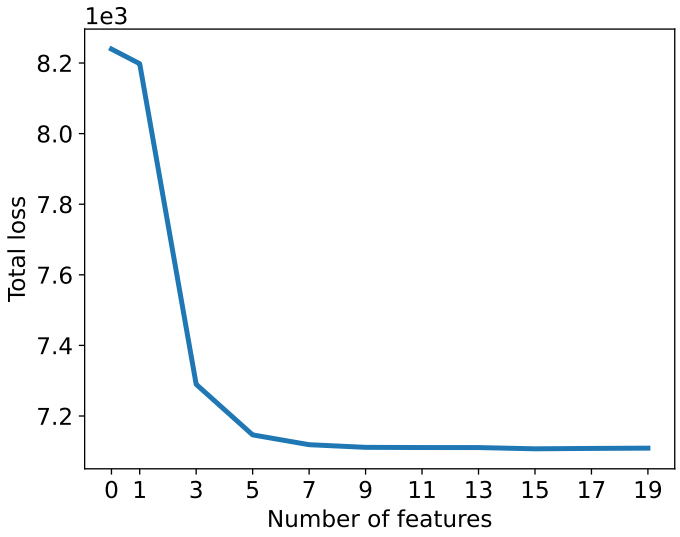


Figure 4·7: Testing our algorithm for temperature forecasting.

algorithm from (Jacobsen and Cutkosky, 2022) cannot handle.

4.6 Summary

This chapter presents a unified study of unconstrained and dynamic online learning, where the two problem structures are naturally connected via comparator adaptivity. Building on the synergy between static parameter-free algorithms and temporal representations, we develop an algorithmic framework achieving a generic sparsity-adaptive regret bound. Equipped with the wavelet dictionary, our framework improves the quantitative results from (Jacobsen and Cutkosky, 2022), by adapting to finer characterizations of the comparator sequence.

For future works, several interesting questions could stem from this paper. For example,

- Our regret bound is stated against individual comparator sequences. One could investigate the implication of this result in stochastic environments, where the comparator statistics may take more concrete forms.

- Besides the sparsity and the energy studied in this paper, an interesting open problem is investigating alternative complexity measures of the comparator, possibly drawing connections to statistical learning theory.

- Our framework builds on pre-defined dictionary inputs. The quantitative benefit of using a data-dependent dictionary is unclear.

- Beyond wavelets, one may investigate the combination of the sparse coding framework with other function approximators, such as neural networks.

4.7 Proofs

4.7.1 General framework

Lemma 4.2

Proof of Lemma 4.2. Subsuming poly-logarithmic factors, the static regret bound of our static subroutine (Algorithm 4.2) can be written as

$$\sum_{t=1}^{T} \hat{g}_t \left(\hat{x}_t - \hat{u}\right) \leq \varepsilon \hat{G} + |\hat{u}| \left(\hat{G} + \sqrt{\sum_{t=1}^{T} \hat{g}_t^2}\right) \cdot \text{polylog}\left(|\hat{u}|, T, \varepsilon^{-1}\right),$$

where $\hat{u}$ is any 1D static comparator that the subroutine handles.

Now, for any single-directional comparator $u_{1:T} \in \text{span}(h_{1:T})$ considered in this lemma, there exists $\hat{u} \in \mathbb{R}$ such that $u_{1:T} = \hat{u} h_{1:T}$. The dynamic regret can be rewritten as

$$\sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u_t) \leq \sum_{t=1}^{T} \langle g_t, x_t - u_t \rangle = \sum_{t=1}^{T} \langle g_t, h_t \hat{x}_t - h_t \hat{u} \rangle = \sum_{t=1}^{T} \hat{g}_t (\hat{x}_t - \hat{u}),$$

and the RHS can be bounded using the static regret bound above. Note that $|\hat{g}_t| = |\langle g_t, h_t \rangle| \leq G$, therefore the surrogate Lipschitz constant $\hat{G}$ from the static regret bound can be assigned to G.

In summary,

$$\begin{aligned}
&\sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u_t) \\
\leq\ & \varepsilon G + |\hat{u}| \left(G + \sqrt{\sum_{t=1}^{T} \langle g_t, h_t \rangle^2} \right) \cdot \text{polylog}\left(|\hat{u}|, T, \varepsilon^{-1} \right) \\
=\ & \varepsilon G + \left(G \frac{\|u_{1:T}\|_2}{\|h_{1:T}\|_2} + \sqrt{\sum_{t=1}^{T} \langle g_t, u_t \rangle^2} \right) \cdot \text{polylog}\left(\frac{\|u_{1:T}\|_2}{\|h_{1:T}\|_2}, T, \varepsilon^{-1} \right) \\
\leq\ & \varepsilon G + \left(G \frac{\|u_{1:T}\|_2}{\|h_{1:T}\|_2} + \sqrt{\sum_{t=1}^{T} \langle g_t, u_t \rangle^2} \right) \cdot \text{polylog}\left(\max_t \|u_t\|_2, T, \varepsilon^{-1} \right),
\end{aligned}$$

where the last line is due to our assumption that $\|h_{1:T}\|_2 \geq 1$. □

Theorem 4.1

Proof of Theorem 4.1. The idea of this theorem is a dynamic analogue of (Cutkosky, 2019b) to aggregate the regret bound of single direction learners. For all decomposition $u_{1:T} = \sum_{n=0}^{N} z^{(n)}$ such that $z^{(n)} \in \text{span}(h_{1:T,n})$ for all $n \in [1:T]$, we have

$$\sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u_t) \leq \langle g_{1:T}, x_{1:T} - u_{1:T} \rangle = \left\langle -g_{1:T}, z^{(0)} \right\rangle + \sum_{n=1}^{N} \left\langle g_{1:T}, w_{1:T,n} - z^{(n)} \right\rangle.$$

For the first term on the RHS, $\left\langle -g_{1:T}, z^{(0)} \right\rangle = -\sum_{t=1}^{T} \left\langle g_t, z_t^{(n)} \right\rangle$. As for the rest,

we plug in Lemma 4.2, with hyperparameter ε/N.

$$\begin{aligned}
&\sum_{n=1}^{N} \left\langle g_{1:T}, w_{1:T,n} - z^{(n)} \right\rangle \\
\leq\ & \sum_{n=1}^{N} \Bigg\{ \varepsilon N^{-1} G \\
&+ \left(G \frac{\left\| z^{(n)} \right\|_2}{\left\| h_{1:T,n} \right\|_2} + \sqrt{\sum_{t=1}^{T} \left\langle g_t, z_t^{(n)} \right\rangle^2} \right) \cdot \operatorname{polylog}\left(\max_t \left\| z_t^{(n)} \right\|_2, T, N, \varepsilon^{-1} \right) \Bigg\} \\
\leq\ & \varepsilon G \\
&+ \left(G \sum_{n=1}^{N} \frac{\left\| z^{(n)} \right\|_2}{\left\| h_{1:T,n} \right\|_2} + \sum_{n=1}^{N} \sqrt{\sum_{t=1}^{T} \left\langle g_t, z_t^{(n)} \right\rangle^2} \right) \cdot \operatorname{polylog}\left(\max_{t,n} \left\| z_t^{(n)} \right\|_2, T, N, \varepsilon^{-1} \right). \square
\end{aligned}$$

4.7.2 Wavelet background

Although the analysis of our framework is simpler than (Jacobsen and Cutkosky, 2022), a challenge is carefully indexing all the quantities to account for the vectorized setting. It is thus important to introduce a few notations to streamline the presentation. Haar_m is the $T \times T$ Haar dictionary matrix defined in Section 4.4.1, with $T = 2^m$. Recall the statistics of the comparator sequence, summarized in Table 4.1.

Local interval Given any scale-location pair (j, l), let the support $I^{(j,l)}$ be the time interval where the feature $h^{(j,l)}$ is nonzero. That is,

$$I^{(j,l)} := [2^j(l-1) + 1 : 2^j l].$$

Moreover, let $I_+^{(j,l)}$ denote the first half of this interval, and $I_-^{(j,l)}$ for the second half. $h^{(j,l)}$ is 1 on $I_+^{(j,l)}$, and -1 on $I_-^{(j,l)}$.

Normalization Let $\tilde{\mathrm{Haar}}_m$ be the orthonormal matrix obtained by scaling the columns of Haar_m. The normalized feature vectors are also denoted by tilde, i.e., instead of h^* and $h^{(j,l)}$, the normalized features are $\tilde{h}^*$ and $\tilde{h}^{(j,l)}$.

Coordinate sequence Consider any comparator sequence $u_{1:T} \in \mathbb{R}^{dT}$. For all coordinate $i \in [1 : d]$, we define its i-th coordinate sequence as $u_{1:T}^{(i)} \in \mathbb{R}^T$: the t-th

entry of this coordinate sequence $u_{1:T}^{(i)}$, denoted by $u_t^{(i)}$, is the i-th coordinate of u_t.

Transform domain coefficient We will also use the transform domain coefficients of $u_{1:T}$, under the Haar wavelet transform. In the single-feature, generic setting (Section 4.3.2), we denoted a single transform domain coefficient by $\hat{u} \in \mathbb{R}$. With wavelets, the transform domain encodes dT-dimensional vectors. According to our convention so far, we will denote them by scale-location pairs (j, l): given a (j, l) pair, the "coefficient" $\hat{u}^{(j,l)}$ is a d-dimensional vector. There are $T - 1$ pairs of (j, l) in total; complementing the representation, we use another $\hat{u}^* \in \mathbb{R}^d$ to represent the "coefficient" for the all-one feature.

Given any scale parameter $j \in [1 : \log_2 T]$ and location parameter $l \in [1 : 2^{-j}T]$, let

$$\hat{u}^{(j,l)} := \left[\left\langle \tilde{h}^{(j,l)}, u_{1:T}^{(1)} \right\rangle, \ldots, \left\langle \tilde{h}^{(j,l)}, u_{1:T}^{(d)} \right\rangle \right],$$

and for the all-one feature,

$$\hat{u}^* := \left[\left\langle \tilde{h}^*, u_{1:T}^{(1)} \right\rangle, \ldots, \left\langle \tilde{h}^*, u_{1:T}^{(d)} \right\rangle \right].$$

That is, each entry is the inner product between the normalized feature and a coordinate sequence from $u_{1:T}$.

Due to the orthonormality of the applied transform (specified by the normalized features $\tilde{h}^*$ and $\tilde{h}^{(j,l)}$), the energy is preserved between the time domain and the transform domain, i.e.,

$$E = \|u_{1:T}\|_2^2 = \|\hat{u}^*\|_2^2 + \sum_{j,l} \left\|\hat{u}^{(j,l)}\right\|_2^2,$$

and also the second order variability (the energy of the centered dynamic component within $u_{1:T}$),

$$\bar{E} = \sum_{t=1}^{T} \|u_t - \bar{u}\|_2^2 = \sum_{j,l} \left\|\hat{u}^{(j,l)}\right\|_2^2. \tag{4.12}$$

Moreover, since $\tilde{h}^*$ equals $1/\sqrt{T}$ times the all-one vector,

$$\|\hat{u}^*\|_2^2 = \sum_{i=1}^{d} \left\langle \tilde{h}^*, u_{1:T}^{(i)} \right\rangle^2 = \sum_{i=1}^{d} \left(\frac{1}{\sqrt{T}} \sum_t u_{1:T}^{(i)} \right)^2 = T \sum_{i=1}^{d} \left(\frac{1}{T} \sum_t u_{1:T}^{(i)} \right)^2 = \|\bar{u}\|_2^2 T. \tag{4.13}$$

Detail reconstruction Given the transform domain coefficients, we can reconstruct details of the comparator $u_{1:T}$ on the time domain. Similar to our notation in the generic framework (Section 4.3.2), we keep the letter z, but replace the index n by (j,l), which is more suitable for indexing wavelets.

Let $z^{(j,l)} \in \mathbb{R}^{dT}$ be the detail of $u_{1:T}$ along the (j,l)-th feature. It is the concatenation of T vectors in $\mathbb{R}^d$, and for all t, the t-th of these vectors is defined by

$$z_t^{(j,l)} := \hat{u}^{(j,l)} \tilde{h}_t^{(j,l)} \in \mathbb{R}^d.$$

Similarly, we can define the detail z^* along the feature $\tilde{h}^*$. Its t-th component is

$$z_t^* := \hat{u}^* \tilde{h}_t^*,$$

and clearly, the RHS does not depend on t since $\tilde{h}^*$ is scaled from the all-one feature h^*.

Let us also sum the details across different locations. Given a scale j, let

$$z^{(j)} := \sum_l z^{(j,l)} \in \mathbb{R}^{dT}.$$

Note that the summands are sequences that do not overlap: at each entry, only one of the summand sequence is nonzero. The full reconstruction is obtained by summing all the details,

$$u_{1:T} := z^* + \sum_{j=1}^{\log_2 T} z^{(j)}.$$

Statistics of the detail sequence We can define statistics of the detail sequences just like the statistics of the comparator $u_{1:T}$. Specifically, define the first order variability of the (j,l)-th detail as

$$\bar{S}^{(j,l)} := \sum_{t=1}^{T} \left\| z_t^{(j,l)} \right\|_2.$$

Note that since the $z_t^{(j,l)}$ sequence is centered (with average being equal to 0), its first order variability equals its norm sum. Summing over the locations, the first order

variability at the j-th scale is

$$\bar{S}^{(j)} := \sum_{t=1}^{T} \left\| z_t^{(j)} \right\|_2 ,$$

which equals $\sum_l \bar{S}^{(j,l)}$.

Similarly, we can define the path length of the detail sequences. A caveat is that we only count the path length *within* the support $I^{(j,l)}$ of the feature $h^{(j,l)}$,

$$P^{(j,l)} := \sum_{t=2^j(l-1)+1}^{2^j l - 1} \left\| z_{t+1}^{(j,l)} - z_t^{(j,l)} \right\|_2 .$$

The comparator's move when the support changes does not count. Summing over the locations,

$$P^{(j)} := \sum_l P^{(j,l)} .$$

4.7.3 Generic Haar OLR result

With the notation from the previous subsection, we now present a generic sparsity adaptive regret bound for Algorithm 4.4 (fixed T Haar OLR). Since the latter is a variant of our main sparse coding framework, the result can be analogously derived. However, we need to be careful with the notations.

Lemma 4.3. *For any m, $T = 2^m$ and $u_{1:T} \in \mathbb{R}^{dT}$, with any hyperparameter $\varepsilon > 0$, Algorithm 4.4 guarantees*

$$\mathrm{Regret}_T(u_{1:T}) \le \varepsilon G + G \left(\left\| z^* \right\|_2 + \sum_{j=1}^{\log_2 T} \sum_{l=1}^{2^{-j}T} \left\| z^{(j,l)} \right\|_2 \right) \cdot \mathrm{polylog}\left(M, T, \varepsilon^{-1} \right) .$$

The proof sums the regret bound of the d-dimensional version of the static subroutine (Lemma 4.1), across T different copies. It is very similar to Theorem 4.1, therefore omitted.

It might be more convenient to use the transform domain coefficients $\hat{u}^{(j,l)}$ in the bound, rather than the reconstructed details $z^{(j,l)}$. In this case, we have

$$\left\| z^{(j,l)} \right\|_2^2 = \sum_t \left\| z_t^{(j,l)} \right\|_2^2 = \sum_t \left[\left\| \hat{u}^{(j,l)} \right\|_2^2 \left| \tilde{h}_t^{(j,l)} \right|^2 \right] = \left\| \hat{u}^{(j,l)} \right\|_2^2 \sum_t \left| \tilde{h}_t^{(j,l)} \right|^2 = \left\| \hat{u}^{(j,l)} \right\|_2^2 .$$

Similarly,

$$\|z^*\|_2^2 = \|\hat{u}^*\|_2^2 .$$

Therefore,

$$\mathrm{Regret}_T(u_{1:T}) \le \varepsilon G + G\left(\|\hat{u}^*\|_2 + \sum_{j=1}^{\log_2 T}\sum_{l=1}^{2^{-j}T}\left\|\hat{u}^{(j,l)}\right\|_2\right)\cdot \mathrm{polylog}\left(M, T, \varepsilon^{-1}\right). \quad (4.14)$$

Besides, our analysis will rely on two auxiliary lemmas. First, we show that local averaging makes a signal more regular. Consider any signal $u_{1:T} \in \mathbb{R}^T$, with the t-th round component $u_t \in \mathbb{R}^d$. Local averaging refers to replacing any k consecutive components of $u_{1:T}$ by their average, i.e., setting

$$u_{\tau+1}, \ldots, u_{\tau+k} = k^{-1}\sum_{i=1}^{k} u_{\tau+i},$$

for some $\tau \in [0 : T-k]$.

Lemma 4.4. *Let a signal $w_{1:T} \in \mathbb{R}^{dT}$ be the result of $u_{1:T}$ after local averaging, and $\bar{w} = T^{-1}\sum_{t=1}^{T} w_t \in \mathbb{R}^d$. Then, the path length, the norm sum and the energy of $w_{1:T}$, including their centered versions, are all dominated by those of $u_{1:T}$. That is,*

1. $\sum_{t=1}^{T-1}\|w_{t+1}-w_t\|_2 \le \sum_{t=1}^{T-1}\|u_{t+1}-u_t\|_2$;
2. $\sum_{t=1}^{T}\|w_t-\bar{w}\|_2 \le \sum_{t=1}^{T}\|u_t-\bar{u}\|_2$;
3. $\sum_{t=1}^{T}\|w_t-\bar{w}\|_2^2 \le \sum_{t=1}^{T}\|u_t-\bar{u}\|_2^2$.
4. $\sum_{t=1}^{T}\|w_t\|_2 \le \sum_{t=1}^{T}\|u_t\|_2$, *and* $\sum_{t=1}^{T}\|w_t\|_2^2 \le \sum_{t=1}^{T}\|u_t\|_2^2$.

Proof of Lemma 4.4. Starting from the first part of the lemma, we prove for the general case of $0 < \tau < T-k$. The boundary cases ($\tau = 0$ and $\tau = T-k$) are analogous.

Local averaging only affects the path length caused by the averaged entries $u_{\tau+1}, \ldots, u_{\tau+k}$, and the two entries u_τ and $u_{\tau+k+1}$ right besides averaging boundary; this original path length quantity in $u_{1:T}$ is $\sum_{i=0}^{k}\|u_{\tau+i+1}-u_{\tau+i}\|_2$. After averaging,

the path length among these entries becomes

$$\begin{aligned}
&\left\| u_\tau - k^{-1}\sum_{i=1}^{k} u_{\tau+i} \right\|_2 + \left\| k^{-1}\sum_{i=1}^{k} u_{\tau+i} - u_{\tau+k+1} \right\|_2 \\
=\ & k^{-1}\left\| \sum_{i=1}^{k} (u_\tau - u_{\tau+i}) \right\|_2 + k^{-1}\left\| \sum_{i=1}^{k} (u_{\tau+i} - u_{\tau+k+1}) \right\|_2 \\
\le\ & k^{-1}\sum_{i=1}^{k} \left(\|u_\tau - u_{\tau+i}\|_2 + \|u_{\tau+i} - u_{\tau+k+1}\|_2 \right) \\
\le\ & \sum_{i=0}^{k} \|u_{\tau+i+1} - u_{\tau+i}\|_2 .
\end{aligned}$$

Now consider the second part of the lemma. After local averaging, $\bar{w} = \bar{u}$. The affected part of the signal contributes to the following first order variability

$$\sum_{t=1}^{k} \|w_{\tau+i} - \bar{w}\|_2 = k\left\| k^{-1}\sum_{i=1}^{k} u_{\tau+i} - \bar{u} \right\|_2 = \left\| \sum_{i=1}^{k} u_{\tau+i} - k\bar{u} \right\|_2 \le \sum_{t=1}^{k} \|u_{\tau+i} - \bar{u}\|_2 .$$

As for the third part of the lemma,

$$\begin{aligned}
\sum_{t=1}^{k} \|w_{\tau+i} - \bar{w}\|_2^2 = k\left\| k^{-1}\sum_{i=1}^{k} u_{\tau+i} - \bar{u} \right\|_2^2 & \\
\le k^{-1}\left(\sum_{i=1}^{k} \|u_{\tau+i} - \bar{u}\|_2 \right)^2 \le \sum_{t=1}^{k} \|u_{\tau+i} - \bar{u}\|_2^2 , &
\end{aligned}$$

where the last inequality is due to AM-QM inequality.

The final part of the proof is the uncentered version of Part 2 and 3, which follows the same steps. In fact, any fixed reference point (for the variability) works, i.e., for all $v \in \mathbb{R}^d$,

$$\begin{aligned}
\sum_{t=1}^{T} \|w_t - v\|_2 &\le \sum_{t=1}^{T} \|u_t - v\|_2 , \\
\sum_{t=1}^{T} \|w_t - v\|_2^2 &\le \sum_{t=1}^{T} \|u_t - v\|_2^2 .
\end{aligned}$$

□

The second lemma is a simple one.

Lemma 4.5. *Consider any comparator sequence $u_{1:T}$. For all t, we have $\|u_t - \bar{u}\|_2 \le P$.*

Proof of Lemma 4.5. Starting from the definition,

$$\|u_t - \bar{u}\|_2 = \left\| u_t - \sum_{i=1}^{T} T^{-1} u_i \right\|_2 \le T^{-1} \sum_{i=1}^{T} \|u_t - u_i\|_2 ,$$

and for all $i, t \in [1:T]$, $\|u_t - u_i\|_2 \le P$ due to triangle inequality. □

Next, we are ready to prove the two main results of the Haar OLR algorithm.

4.7.4 Switching regret

For the fixed-T setting, we have

Lemma 4.6. *For any m, $T = 2^m$ and $u_{1:T} \in \mathbb{R}^{dT}$, Algorithm 4.4 with the hyperparameter $\varepsilon = 1$ guarantees*

$$\mathrm{Regret}_T(u_{1:T}) = \tilde{O}\left(\|\bar{u}\|_2 \sqrt{T} + \sqrt{K\bar{E}} \right).$$

Proof of Lemma 4.6. Consider any scale j. Since the supports $\{I^{(j,l)}\}_l$ do not overlap, if $u_{1:T}$ shifts K times, then there are at most K choices of location l such that the transform domain coefficient $\hat{u}^{(j,l)}$ is nonzero. Furthermore, since there are $\log_2 T$ scales in total, there are at most $K \log_2 T$ pairs of (i,l) such that $\hat{u}^{(j,l)}$ is nonzero. Therefore, using Cauchy-Schwarz and (4.12),

$$\sum_{j,l} \left\| \hat{u}^{(j,l)} \right\|_2 \le \sqrt{K \log_2 T} \sqrt{\sum_{j,l} \|\hat{u}^{(j,l)}\|_2^2} = \sqrt{K\bar{E} \log_2 T}.$$

Plugging this into (4.14), and further using (4.13) for $\|\hat{u}^*\|_2$ complete the proof. □

The anytime bound in general follows from the classical doubling trick. A twist is that the analysis is slightly more involved than the standard one, e.g., (Shalev-Shwartz, 2011), as we also need to relate the comparator statistics on each block to those for the entire signal $u_{1:T}$.

Theorem 4.2

Proof of Theorem 4.2. First, assume T can be exactly decomposed into m^* segments with dyadic lengths $2^1, \ldots, 2^{m^*}$. We use $\bar{u}_m$, K_m and $\bar{E}_m$ to represent the statistics of the comparator sequence on the length 2^m block, and let I^m denote the time interval that this block operates on. $\bar{u}$, K and S denote the statistics of the entire signal $u_{1:T}$, cf., Table 4.1. From Lemma 4.6,

$$\begin{aligned}\operatorname{Regret}_T(u_{1:T}) &\leq \sum_{m=1}^{m^*} \tilde{O}\left(\|\bar{u}_m\|_2 \sqrt{2^m} + \sqrt{K_m \bar{E}_m}\right) \\ &\leq \tilde{O}\left[\|\bar{u}\|_2 \left(\sum_{m=1}^{m^*} \sqrt{2^m}\right) + \sum_{m=1}^{m^*} \|\bar{u}_m - \bar{u}\|_2 \sqrt{2^m} + \sum_{m=1}^{m^*} \sqrt{K_m \bar{E}_m}\right].\end{aligned} \tag{4.15}$$

The first term follows from the standard doubling trick,

$$\sum_{m=1}^{m^*} \sqrt{2^m} \leq \frac{\sqrt{2}}{\sqrt{2}-1} \sqrt{2^{m^*}} = O\left(\sqrt{T}\right). \tag{4.16}$$

As for the second term in (4.15), using Cauchy-Schwarz,

$$\sum_{m=1}^{m^*} \|\bar{u}_m - \bar{u}\|_2 \sqrt{2^m} \leq \sqrt{m^* \left(\sum_{m=1}^{m^*} 2^m \|\bar{u}_m - \bar{u}\|_2^2\right)}.$$

$m^* = O(\log T)$, and also observe that the sum (in the parenthesis) on the RHS equals the second order variability of the following signal: for any time t in the m-th block, the signal's component is $\bar{u}_m \in \mathbb{R}^d$. This signal is a locally averaged version of the original comparator $u_{1:T}$, and local averaging decreases the variability. Formally, due to Lemma 4.4, we have

$$\sum_{m=1}^{m^*} \|\bar{u}_m - \bar{u}\|_2 \sqrt{2^m} \leq \tilde{O}\left(\sqrt{\bar{E}}\right). \tag{4.17}$$

For the third term in (4.15), using Cauchy-Schwarz again,

$$\sum_{m=1}^{m^*} \sqrt{K_m \bar{E}_m} \leq \sqrt{\left(\sum_{m=1}^{m^*} K_m\right)\left(\sum_{m=1}^{m^*} \bar{E}_m\right)} \leq \sqrt{K \bar{E}}.$$

The sum on K_m is straightforward. The sum on $\bar{E}_m$ follows the observation that on the m-th block, $\bar{u}_m$ minimizes $\sum_{t\in I^m} \|u_t - x\|_2^2$ with respect to $x \in \mathbb{R}^d$.

Also, notice that the second term in (4.15) is dominated by the third term. If $K = 0$, then both $\sqrt{\bar{E}}$ and $\sqrt{K\bar{E}}$ equal 0. If $K \geq 1$, then $\sqrt{\bar{E}} \leq \sqrt{K\bar{E}}$. Therefore, (4.15) can be written as

$$\text{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\|\bar{u}\|_2 \sqrt{T} + \sqrt{K\bar{E}}\right).$$

As for the general setting where T cannot be exactly decomposed into dyadic blocks: consider the smallest $T^* > T$ such that T^* can be decomposed. Due to doubling intervals, $T^* \leq 2T$. Let us consider a hypothetical length T^* game with the rounds $t > T$ constructed as follows: the loss gradient $g_t = 0 \in \mathbb{R}^d$, and $u_t = \bar{u}$. In this case, with K and $\bar{E}$ still representing the statistics of the length T sequence $u_{1:T}$, the number of switches on the entire time interval $[1 : T^*]$ is at most $K + 1$, and the second order variability on $[1 : T^*]$ is $\bar{E}$. The regret of any algorithm on this hypothetical length T^* game is the same as the length T game, therefore bounding the latter follows from bounding the former. □

4.7.5 Path-length bound

Let us first consider the fixed T setting (Algorithm 4.4) and assume $T = 2^m$ for some m. The static component (i.e., z^*) and the dynamic component (i.e., $u_{1:T} - z^*$) of $u_{1:T}$ are analyzed separately; the former is fairly standard, while the latter is more challenging. We will first consider the dynamic component, and proceed in three steps.

Step 1 Considering any scale j, we aim to show $\sum_l \left\|\hat{u}^{(j,l)}\right\|_2 \leq \sqrt{P^{(j)}\bar{S}^{(j)}}$, which relates the transform domain coefficients to the regularity of the reconstructed signals.

Lemma 4.7. *For all (j, l) pair,*

$$\left\|\hat{u}^{(j,l)}\right\|_2 = 2^{-1/2}\sqrt{P^{(j,l)}\bar{S}^{(j,l)}},$$

and

$$\sum_l \left\|\hat{u}^{(j,l)}\right\|_2 \leq 2^{-1/2}\sqrt{P^{(j)}\bar{S}^{(j)}}.$$

Proof of Lemma 4.7. Let us start from the first part of this lemma, and express the detail sequence $z^{(j,l)}$ more explicitly on its support $I^{(j,l)}$.

$$z_t^{(j)} = \begin{cases} 2^{-j/2}\hat{u}^{(j,l)}, & t \in I_+^{(j,l)}; \\ -2^{-j/2}\hat{u}^{(j,l)}, & t \in I_-^{(j,l)}. \end{cases}$$

Rewriting $P^{(j,l)}$ and $\bar{S}^{(j,l)}$,

$$P^{(j,l)} = \sum_{t=2^j(l-1)+1}^{2^j l - 1} \left\| z_{t+1}^{(j)} - z_t^{(j)} \right\|_2 = 2^{1-j/2} \left\| \hat{u}^{(j,l)} \right\|_2 .$$

$$\bar{S}^{(j,l)} = \sum_{t \in I^{(j,l)}} \left\| z_t^{(j)} \right\|_2 = 2^{-j/2} \left\| \hat{u}^{(j,l)} \right\|_2 \cdot 2^j = 2^{j/2} \left\| \hat{u}^{(j,l)} \right\|_2 ,$$

which yields the equality in the lemma. The second part follows from Cauchy-Schwarz. □

Step 2 Showing that $P^{(j)} \leq P$ and $\bar{S}^{(j)} \leq \bar{S}$. That is, the reconstructed signals are easier than the original comparator $u_{1:T}$. Here, P and $\bar{S}$ should be considered independently.

Lemma 4.8. *For any $u_{1:T}$ and any scale parameter j^*, $P^{(j^*)} \leq P$.*

Proof of Lemma 4.8. From the definition of P and the reconstruction of $u_{1:T}$ from detail sequences,

$$\begin{aligned} P = \sum_{t=1}^{T-1} \|u_{t+1} - u_t\|_2 = \sum_{t=1}^{T-1} \left\| z_{t+1}^* - z_t^* + \sum_j \left(z_{t+1}^{(j)} - z_t^{(j)} \right) \right\|_2 \\ = \sum_{t=1}^{T-1} \left\| \sum_j \left(z_{t+1}^{(j)} - z_t^{(j)} \right) \right\|_2 , \end{aligned}$$

where the last equality is due to z^* being a constant sequence.

Consider removing "shorter" scales with $1 \leq j < j^*$, which is equivalent to local averaging. Due to Lemma 4.4, the path length does not increase, i.e,

$$\sum_{t=1}^{T-1} \left\| \sum_j \left(z_{t+1}^{(j)} - z_t^{(j)} \right) \right\|_2 \geq \sum_{t=1}^{T-1} \left\| \sum_{j \geq j^*} \left(z_{t+1}^{(j)} - z_t^{(j)} \right) \right\|_2 .$$

Then, we can further remove the rounds where the path length is not counted in $P^{(j^*)}$, i.e., when a time $t \in I^{(j^*,l)}$ but $t+1 \in I^{(j^*,l+1)}$.

$$\text{RHS} \geq \sum_l \sum_{t=2^{j^*}(l-1)+1}^{2^{j^*}l-1} \left\| \sum_{j \geq j^*} \left(z_{t+1}^{(j)} - z_t^{(j)} \right) \right\|_2 .$$

Now, consider any location l, which determines the time interval $I^{(j^*,l)} = [2^{j^*}(l-1)+1 : 2^{j^*}l]$. Any detail sequence $z^{(j)}$ with scale $j > j^*$ is constant on this time interval, thus removing it does not change the path length at all. Therefore,

$$P \geq \sum_l \sum_{t=2^{j^*}(l-1)+1}^{2^{j^*}l-1} \left\| z_{t+1}^{(j^*)} - z_t^{(j^*)} \right\|_2 = P^{(j^*)}. \qquad \square$$

As for the first order variability,

Lemma 4.9. *For any $u_{1:T}$ and any scale parameter j^*, $\bar{S}^{(j^*)} \leq \bar{S}$.*

Proof of Lemma 4.9. From the definition, noticing that $\bar{u}$ is entirely captured by the all-one feature,

$$\bar{S} = \sum_{t=1}^{T} \left\| u_t - \bar{u} \right\|_2 = \sum_{t=1}^{T} \left\| \sum_{j=1}^{\log_2 T} z_t^{(j)} \right\|_2 .$$

Due to Lemma 4.4, removing short scales amounts to local averaging, which decreases the variability.

$$\bar{S} \geq \sum_{t=1}^{T} \left\| \sum_{j \geq j^*} z_t^{(j)} \right\|_2 = \sum_l \sum_{t \in I^{(j^*,l)}} \left\| z_t^{(j^*)} + \sum_{j > j^*} z_t^{(j)} \right\|_2 .$$

For any l, consider the support of the (j^*,l)-th feature, $I^{(j^*,l)}$. Observe that $\sum_{j>j^*} z_t^{(j)}$ is time invariant throughout $I^{(j^*,l)}$, let us denote it as $v \in \mathbb{R}^d$. Meanwhile, for some $w \in \mathbb{R}^d$, $z_t^{(j^*)}$ equals w on $I_+^{(j^*,l)}$, the first half of this interval, while being $-w$ on the second half $I_-^{(j^*,l)}$ of this interval. Therefore,

$$\begin{aligned} \sum_{t \in I^{(j^*,l)}} \left\| z_t^{(j^*)} + \sum_{j>j^*} z_t^{(j)} \right\|_2 &= 2^{j^*-1} \left(\|v+w\|_2 + \|v-w\|_2 \right) \\ &\geq 2^{j^*} \|w\|_2 = \sum_{t \in I^{(j^*,l)}} \left\| z_t^{(j^*)} \right\|_2 . \end{aligned}$$

Combining the above,

$$\bar{S} \geq \sum_{l} \sum_{t \in I^{(j^*,l)}} \left\| z_t^{(j^*)} \right\|_2 = \bar{S}^{(j^*)}. \qquad \square$$

Step 3 Summarizing the above relations, and using the property that there are only $\log_2 T$ scales.

Lemma 4.10. *For any m, $T = 2^m$ and $u_{1:T} \in \mathbb{R}^{dT}$, Algorithm 4.4 with the hyperparameter $\varepsilon = 1$ guarantees*

$$\mathrm{Regret}_T(u_{1:T}) = \tilde{O}\left(\|\bar{u}\|_2 \sqrt{T} + \sqrt{P\bar{S}} \right).$$

Proof of Lemma 4.10. Starting from the generic regret bound, (4.14) for Algorithm 4.4.

$$\mathrm{Regret}_T(u_{1:T}) \leq \varepsilon G + G \left(\|\hat{u}^*\|_2 + \sum_{j,l} \left\| \hat{u}^{(j,l)} \right\|_2 \right) \cdot \mathrm{polylog}\left(M, T, \varepsilon^{-1}\right).$$

Due to (4.13), $\|\hat{u}^*\|_2 = \|\bar{u}\|_2 \sqrt{T}$. Then, combining Lemma 4.7, 4.8 and 4.9,

$$\sum_{j,l} \left\| \hat{u}^{(j,l)} \right\|_2 \leq O\left(\sqrt{P\bar{S}} \log_2 T \right).$$

Plugging it into the generic bound completes the proof. $\square$

Finally, we relax our previous assumption of fixed dyadic time horizon T.

Theorem 4.3

Proof of Theorem 4.3. Similar to the analysis of the switching regret (Theorem 4.2), we first consider the situation where the time horizon T can be exactly decomposed into m^* segments with dyadic lengths $2^1, \ldots, 2^{m^*}$. In this situation, we have

$$\begin{aligned} \mathrm{Regret}_T(u_{1:T}) &\leq \tilde{O}\left[\sum_{m=1}^{m^*} \|\bar{u}_m\|_2 \sqrt{2^m} + \sum_{m=1}^{m^*} \sqrt{P_m \bar{S}_m} \right] \\ &\leq \tilde{O}\left[\|\bar{u}\|_2 \sqrt{T} + \sqrt{\bar{E}} + \sum_{m=1}^{m^*} \sqrt{P_m \bar{S}_m} \right], \end{aligned}$$

where the second line follows from the proof of Theorem 4.2, specifically Eq.(4.16) and Eq.(4.17).

Now let us consider the remaining sum on the RHS. Using Cauchy-Schwarz,

$$\sum_{m=1}^{m^*} \sqrt{P_m \bar{S}_m} \leq \sqrt{\left(\sum_{m=1}^{m^*} P_m\right)\left(\sum_{m=1}^{m^*} \bar{S}_m\right)} \leq \sqrt{P\left(\sum_{m=1}^{m^*} \sum_{t=2^m-1}^{2^{m+1}-2} \|u_t - \bar{u}_m\|_2\right)},$$

where

$$\sum_{m=1}^{m^*} \sum_{t=2^m-1}^{2^{m+1}-2} \|u_t - \bar{u}_m\|_2 \leq \sum_{m=1}^{m^*} \sum_{t=2^m-1}^{2^{m+1}-2} (\|u_t - \bar{u}\|_2 + \|\bar{u}_m - \bar{u}\|_2) = \bar{S} + \sum_{m=1}^{m^*} 2^m \|\bar{u}_m - \bar{u}\|_2 .$$

The last sum on the RHS is the first order variability of a locally averaged version of $u_{1:T}$. Due to Lemma 4.4,

$$\sum_{m=1}^{m^*} 2^m \|\bar{u}_m - \bar{u}\|_2 \leq \bar{S}.$$

Combining everything above,

$$\text{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\|\bar{u}\|_2 \sqrt{T} + \sqrt{\bar{E}} + \sqrt{P\bar{S}}\right).$$

It remains to show that $\sqrt{\bar{E}} \leq \sqrt{P\bar{S}}$, thus the former can be absorbed into the latter. Plugging in the definitions, this is equivalent to showing

$$\sum_{t=1}^{T} \|u_t - \bar{u}\|_2^2 \leq \sum_{t=1}^{T} P \|u_t - \bar{u}\|_2 ,$$

and it suffices to prove $\|u_t - \bar{u}\| \leq P$ for all $t \in [1:T]$. This is completed in Lemma 4.5. Till this point, we have shown the desirable result in the situation of "exact dyadic partitioning".

To complete the proof, we turn to the general situation where T cannot be partitioned into dyadic blocks. This follows from a similar "padding" construction from the proof of Theorem 4.2. Let $T^* = 2^{\lceil \log_2 T \rceil}$, and by definition, $T^* \leq 2T$. Let us consider a hypothetical length T^* game with the rounds $t > T$ constructed as follows: the loss gradient $g_t = 0 \in \mathbb{R}^d$, and $u_t = \bar{u}$. Then, the regret of any algorithm on the length T^* hypothetical game equals its regret on the actual length T game, and the regret bound for the former applies to the latter as well: if we write P^* and $\bar{S}^*$ as the

statistics of the extended length T^* comparator, then

$$\text{Regret}_T(u_{1:T}) \leq \tilde{O}\left(\|\bar{u}\|_2 \sqrt{T^*} + \sqrt{P^* \bar{S}^*}\right).$$

Clearly, $\bar{S}^* = \bar{S}$ and $T^* \leq 2T$. As for the path length, $P^* = P + \|u_T - \bar{u}\|_2$, and due to Lemma 4.5, $\|u_T - \bar{u}\|_2 \leq P$. Plugging it back completes the proof. □

Chapter 5

Conclusion and future work

Concluding this dissertation, we summarize our contributions as follows.

- In the first part, we demonstrated how to design better comparator adaptive online learning algorithms using a continuous time scaling approach. Compared to existing works, the new approach reduces the amount of heuristic guessing, and leads to quantitatively stronger performance guarantees in certain scenarios. The workflow is first developed in the standard OCO setting (Chapter 2), and then extended to the variant of OCO with switching costs (Chapter 3). Downstream benefits in the expert problem are also presented.

- The second part of the dissertation uses temporal features to construct better adaptive algorithms for nonstationary environments. Inspired by the ideas in signal processing, we run static comparator adaptive online learning algorithms on a user-specified transform domain, which effectively decomposes the tasks of regret minimization and (temporal) function approximation. The final result is a quantitative improvement over the state of the art.

At the end of each chapter, we discussed fairly specific directions for future works. Here we discuss some future directions on a higher level, which are essentially topics we aim to explore in the years to come.

- A central idea in adaptive online learning (Section 1.3) is to introduce Bayesian prior knowledge. Evidently, this bears much similarity to Bayesian statistics,

but to the best of our knowledge, the characterization of their connection is still quite far from being complete in the existing literature. Addressing this gap is an intriguing future direction, which could lead to algorithmic benefits in both fields.

- Regarding the continuous time scaling approach, a particular interesting question is the role of an adversarial environment in the continuous time limit. It is long known that PDEs, which we obtained by scaling the minimax Bellman equation towards the continuous time limit, are intrinsically connected to stochastic processes and stochastic decision making (e.g., through the Feynman-Kac formula). Meanwhile, within the field of online learning, there is evidence (Freund, 2009; Harvey et al., 2020) suggesting a strong parallel relation between the discrete time adversarial problem and the continuous time stochastic problem. We are interested in digging deeper into the literature (especially those with a mathematical focus) and completing the missing pieces (if there are any). This may bring a better understanding to the "best-of-both-worlds" results in online learning (i.e., performance guarantees that are simultaneously optimal in stochastic and adversarial environments).

- The second part of this dissertation relies critically on complexity measures of function classes. This is a rich topic on its own, and often studied in related fields such as *nonparametric statistics* and *statistical learning theory*. Due to our currently limited scope, the connection to these related fields has not been investigated deeply in this dissertation. In this regard, it could be interesting to extend our results to other (possibly combinatorial) complexity measures, and further investigate the resulting algorithmic benefits.

- Finally, we believe that adaptive online learning has significant potential in many

real world problems, which has not been sufficiently explored yet. Addressing this gap between theory and practice requires a deeper dive into specific application domains and exploiting the corresponding domain structures. As a next step, we plan to investigate a diverse set of applications, including but not limited to finance, robotic control and physical science.

Bibliography

Abernethy, J., Agarwal, A., Bartlett, P. L., and Rakhlin, A. (2009). A stochastic view of optimal regret through minimax duality. In *Conference on Learning Theory*.

Abernethy, J., Bartlett, P. L., Rakhlin, A., and Tewari, A. (2008a). Optimal strategies and minimax lower bounds for online convex games. In *Conference on Learning Theory*, pages 415–423.

Abernethy, J., Hazan, E. E., and Rakhlin, A. (2008b). Competing in the dark: An efficient algorithm for bandit linear optimization. In *Conference on Learning Theory*, pages 263–273.

Agarwal, N., Bullins, B., Hazan, E., Kakade, S., and Singh, K. (2019). Online control with adversarial disturbances. In *International Conference on Machine Learning*, pages 111–119. PMLR.

Alquier, P. (2021). Non-exponentially weighted aggregation: Regret bounds for unbounded loss functions. In *International Conference on Machine Learning*, pages 207–218. PMLR.

Altschuler, J. and Talwar, K. (2018). Online learning over a finite action set with limited switching. In *Conference On Learning Theory*, pages 1569–1573. PMLR.

Anava, O., Hazan, E., and Mannor, S. (2015a). Online learning for adversaries with memory: price of past mistakes. *Advances in Neural Information Processing Systems*, 28.

Anava, O., Hazan, E., Mannor, S., and Shamir, O. (2013). Online learning for time series prediction. In *Conference on learning theory*, pages 172–184. PMLR.

Anava, O., Hazan, E., and Zeevi, A. (2015b). Online time series prediction with missing data. In *International conference on machine learning*, pages 2191–2199. PMLR.

Anava, O. and Mannor, S. (2016). Heteroscedastic sequences: beyond gaussianity. In *International Conference on Machine Learning*, pages 755–763. PMLR.

Andoni, A. and Panigrahy, R. (2013). A differential equations approach to optimizing regret trade-offs. *arXiv preprint arXiv:1305.1359*.

Andrew, L., Barman, S., Ligett, K., Lin, M., Meyerson, A., Roytman, A., and Wierman, A. (2013). A tale of two metrics: Simultaneous bounds on competitiveness and regret. In *Conference on Learning Theory*, pages 741–763. PMLR.

Arfken, G. B., Weber, H. J., and Harris, F. E. (2013). *Mathematical Methods for Physicists: A Comprehensive Guide*. Academic Press.

Azoury, K. S. and Warmuth, M. K. (2001). Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246.

Baby, D. and Wang, Y.-X. (2019). Online forecasting of total-variation-bounded sequences. *Advances in Neural Information Processing Systems*, 32.

Baby, D. and Wang, Y.-X. (2020). Adaptive online estimation of piecewise polynomial trends. *Advances in Neural Information Processing Systems*, 33:20462–20472.

Baby, D. and Wang, Y.-X. (2021). Optimal dynamic regret in exp-concave online learning. In *Conference on Learning Theory*, pages 359–409. PMLR.

Baby, D. and Wang, Y.-X. (2022). Optimal dynamic regret in proper online learning with strongly convex losses and beyond. In *International Conference on Artificial Intelligence and Statistics*, pages 1805–1845. PMLR.

Baby, D., Zhao, X., and Wang, Y.-X. (2021). An optimal reduction of tv-denoising to adaptive online learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2899–2907. PMLR.

Bayraktar, E., Ekren, I., and Zhang, X. (2020a). Finite-time 4-expert prediction problem. *Communications in Partial Differential Equations*, 45(7):714–757.

Bayraktar, E., Ekren, I., and Zhang, X. (2021). Prediction against a limited adversary. *Journal of Machine Learning Research*, 22(72):1–33.

Bayraktar, E., Ekren, I., and Zhang, Y. (2020b). On the asymptotic optimality of the comb strategy for prediction with expert advice. *The Annals of Applied Probability*, 30(6):2517–2546.

Bayraktar, E., Poor, H. V., and Zhang, X. (2020c). Malicious experts versus the multiplicative weights algorithm in online prediction. *IEEE Transactions on Information Theory*, 67(1):559–565.

Bertsekas, D. (2012). *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific.

Besbes, O., Gur, Y., and Zeevi, A. (2015). Non-stationary stochastic optimization. *Operations research*, 63(5):1227–1244.

Bhaskara, A., Cutkosky, A., Kumar, R., and Purohit, M. (2021). Power of hints for online learning with movement costs. In *International Conference on Artificial Intelligence and Statistics*, pages 2818–2826. PMLR.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Bubeck, S., Lee, Y. T., Li, Y., and Sellke, M. (2019). Competitively chasing convex bodies. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 861–868.

Cesa-Bianchi, N., Dekel, O., and Shamir, O. (2013). Online learning with switching costs and other adaptive adversaries. *Advances in Neural Information Processing Systems*, 26.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

Chaudhuri, K., Freund, Y., and Hsu, D. J. (2009). A parameter-free hedging algorithm. *Advances in neural information processing systems*, 22.

Chen, K., Langford, J., and Orabona, F. (2022). Better parameter-free stochastic optimization with ode updates for coin-betting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6239–6247.

Chen, L., Luo, H., and Wei, C.-Y. (2021). Impossible tuning made possible: A new expert algorithm and its applications. In *Conference on Learning Theory*, pages 1216–1259. PMLR.

Chen, L., Yu, Q., Lawrence, H., and Karbasi, A. (2020). Minimax regret of switching-constrained online convex optimization: No phase transition. *Advances in Neural Information Processing Systems*, 33:3477–3486.

Chen, N., Goel, G., and Wierman, A. (2018). Smoothed online convex optimization in high dimensions via online balanced descent. In *Conference On Learning Theory*, pages 1574–1594. PMLR.

Chen, X., Wang, Y., and Wang, Y.-X. (2019). Nonstationary stochastic optimization under $L_{p,q}$-variation measures. *Operations Research*, 67(6):1752–1765.

Chernov, A. and Vovk, V. (2010). Prediction with advice of unknown number of experts. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 117–125.

Chiang, C.-K., Yang, T., Lee, C.-J., Mahdavi, M., Lu, C.-J., Jin, R., and Zhu, S. (2012). Online optimization with gradual variations. In *Conference on Learning Theory*, pages 6.1–6.20.

Cohen, A., Daubechies, I., and Vial, P. (1993). Wavelets on the interval and fast wavelet transforms. *Applied and computational harmonic analysis*.

Cover, T. M. (1966). Behavior of sequential predictors of binary sequences. *Transactions of the Fourth Prague Conference on Information Theory, Statistical Decision Functions, Random Processes*, pages 263–272.

Cutkosky, A. (2019a). Artificial constraints and hints for unbounded online learning. In *Conference on Learning Theory*, pages 874–894. PMLR.

Cutkosky, A. (2019b). Combining online learning guarantees. In *Conference on Learning Theory*, pages 895–913. PMLR.

Cutkosky, A. (2020). Parameter-free, dynamic, and strongly-adaptive online learning. In *International Conference on Machine Learning*, pages 2250–2259. PMLR.

Cutkosky, A. and Orabona, F. (2018). Black-box reductions for parameter-free online learning in banach spaces. In *Conference On Learning Theory*, pages 1493–1529. PMLR.

Daniely, A., Gonen, A., and Shalev-Shwartz, S. (2015). Strongly adaptive online learning. In *International Conference on Machine Learning*, pages 1405–1411. PMLR.

Daniely, A. and Mansour, Y. (2019). Competitive ratio vs regret minimization: achieving the best of both worlds. In *Algorithmic Learning Theory*, pages 333–368. PMLR.

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.

Doob, J. L. (1984). *Classical potential theory and its probabilistic counterpart*. Springer.

Drenska, N. and Calder, J. (2022). Online prediction with history-dependent experts: the general case. *Communications on Pure and Applied Mathematics*.

Drenska, N. and Kohn, R. V. (2020a). A pde approach to the prediction of a binary sequence with advice from two history-dependent experts. *arXiv preprint arXiv:2007.12732*.

Drenska, N. and Kohn, R. V. (2020b). Prediction with expert advice: A PDE perspective. *Journal of Nonlinear Science*, 30(1):137–173.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Duchi, J. C., Shalev-Shwartz, S., Singer, Y., and Tewari, A. (2010). Composite objective mirror descent. In *Conference on learning theory*, pages 14–26.

Foster, D., Kale, S., and Karloff, H. (2016). Online sparse linear regression. In *Conference on Learning Theory*, pages 960–970. PMLR.

Foster, D. J., Rakhlin, A., and Sridharan, K. (2018). Online learning: Sufficient statistics and the Burkholder method. In *Conference On Learning Theory*, pages 3028–3064. PMLR.

Freund, Y. (2009). A method for hedging in continuous time. *arXiv preprint arXiv:0904.3356*.

Gaillard, P. and Gerchinovitz, S. (2015). A chaining algorithm for online nonparametric regression. In *Conference on Learning Theory*, pages 764–796. PMLR.

Gaillard, P. and Wintenberger, O. (2018). Efficient online algorithms for fast-rate regret bounds under sparsity. *Advances in Neural Information Processing Systems*, 31.

Gerchinovitz, S. (2013). Sparsity regret bounds for individual sequences in online linear regression. *The Journal of Machine Learning Research*, 14(1):729–769.

Gerchinovitz, S. and Yu, J. Y. (2014). Adaptive and optimal online linear regression on L_1-balls. *Theoretical Computer Science*, 519:4–28.

Geulen, S., Vöcking, B., and Winkler, M. (2010). Regret minimization for online buffering problems using the weighted majority algorithm. In *Conference on Learning Theory*, pages 132–143.

Goel, G., Lin, Y., Sun, H., and Wierman, A. (2019). Beyond online balanced descent: An optimal algorithm for smoothed online optimization. *Advances in Neural Information Processing Systems*, 32.

Gofer, E. (2014). Higher-order regret bounds with switching costs. In *Conference on Learning Theory*, pages 210–243. PMLR.

Gordon, R. D. (1941). Values of mills' ratio of area to bounding ordinate and of the normal probability integral for large values of the argument. *The Annals of Mathematical Statistics*, 12(3):364–366.

Gyorgy, A. and Szepesvári, C. (2016). Shifting regret, mirror descent, and matrices. In *International Conference on Machine Learning*, pages 2943–2951. PMLR.

Hall, E. C. and Willett, R. M. (2015). Online convex optimization in dynamic environments. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):647–662.

Harvey, N. J., Liaw, C., Perkins, E., and Randhawa, S. (2020). Optimal anytime regret with two experts. *arXiv preprint arXiv:2002.08994v2*.

Hazan, E. (2016). Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325.

Hazan, E., Lee, H., Singh, K., Zhang, C., and Zhang, Y. (2018). Spectral filtering for general linear dynamical systems. *Advances in Neural Information Processing Systems*, 31.

Hazan, E. and Seshadhri, C. (2009). Efficient learning algorithms for changing environments. In *International Conference on Machine Learning*, pages 393–400.

Herbster, M. and Warmuth, M. K. (2001). Tracking the best linear predictor. *The Journal of Machine Learning Research*, 1:281–309.

Hurley, N. and Rickard, S. (2009). Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741.

Jacobsen, A. and Cutkosky, A. (2022). Parameter-free mirror descent. In *Conference on Learning Theory*, pages 4160–4211. PMLR.

Jadbabaie, A., Rakhlin, A., Shahrampour, S., and Sridharan, K. (2015). Online optimization: Competing with dynamic comparators. In *Artificial Intelligence and Statistics*, pages 398–406. PMLR.

Johnstone, I. M. (2019). Gaussian estimation: Sequence and wavelet models. *Unpublished lecture notes*. `https://imjohnstone.su.domains/GE_09_16_19.pdf`.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

Jun, K.-S. and Orabona, F. (2019). Parameter-free locally differentially private stochastic subgradient descent. *arXiv preprint arXiv:1911.09564*.

Jun, K.-S., Orabona, F., Wright, S., and Willett, R. (2017). Improved strongly adaptive online learning using coin betting. In *Artificial Intelligence and Statistics*, pages 943–951. PMLR.

Kalai, A. and Vempala, S. (2005). Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307.

Kale, S. (2014). Open problem: Efficient online sparse regression. In *Conference on Learning Theory*, pages 1299–1301. PMLR.

Kale, S., Karnin, Z., Liang, T., and Pál, D. (2017). Adaptive feature selection: Computationally efficient online sparse linear regression under rip. In *International Conference on Machine Learning*, pages 1780–1788. PMLR.

Kapralov, M. and Panigrahy, R. (2011). Prediction strategies without loss. *Advances in Neural Information Processing Systems*, 24.

Klenke, A. (2013). *Probability theory: a comprehensive course*. Springer Science & Business Media.

Kobzar, V. A., Kohn, R. V., and Wang, Z. (2020a). New potential-based bounds for prediction with expert advice. In *Conference on Learning Theory*, pages 2370–2405. PMLR.

Kobzar, V. A., Kohn, R. V., and Wang, Z. (2020b). New potential-based bounds for the geometric-stopping version of prediction with expert advice. In *Mathematical and Scientific Machine Learning*, pages 537–554. PMLR.

Koolen, W. M. and Van Erven, T. (2015). Second-order quantile methods for experts and combinatorial games. In *Conference on Learning Theory*, pages 1155–1175. PMLR.

Korolev, V. and Shevtsova, I. (2012). An improvement of the berry–esseen inequality with applications to poisson and mixed poisson random sums. *Scandinavian Actuarial Journal*, 2012(2):81–105.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Kuznetsov, V. and Mohri, M. (2016). Time series prediction and online learning. In *Conference on Learning Theory*, pages 1190–1213. PMLR.

Langford, J., Li, L., and Zhang, T. (2009). Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(3).

Li, Y., Qu, G., and Li, N. (2020). Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit. *IEEE Transactions on Automatic Control*, 66(10):4761–4768.

Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm. *Information and computation*, 108(2):212–261.

Luo, H. and Schapire, R. E. (2015). Achieving all with no parameters: Adanormalhedge. In *Conference on Learning Theory*, pages 1286–1304. PMLR.

Mallat, S. (2008). *A Wavelet Tour of Signal Processing: The Sparse Way*. Academic Press.

McMahan, B. and Abernethy, J. (2013). Minimax optimal algorithms for unconstrained linear optimization. *Advances in Neural Information Processing Systems*, 26:2724–2732.

Mcmahan, B. and Streeter, M. (2012). No-regret algorithms for unconstrained online convex optimization. *Advances in neural information processing systems*, 25.

McMahan, H. B. and Orabona, F. (2014). Unconstrained online linear learning in hilbert spaces: Minimax algorithms and normal approximations. In *Conference on Learning Theory*, pages 1020–1039. PMLR.

Mhammedi, Z. and Koolen, W. M. (2020). Lipschitz and comparator-norm adaptivity in online learning. In *Conference on Learning Theory*, pages 2858–2887. PMLR.

Miranker, W. L. (1961). A well posed problem for the backward heat equation. *Proceedings of the American Mathematical Society*, 12(2):243–247.

Negrea, J., Bilodeau, B., Campolongo, N., Orabona, F., and Roy, D. (2021). Minimax optimal quantile and semi-adversarial regret via root-logarithmic regularizers. *Advances in Neural Information Processing Systems*, 34.

Orabona, F. (2013). Dimension-free exponentiated gradient. *Advances in Neural Information Processing Systems*, 26.

Orabona, F. (2019). A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*.

Orabona, F. and Pál, D. (2016). Coin betting and parameter-free online learning. *Advances in Neural Information Processing Systems*, 29.

Orabona, F. and Tommasi, T. (2017). Training deep networks without learning rates through coin betting. *Advances in Neural Information Processing Systems*, 30:2160–2170.

Payne, L. E. (1975). *Improperly posed problems in partial differential equations*. SIAM.

Portella, V. S., Liaw, C., and Harvey, N. J. (2022). Continuous prediction with experts' advice. *arXiv preprint arXiv:2206.00236*.

Price, E. (2021). Sparse recovery. In Roughgarden, T., editor, *Beyond the Worst-Case Analysis of Algorithms*, page 140–164. Cambridge University Press.

Rakhlin, A. and Sridharan, K. (2013). Online learning with predictable sequences. In *Conference on Learning Theory*, pages 993–1019. PMLR.

Rakhlin, A. and Sridharan, K. (2014a). Online non-parametric regression. In *Conference on Learning Theory*, pages 1232–1264. PMLR.

Rakhlin, A. and Sridharan, K. (2014b). Statistical learning and sequential prediction. *Unpublished lecture notes*. `http://www.mit.edu/~rakhlin/courses/stat928/stat928_notes.pdf`.

Rakhlin, S., Shamir, O., and Sridharan, K. (2012). Relax and randomize: From value to algorithms. *Advances in Neural Information Processing Systems*, 25:2141–2149.

Rockafellar, R. T. (2015). *Convex analysis*. Princeton university press.

Rokhlin, D. B. (2017). PDE approach to the problem of online prediction with expert advice: a construction of potential-based strategies. *arXiv preprint arXiv:1705.01091*.

Sellke, M. (2020). Chasing convex bodies optimally. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1509–1518. SIAM.

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

Shalev-Shwartz, S. and Tewari, A. (2011). Stochastic methods for l1-regularized loss minimization. *Journal of Machine Learning Research*, 12:1865–1892.

Sherman, U. and Koren, T. (2021). Lazy OCO: Online convex optimization on a switching budget. In *Conference on Learning Theory*, pages 3972–3988. PMLR.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Simchowitz, M. (2020). Making non-stochastic control (almost) as easy as stochastic. *Advances in Neural Information Processing Systems*, 33:18318–18329.

Simchowitz, M., Singh, K., and Hazan, E. (2020). Improper learning for non-stochastic control. In *Conference on Learning Theory*, pages 3320–3436. PMLR.

Steinhardt, J. and Liang, P. (2014). Adaptivity and optimism: An improved exponentiated gradient algorithm. In *International conference on machine learning*, pages 1593–1601. PMLR.

Streeter, M. and Mcmahan, B. (2012). No-regret algorithms for unconstrained online convex optimization. *Advances in Neural Information Processing Systems*, 25.

Streeter, M. and McMahan, H. B. (2010). Less regret via online conditioning. *arXiv preprint arXiv:1002.4862*.

van der Hoeven, D. (2019). User-specified local differential privacy in unconstrained adaptive online learning. *Advances in Neural Information Processing Systems*, 32.

Vovk, V. (2001). Competitive on-line statistics. *International Statistical Review*, 69(2):213–248.

Wang, G., Wan, Y., Yang, T., and Zhang, L. (2021). Online convex optimization with continuous switching constraint. *Advances in Neural Information Processing Systems*, 34.

Wang, Z. and Kohn, R. V. (2022). A new approach to drifting games, based on asymptotically optimal potentials. *arXiv preprint arXiv:2207.11405*.

Xiao, L. (2009). Dual averaging method for regularized stochastic learning and online optimization. *Advances in Neural Information Processing Systems*, 22.

Yang, T., Zhang, L., Jin, R., and Yi, J. (2016). Tracking slowly moving clairvoyant: Optimal dynamic regret of online learning with true and noisy gradient. In *International Conference on Machine Learning*, pages 449–457. PMLR.

Zhang, L., Jiang, W., Lu, S., and Yang, T. (2021). Revisiting smoothed online learning. *Advances in Neural Information Processing Systems*, 34.

Zhang, L., Lu, S., and Zhou, Z.-H. (2018a). Adaptive online learning in dynamic environments. *Advances in neural information processing systems*, 31.

Zhang, L., Yang, T., Yi, J., Jin, R., and Zhou, Z.-H. (2017). Improved dynamic regret for non-degenerate functions. *Advances in Neural Information Processing Systems*, 30.

Zhang, L., Yang, T., and Zhou, Z.-H. (2018b). Dynamic regret of strongly adaptive methods. In *International Conference on Machine Learning*, pages 5882–5891. PMLR.

Zhang, Z., Cutkosky, A., and Paschalidis, I. (2022a). Adversarial tracking control via strongly adaptive online learning with memory. In *International Conference on Artificial Intelligence and Statistics*, pages 8458–8492. PMLR.

Zhang, Z., Cutkosky, A., and Paschalidis, I. (2022b). PDE-based optimal strategy for unconstrained online learning. In *International Conference on Machine Learning*, pages 26085–26115. PMLR.

Zhang, Z., Cutkosky, A., and Paschalidis, I. C. (2023). Unconstrained dynamic regret via sparse coding. *arXiv preprint arXiv:2301.13349*.

Zhang, Z., Cutkosky, A., and Paschalidis, Y. (2022c). Optimal comparator adaptive online learning with switching cost. *Advances in Neural Information Processing Systems*, 35:23936–23950.

Zhu, K. (2014). *Two problems in applications of PDE*. PhD thesis, New York University. ProQuest Dissertations & Theses Global (Publication No. 3635320).

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*, pages 928–936.