POLITECNICO DI MILANO
Mathematical Engineering Master Degree
Dipartimento di Matematica

Online Gradient Descent for Online Portfolio Optimization with Transaction Costs

Artificial Intelligence and Robotic Laboratory
of Politecnico di Milano

Supervisor: Prof. Marcello Restelli

Co-supervisors: Francesco Trovò Ph.D., Edoardo Vittori,

Master thesis of:
Martino Bernasconi de Luca, 10465492

Academic Year 2018-2019

Contents

1 Introduction XI

2 Online Learning 1
  2.1 Online Learning . . . 2
    2.1.1 Regret and Experts . . . 3
    2.1.2 Existence of No-Regret Strategies . . . 5
  2.2 Experts . . . 7
    2.2.1 Uncountable Experts . . . 7
  2.3 Exp-Concave loss functions . . . 8
  2.4 Regret Minimization in Games . . . 10
    2.4.1 Mixed extension . . . 10
    2.4.2 MinMax Consistency . . . 11
  2.5 Online Convex Optimization for Regret Minimization . . . 13
    2.5.1 A General Algorithm for Online Convex Optimization 14
    2.5.2 Mirror Version of the Online Mirror Descent Algorithm 16
  2.6 From Online Learning to Statistical Learning . . . 18

3 Information, Prediction and Investing 23
  3.1 Probability assignment . . . 23
    3.1.1 Laplace Mixture Forecaster . . . 25
    3.1.2 Connection to Information Theory . . . 25
    3.1.3 Horse Races . . . 26
  3.2 From Horse Races to Online Portfolio Optimization . . . 28
    3.2.1 The Online Portfolio Optimization Model . . . 28
    3.2.2 Effectiveness of Constant Rebalancing Portfolios . . . 29

4 Problem Formulation 31
  4.1 Online Portfolio Optimization with Transaction Costs . . . 32
  4.2 Related Works in Online Learning and Switching Costs . . . 34

5 Algorithms for the Online Portfolio Optimization Problem 37
5.1 Algorithm with regret bound 38
5.1.1 Universal Portfolios 38
5.1.2 Exponential Gradient 39
5.1.3 Online Newton Step 40
5.2 Algorithm with total regret bound 41
5.2.1 Online Lazy Updates 42
5.2.2 Implementation of Online Lazy Update 44
5.3 Heuristic Algorithms without Regret Bound 45

6 Online Gradient Descent for Online Portfolio Optimization with Transaction Costs 47
6.1 Using OGD for Portfolio Optimization 48
6.1.1 The OGD Algorithm 48
6.2 Regret Analysis 50
6.2.1 OGD Regret on the Wealth 50
6.2.2 OGD Regret on the Costs 51
6.2.3 Total Regret 52
6.3 Implementation of the Online Gradient Descent Algorithm . . 53
6.4 Discussion on the Regret Bound 54

7 Numerical Experiments 55
7.0.1 Results on the NYSE(O) dataset 56
7.0.2 Results on the TSE and SP500 dataset 59
7.0.3 Result on Custom Dataset 60

Bibliography 63

List of Figures

2.1 Online Learning with Expert Advice as Multi Agent-Environment interaction. 3
2.2 Rock Paper Scissor Dynamics in the Δ_2 simplex, generated by the Exponentially Weighted Majority algorithm against an adversarial opponent. 13
2.3 Contour lines of $f(x, y) = \max\{x^2 + (y - 1)^2, x^2 + (y + 1)^2\}$, together with sub-gradient directions from $(1, 0)$. All sub-gradient will point to increasing values of the function 17
2.4 Fenchel Conjugate Function. 18
2.5 Online Mirror Descent as Mirror Updates. 19

6.1 Online Gradient Descent. 48
6.2 Generalized Pythagorean Theorem. 49

7.1 Wealth $W_T^C(\mathcal{A})$ on two runs of the NYSE(O) for $\gamma = 0$ (a), and $\gamma = 0.001$ (b). 57
7.2 Average APY computed on the wealth W_T^C assuming the costs given by $C_T(\mathcal{A})$ for the NYSE(O) dataset. 57
7.3 Average APY computed on the wealth $\tilde{W}_T(\mathcal{A})$ assuming the costs given by Equation (4.2), for the NYSE(O) dataset. . . . 58
7.4 Average costs $C_T(\mathcal{A})$ with $\gamma = 1$, for the NYSE(O) dataset. . 58
7.5 Average APY computed on the wealth W_T^C assuming the costs given by $C_T(\mathcal{A})$ for the TSE dataset. 59
7.6 Average APY computed on the wealth W_T^C assuming the costs given by $C_T(\mathcal{A})$ for the SP500 dataset. 59
7.7 Grid search for the parameters of ONS (a) and OLU (b) on the validation set of the custom dataset. 61
7.8 Wealth W_t achieved on the custom dataset described in table 7.2, with $\gamma = 0.001$. 61

List of Tables

7.1 Description of the main datasets used commonly in the Online Portfolio Optimization literature. 55
7.2 Detailed description of the custom dataset. 60

List of Algorithms

1 OMD for Online Convex Optimization 16
2 Alternating Direction Method of Multipliers 45
3 OGD in Online Portfolio Optimization with Transaction Costs 47
4 Near Linear Time Projection Onto The Probability Simplex . 53

Chapter 1

Introduction

Classical investment techniques for the portfolio management problem derive from the knowledge of the statistical distribution of the assets return. Then, once the statistical model has been chosen, the problem get solved by optimizing the expected value of the utility of some random variable (usually accounting for the trade-off between risk and return), that describes the value of the portfolio in some fixed time in the future. This line of thinking has been proposed and sustained by Markovitz, Samuelson, Fama ecc... , and it is now called Modern Portfolio Theory (MPT).

This approach is known to be very susceptible to the errors in the modelling of the random variable that model the asset return. In fact, it is known that the markets have a non stationary behaviour, which means that every statistical assumption is ephemeral and unreliable. and they are usually referred to to backward looking, i.e. that they optimize

A different approach has been originated from the fields of information theory at the Bell Labs in the 1950, from the works of Shannon, Kelly and Cover. This methods were first included in the classical portfolio theory framework, under the name of Capital Growth Theory (CGT) [Hakansson et al., 1995] and then got included in the machine learning literature under the framework of Online Game Playing. Only recently this field has been taken into the Online Optimization This formulation has very interesting properties such as stability in a game theory fashion (equilibrium) and robustness versus adversarial manipulation.

One of the strongest points in favor of this techniques are the strong theoretical guarantees that algorithms developed under this framework can give. This guarantees come from the game theory concept of Regret, which is a form of dissatisfaction originated from having taken an action, instead of another action.

Principal in this thesis will be the extension of the modelling of the financial applications of this methodologies to the presence of transaction costs and to provide strong theoretical assurance even in the presence of transaction costs. In fact in many financial situations, transaction costs are not modelled and

Chapter 2

Online Learning

Online Learning is a theoretical framework to formalize a sequential decision problem in which an agent has to take consecutive actions in an environment. Every time the agent takes an action, the environment returns a loss signal (or reward depending on the convention on the sign). This framework is similar to other sequential decision problems such as Reinforcement Learning [Sutton et al.,], with the main difference that the loss function is decided by an adversary which has complete knowledge of your strategy in advance, rather that be described by a stochastic probability kernel. The purpose of this section is to present the general framework of Online Game Playing and to introduce the notation necessary for the development of the theory for Online Portfolio Optimization. We will define formally the framework of Online Learning with Expert Advice, which is one the most studied framework of Online Learning, due to its ability to include many other frameworks, such as Multi Armed Bandit [Bubeck et al., 2012] or Online Convex Optimization [Hazan et al., 2016]. Then we will present the concept of *regret* and present the relationship of Online Learning to classical repeated games, a classical framework coming from the field of Game Theory. We are interested in this framework in order to model repeated investments in this framework. Modern finance has more and more the need for a Game Theoretic approach, this is evident when one looks at the field of *On venue Market Making*, that can be modeled naturally as a repeated game, or in merger and acquisition that can be modeled as a normal form game [Jiang et al., 2016]. Finally we will introduce Online Convex Optimization as a special case of Online Learning with expert advice and its interesting relationship to theoretical statistical learning. The choice of this path, from Online Learning to Online Convex Optimization, has been done to show how general and powerful Online Learning is in its simplicity, and

F: è sbagliato.

F: direi anche che ci immaginiamo che esitanto degli avversari che possono giocare strategicamente... M: non pensavo fosse un concern i veri avversari.

why Online Convex Optimization is the most suitable framework to present our contribution to Online Portfolio Selection, a framework that will be presented in Chapter 3.

In fact, even if we will focus on the portfolio problem, the apparently simple formulation of this framework is capable to encompass many other applications and problems, such as network routing [Belmega et al., 2018] and dark pool order allocation [Agarwal et al., 2010]. A thorough dissertation of the techniques that have been developed in the field of Online Learning can be found in [Cesa-Bianchi and Lugosi, 2006].

2.1 Online Learning

Definition 2.1.1. *(Online Game Playing). Let $\mathcal{Y}$ be the outcome space, $\mathcal{D}$ the prediction space and $f : \mathcal{D} \times \mathcal{Y} \to \mathbb{R}$ is a loss function, an Online Game is the following sequential game played by the forecaster $\mathcal{A}$ and the environment:*

For each round $t \in \mathbb{N}$

1. *The learner $\mathcal{A}$ chooses an element of the decision space $x_t \in \mathcal{D}$.*
2. *The environment chooses the element $y_t \in \mathcal{Y}$, and subsequently determines the loss function $f(\cdot, y_t)$.*
3. *The agent $\mathcal{A}$ incurs in a loss $f(x_t, y_t)$.*
4. *The agent updates its cumulative losses $L_t = L_{t-1} + f(x_t, y_t)$ with $L_0 = 0$.*

F: algorithm

In Online Learning an agent $\mathcal{A}$ has to guess the outcome y_t based on the past sequence $y_1, y_2, \dots, y_{t-1}$ of events that are in the outcome space $\mathcal{Y}$, at each time step the agent will play (sometimes we will also say *predict*) x_t, that is an element of the prediction space $\mathcal{D}$, and the environment will choose a loss function $f(\cdot, y_t)$ by determining the outcome y_t. The agent $\mathcal{A}$ is essentially the identification of the functions that map the history of past outcomes to the new prediction:

$$\mathcal{A} \equiv \{h_{t-1} := (y_1, \dots, y_{t-1}) \longmapsto x_t\}_{t \geq 1}.$$

The simplest case is for $\mathcal{Y} = \mathcal{D}$ and both of finite cardinality, meaning that there are only a finite number of actions that the agent $\mathcal{A}$ can choose from. We will sometimes refer to the environment defined in Section 2.1.1 as “adversarial”, since no stochastic characterization is given to the outcome sequence y_t and the analysis of the regret is done assuming a worst case scenario. Since the adversary knows the prediction x_t, before deciding the outcome y_t, designing an algorithm which tries to minimize the loss is a hopeless task and so we have to set an easier scope. In Section 2.1.1 we will also present the counterexample to why the absolute minimization of the loss is an hopeless task, and present the adapt minimal framework to successful Online Learning in Adversarial Environment.

2.1.1 Regret and Experts

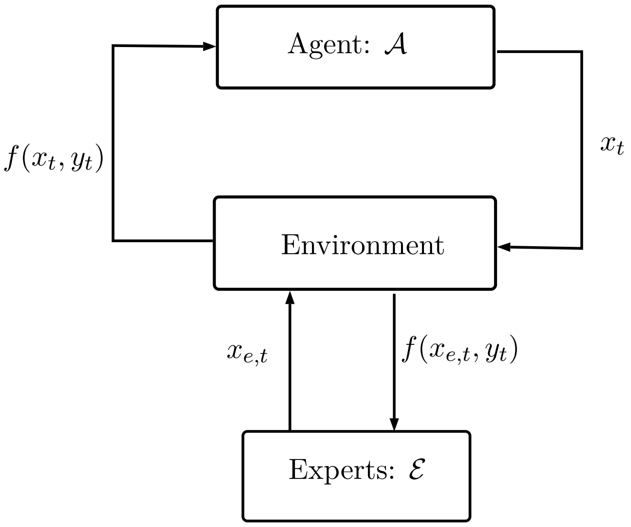


Figure 2.1: Online Learning with Expert Advice as Multi Agent-Environment interaction.

We stated that the objective of absolute loss minimization is hopeless in an adversarial framework, as the adversary can always choose the outcome y_t that maximizes the loss $f(x, y_t)$ regardless of the decision $x \in \mathcal{D}$ taken by the learner. More formally, assume $\mathcal{D}$ to be the space of binary outcomes, *i.e.* $|\mathcal{D}| = 2$, and that f is the absolute loss $f(x, y) = |x - y|$. Since the adversary plays after the learner $\mathcal{A}$, it can make the loss of the learner $L_T = T$ by choosing $y = 1 - x$ as outcome the bit non predicted by the learner, making $f(x, y) = 1$ at each time step. Notice that no assumption has been made on the strategy followed by the learner $\mathcal{A}$. From this example it is clear that the learner has to set a less ambitious goal.

We do so by extending the theoretical formulation in Section 2.1 by including a set $\mathcal{E}$ of other players, this setting is called *prediction with expert* *advice.* At each time step of the prediction game, each expert $e \in \mathcal{E}$, predicts an element $x_{e,t} \in \mathcal{D}$, and incurs in a loss $f(x_{e,t}, y_t)$, just as the agent $\mathcal{A}$, creating a general multi-agent interaction as in Figure 2.1. The goal of the learner is to obtain small losses with respect to the best expert in the class $\mathcal{E}$. This concept is captured by the definition of regret. Formally, we define the regret $R_{e,T}$ for the agent $\mathcal{A}$ with respect to expert $e \in \mathcal{E}$ (assumed finite for the moment) as follows:

$$R_{e,T} = L_T - L_{e,T}. \tag{2.1}$$

The regret observed by the agent $\mathcal{A}$ with respect to the entire class of experts $\mathcal{E}$ is defined as:

$$R_T = \sup_{e \in \mathcal{E}} R_{e,T} = L_T - \inf_{e \in \mathcal{E}} L_{e,T}. \tag{2.2}$$

The task agent $\mathcal{A}$ is to find a sequence x_t, function of the information obtained up to the time t in order to obtain small regret y_T with respect to any sequence $y_1, y_2, \ldots$ chosen by the environment.

In particular we aim to achieve sub-linear regret $R_T = o(T)$, meaning that the per-round regret R_T/T will asymptotically vanish:

$$R_T = o(T) \implies \lim_{T \to \infty} \frac{R_T}{T} = 0, \tag{2.3}$$

where $o(T)$ is the space of sub-linear affine functions. A strategy $\mathcal{A}$ that attains sub-linear regret is called *Hannan-Consistent* [Hannan, 1957].

The regret is a measure of the distance between our online performance and the best offline (in retrospect) performance among the expert class $\mathcal{E}$, this is also called *external regret* since it is compared to the external set of experts $\mathcal{E}$. A surprising fact is even that such algorithms do even exist. Indeed a first result is that *in general* there are no Hannan-consistent strategies, and just introducing the concept of regret is not enough by itself for successful Online Learning.

A first simple counterexample can be found in [Cover, 1966]. If the decision space $\mathcal{D}$ is finite then there exists a sequence of loss function such that $R_T = \Omega(T)$. Again take $\mathcal{D}$ as a space of binary outcomes, absolute loss as $f(x, y) = |x - y|$, and the class of experts is composed by two experts, one predicting always 0 and the other always 1. Taking T odd, we have that the loss of the best expert is $L_{e,T} < \frac{T}{2}$, and we have already shown that the adversary can make the loss of the learner $L_T = T$. It is now evident that the regret is $R_T > T - \frac{T}{2}$, which does not allow $R_T/T \to 0$. This argument is easily extended in the case of any finite decision space $\mathcal{D}$.

To achieve sub-linear regret, the learner has to randomize its predictions, at each turn t, the agent choose a probability distribution on the decision space and plays x_t according to this distribution. Clearly the adversary has knowledge of the probability distribution of the learner $\mathcal{A}$, but has no knowledge of the random seed used by the agent $\mathcal{A}$, *i.e.* does not know the actual decision sampled from the distribution held by the agent. If the original decision space was $\mathcal{D}$ with $|\mathcal{D}| = N$ after the randomization of the decision, we effectively transformed the decision space $\mathcal{D}$ into the $\Delta_{N-1} \in \mathbb{R}^N$ probability simplex. By doing so we are formally extending the game into its mixed extension, as will be discussed further in Section 2.4. It can be viewed also as a *covexification* of the domain, pointing to the undeniably necessity of convex geometry in this context, that will be discussed in Section 2.5. Therefore, from now on the domain $\mathcal{D}$ will be convex, either by the problem specification or by randomized convexification if the problem has discrete decision space.

2.1.2 Existence of No-Regret Strategies

In this section we will show the existence of Hannan-consistent strategies in the case of finite experts and provide a general form to generate sub-linear regret strategies. The general idea with a finite class of experts is given by the Weighted Average Forecaster, which implements a the natural idea of playing as the weighted average of the experts predictions:

Definition 2.1.2. *(Weighted Average Forecaster). For a finite class of experts $\mathcal{E} = \{E_1, \dots, E_N\}$, the weighted average prediction is defined as*

$$x_t = \frac{\sum_{i=1}^{N} w_{i,t-1} x_{i,t}}{\sum_{i=1}^{N} w_{i,t-1}}, \tag{2.4}$$

where $w_{i,t-1} > 0$ and $x_{i,t}$ is the prediction of expert $E_i \in \mathcal{E}$ at round t.

Since $\mathcal{D}$ is convex we have that $x_t \in \mathcal{D}$. Then it is natural to assume that the weights are a function of the cumulated regret suffered by the agent with respect to the experts, and also that the change in weight is proportional to the change in a potential function. We can generalize the simple weighted average prediction in Equation (2.1.2) in the following general form, introduced in [Cesa-Bianchi and Lugosi, 2003]:

$$x_t = \frac{\sum_{i=1}^{N} \partial_i \Phi(\mathbf{R}_{t-1}) x_{t,i}}{\sum_{i=1}^{N} \partial_i \Phi(\mathbf{R}_{t-1})}, \tag{2.5}$$

where $\Phi(\mathbf{u}) = \varphi\left(\sum_{i=1}^{N} \phi(u_i)\right)$ is a function $\Phi : \mathbb{R}^N \to \mathbb{R}^+$ defined through two increasing functions $\phi, \varphi : \mathbb{R} \to \mathbb{R}^+$, $\varphi, \phi \in \mathcal{C}^2(\mathbb{R})$ concave and convex, respectively and $\mathbf{R}_T = (R_{1,T}, \dots, R_{N,T})$. By specializing the two functions φ, ϕ we can derive most of the algorithms for dealing with prediction under expert advice. The reasons behind the general form of Equation (2.5) and an extended discussion can be found in [Hart and Mas-Colell, 2001], [Cesa-Bianchi and Lugosi, 2003] and [Blackwell et al., 1956], but the general idea is that the form of Equation (2.5) has the following property:

Theorem 2.1.1. *[Cesa-Bianchi and Lugosi, 2003] If x_t is given by Equation (2.5) and the loss $f(\cdot, y)$ is convex in the first argument then the instantaneous weighted regret satisfies:*

$$\sup_{y_t \in \mathcal{Y}} \sum_{i=1}^{N} [f(x_t, y_t) - f(x_{i,t}, y_t)] \partial_i \Phi(\mathbf{R}_{t-1}) \leq 0.$$

Proof. By convexity of $f(\cdot, y_t)$ we have that

$$f(x_t, y_t) \leq \frac{\sum_{i=1}^{N} \partial_i \Phi(\mathbf{R}_{t-1}) f(x_{i,t}, R_t)}{\sum_{i=1}^{N} \partial_i \Phi(\mathbf{R}_{t-1})}, \forall y_t \in \mathcal{Y}. \tag{2.6}$$

And since $\Phi(\mathbf{x}) = \varphi\left(\sum_{i=1}^{N} \phi(x_i)\right)$ we have that

$$\partial_i \Phi(\mathbf{x}) = \varphi'\left(\sum_{i=1}^{N} \phi(x_i)\right) \phi'(x_i) \geq 0.$$

Hence we can rearrange the terms in Equation (2.6) to obtain the statement.

□

Note that fixing the structure for the weights as in Equation (2.5) we have that $w_{t,i} \propto \phi'(R_{i,t})$ is an increasing function in $R_{i,t}$ (since ϕ is convex and increasing) that essentially states that we are increasing the probability of playing actions on which we saw high regret $R_{i,t}$.

check!!!

Definition 2.1.3. *(Exponentially Weighted Algorithm) The Exponentially weighted algorithm is obtained from Eqaution* (2.1.2) *by defining:*

$$w_{i,t-1} = e^{\eta y_{i,t-1}} / \sum_{j=1}^{N} e^{\eta y_{j,t-1}}. \tag{2.7}$$

Note that, the exponentially weighted algorithm is also Equation (2.5) where we defined $\varphi(x) = \frac{1}{\eta} ln(x)$ and $\phi(x) = e^{\eta x}$ giving weights defined in Equation (2.7).

It can be shown ([Cesa-Bianchi and Lugosi, 2006] Theorem 2.2) that the algorithm defined by the update rule in Equation (2.1.3), and for a convex loss function $f(\cdot, y_t)$, gives the following guarantee on the regret:

$$R_T \leq \frac{log(N)}{\eta} + \frac{T\eta}{8}. \tag{2.8}$$

By choosing $\eta = O\left(\sqrt{\frac{1}{T}}\right)$ we obtain a sub-linear regret $R_T = \mathcal{O}(\sqrt{T})$.

2.2 Experts

The theoretical framework described in Section 2.1 is very general and most suited for a game theory analysis of the problem. This helps us describe many other frameworks, such as Online Optimization [Hazan et al., 2016], or Multi Armed Bandit [Bubeck et al., 2012] as embedded into a Game Playing framework with expert advice. It can then be specialized by fixing many elements of the definition, in order to be applied to the specific problem we are willing to solve. For instance, the class of experts $\mathcal{E}$ is most of the time completely fictitious, meaning that the experts are not real players of the game, but most of the time they are *simulable* meaning that the agent $\mathcal{A}$ is able to compute $x_{e,t}$ for each expert $e \in \mathcal{E}$ and most of the times the class of expert is very limited in its actions, *e.g.*, $\mathcal{E}$ is the class of experts for which $x_{e,t}$ is constant in t. In this case, which is the most studied class of experts, we are basically just comparing our learner $\mathcal{A}$ to the best fixed action x^* in hindsight. This is a clairvoyant strategy that attains the minimum cumulative loss over the entire length of the game T.

2.2.1 Uncountable Experts

In the case of uncountable experts the Exponentially Averaged Prediction cannot be applied directly, but can be extended to a continuous mixture of experts predictions. More specifically we need the case of the class $\mathcal{E}$ being generated by a convex hull of a finite number of a base class of experts, $\mathcal{E}_N$. With continuous class of experts $\mathcal{E}$ defined in this way, the regret definition becomes:

$$R_T = \sup_{\mathbf{q}\in\Delta_{N-1}} R_{\mathbf{q},T} := L_T - \inf_{\mathbf{q}\in\Delta_{N-1}} L_{\mathbf{q},T}, \tag{2.9}$$

where $\Delta_{N-1} \subset \mathbb{R}^N$ is the N-simplex, and

$$L_{\mathbf{q},T} = \sum_{t=1}^{T} f(\langle \mathbf{q}, \mathbf{x}_{e,t}\rangle, y_t),$$

where $\mathbf{x}_{e,t} = (x_{1,t}, \dots, x_{N,t}) \in \mathbb{R}^N$ is the vector of expert predictions at time t.

2.3 Exp-Concave loss functions

Very important for the study of Portfolio Optimization is the exp-concave class of loss functions. The reason is that the natural loss function used in the Online Portfolio Optimization framework, is 1 exp-concave, as we shall see in Chapter 3.

Definition 2.3.1. *(Exp-concave function). $g(x)$ is said ν exp-concave if $e^{-\nu g(x)}$ is concave.*

When speaking about loss functions we are interested in concavity of the function in its first argument. Therefore we will say that a loss function f is ν exp-concave if $f(\cdot, y)$ is ν exp-concave $\forall y \in \mathcal{Y}$.

Theorem 2.3.1. *([Cesa-Bianchi and Lugosi, 2006] Theorem 3.2) The Exponentially Weighted Average forecaster, for ν-exp concave loss functions has the following property taking $\eta = \nu$:*

$$\Phi(\mathbf{R}_T) \leq \Phi(\mathbf{R}_0),$$

where $\Phi(x) = \varphi\left(\sum_{i=1}^{N} \phi(x_i)\right)$ is chosen as $\varphi(x) = \frac{1}{\nu} log(x)$ and $\phi(x) = e^{\nu x}$.

Proof. The weights are given by $w_{i,t-1} = e^{\nu y_{i,t-1}} / \sum_{j=1}^{N} e^{\nu y_{j,t-1}}$. By exp-concavity we have that

$$e^{-\nu f(x_t,y_t)} = exp\left\{-\nu f\left(\frac{\sum\limits_{i=1}^{N} w_{i,t-1}x_{i,t}}{\sum\limits_{i=1}^{N} w_{i,t-1}}, y_t\right)\right\} \geq \frac{\sum\limits_{i=1}^{N} w_{i,t-1}e^{-\nu f(x_{i,t},y_t)}}{\sum\limits_{i=1}^{N} w_{i,t-1}}. \tag{2.10}$$

This can be rewritten as

$$\sum_{i=1}^{N} e^{\nu y_{i,t-1}} e^{\nu[f(x_t,y_t)-f(x_{i,t},y_t)]} \leq \sum_{i=1}^{N} e^{\nu y_{i,t-1}}. \tag{2.11}$$

Applying $\varphi(x) = \frac{1}{\nu}log(x)$ to both sides of Equation (2.11) we obtain that

$$\Phi(\mathbf{R}_t) \leq \Phi(\mathbf{R}_{t-1}),$$

that proves the thesis. $\square$

The case of exp-concave functions is very special, since we can obtain Theorem 2.3.1 that can be used to prove regret bounds very easily as:

$$R_T \leq \frac{1}{\eta}\log\left(\sum_{i=1}^{N} e^{\nu R_{j,T}}\right) = \Phi(\mathbf{R}_T) \leq \Phi(\mathbf{R}_0) = \frac{logN}{\eta}. \tag{2.12}$$

The case of exp-concave losses is also useful for the case of uncountable experts sketched in Section 2.2.1. This formulation will be of central importance for the portfolio optimization problem.

It is natural to extend the Exponential Weighted Majority algorithm described by Equation (2.1.2) into the case of uncountable expert class $\mathcal{E}$ generated by the convex hull over the countable class $\mathcal{E}_N$, by:

$$x_t = \frac{\int\limits_{\Delta_{N-1}} w_{\mathbf{q},t-1}\langle \mathbf{q}, \mathbf{x}_{e,t}\rangle d\mathbf{q}}{\int\limits_{\Delta_{N-1}} w_{\mathbf{q},t-1} d\mathbf{q}}. \tag{2.13}$$

Theorem 2.3.2. *(Mixture forecaster for exp-concave losses) ([Cesa-Bianchi and Lugosi, 2006] Theorem 3.3).* *Choosing* $w_{\mathbf{q},t-1} = exp\left\{-\nu \sum\limits_{s=1}^{t-1} f(\langle \mathbf{q}, \mathbf{x}_{e,t}\rangle, y_s)\right\}$ *in Equation* (2.13), *for a bounded* ν*-exp concave loss function* $f(\cdot, y)$, *we obtain*

$$R_T \leq \frac{N}{\nu}\left(log\left(\frac{\nu T}{N}\right) + 1\right).$$

Even in the case of uncountable many experts, exp-concavity of the loss function gives a better convergence rate of $\mathcal{O}(\log T)$ then the exponentially weighted algorithm in Equation (2.8), which is $\mathcal{O}(\sqrt{T})$.

2.4 Regret Minimization in Games

In this section we explore the connection of the framework of Section 2.1 into a more classical repeated game framework. In the previous section we looked at the adversary as a black box, without any specific model in mind. The reason of this chapter is to clarify its role as a player in the game and to show the game theoretical properties of Hannan-consistent agents. Since in Online Learning the convention is to speak about losses, we shall speak about losses (players are minimizing) also in the classical definitions of game theory instead of payoffs (players are maximizing).

Definition 2.4.1. *(Strategic Form K-Player Game). A Strategic form K-player game is t a tuple $\langle \mathcal{K}, \{X_i\}_{i\in\mathcal{K}}, \{l_i\}_{i\in\mathcal{K}} \rangle$ where*

1. $\mathcal{K} = \{1, \dots, K\}$ *is the finite set of players.*
2. X_i *is the set of actions available to player* $i \in \mathcal{K}$*.*
3. $l_i : \bigotimes_{k=1}^{K} X_i \to \mathbb{R}$ *is the loss observed by player* $i \in \mathcal{K}$*.*

The game is called finite if $|X_i| < +\infty$ *for all* $i \in \mathcal{K}$*.*

2.4.1 Mixed extension

In Section 2.1 we saw that it is impossible to obtain sub-linear regret in adversarial environment with finite decision space $\mathcal{D}$. A first step to solve this has been the *randomized convexification* technique, where finite action spaces are extended into convex sets, given by their probability simplex. Losses are to be interpreted as expected losses when the mixed extension is applied to the formal game. More formally:

Definition 2.4.2. *(Mixed-extension for finite games). A finite game $\langle \mathcal{K}, \{X_i\}_{i\in\mathcal{K}}, \{l_i\}_{i\in\mathcal{K}} \rangle$ can be extended into the game $\langle \mathcal{K}, \{\tilde{X}_i\}_{i\in\mathcal{K}}, \{\tilde{l}_i\}_{i\in\mathcal{K}} \rangle$*

1. $\tilde{X}_i = \Delta_{|X_i|-1} \subset \mathbb{R}^{|X_i|}$ *for all* $i \in \mathcal{K}$
2. $\tilde{l} : \bigotimes \tilde{X}_i \to \mathbb{R}$ *is defined as*

$$\tilde{l}(x_1, \dots, x_K) = \sum_{i_1=1}^{N} \cdots \sum_{i_K=1}^{N} p_{i_1} \dots p_{i_K} l(i_1, \dots, i_K).$$

Due to the impossibility result of Cover [Cover, 1966], we have to work with the mixed extension formulation of the game. So from now on we take this step implicitly. The taxonomy of game definition is quite extended and complex, we will focus on non-cooperative games [Nash, 1951] since they are closely related to the setting tacked in the Online Learing field. More specifically, we will need the model for *Zero Sum Game.*

Definition 2.4.3. *(2-Player Zero-Sum Game). A Zero Sum game is a tuple* $\langle \{X_1, X_2\}, l : X_1 \times X_2 \to \mathbb{R} \rangle$*. As in Definition 2.4.1* X_1, X_2 *are the action spaces for Player* 1 *(row player) and Player 2 (columns player) respectively and* $l(x_1, x_2)$ *for* $x_i, x_2 \in X_1 \times X_2$*, represents the losses for Player* 1 *and profits for player* 2*.*

If this game is played for T turns, we can call it a repeated game, and the losses for each player will be $L_1^{(T)} = \sum_{t=1}^{T} l_i \left(x_i^{(t)}, x_2^{(t)} \right)$ and $L_2^{(T)} = -L_1^{(T)}$.

2.4.2 MinMax Consistency

The field that tries to answer to questions of what guarantees do Hannan-consistent strategies bring to the game theoretical formulation of the problem is referred in literature as *Learning in Games.* For formal games we can define the *values* for the game as:

$$V_1 = \inf_{x_1 \in X_1} \sup_{x_2 \in X_2} l(x_1, x_2), \tag{2.14}$$

$$V_2 = \sup_{x_2 \in X_2} \inf_{x_1 \in X_1} l(x_1, x_2). \tag{2.15}$$

These are the values that the players can guarantees themselves, meaning that no matter the strategy of the columns player, the row player could guarantee himself a loss of at maximum V_1, the converse holds for the row player. It can be interpreted as the minimum loss (best payoff) that player could achieve if we know that the other player would play adversarially. It is clear that $V_2 \leq V_1$. In the case the zero sum-game is a mixed extension of a finite game, then the Von Neumann theorem states that $V_1 = V_2$.

Now we will embed the framework of Online Game Playing of Section 2.1 in a two player zero sum game. Online Learning is a special form of Zero Sum Game (possibly considering its mixed extension described in Definition 2.4.1) where $X_1 \equiv \mathcal{D}$ and $X_2 \equiv \mathcal{Y}$. The loss function $l : X_1 \times X_2 \to \mathbb{R}$ can be identified by the loss $f : \mathcal{D} \times \mathcal{Y} \to \mathbb{R}$ of the Online Learning Agent $\mathcal{A}$. Now we will explore interesting properties of Hannan-consistent strategies. A surprising fact is that if the row player plays accordingly to a Hannan-consistent strategy then it achieves the value of the game V_1.

Theorem 2.4.1. *Hannan-consistent agents in Online Game Playing reach asymptotically the minmax value of the one shot game, formally:*

$$\limsup_{T\to+\infty} \frac{1}{T}\sum_{t=1}^{T} f(x_t, y_t) \le V_1.$$

Proof. Let us suppose that player 1 plays an Hannan-consisten strategy and that $y_1, y_2, \ldots \subset \mathcal{Y}$ is a generic sequence played by the columns player.

$$\limsup_{T\to+\infty} \frac{R_T}{T} \le 0, \tag{2.16}$$

that can be translated into

$$\limsup_{T\to+\infty} \frac{1}{T}\sum_{t=1}^{T} f(x_t, y_t) \le \limsup_{T\to+\infty} \frac{1}{T} \inf_{x\in\mathcal{D}} \sum_{t=1}^{T} f(x, y_t). \tag{2.17}$$

Let us define $\hat{y}_T$ as the empirical distribution played by player 2 up to T:

$$\hat{y}_T = \frac{1}{T}\sum_{t=1}^{T} y_t.$$

By Equation (2.17) we just need to show that $\frac{1}{T} \inf_{x\in\mathcal{D}} \sum_{t=1}^{T} f(x, y_t) \le V_1$.

That follow easily:

$$\inf_{x\in\mathcal{D}} \frac{1}{T}\sum_{t=1}^{T} f(x, y_t) = \inf_{x\in\mathcal{D}} f(x, y_T) \le \sup_{y\in\mathcal{Y}} \inf_{x\in\mathcal{D}} f(x, y) \le V_1. \tag{2.18}$$

□

We showed that regardless of the strategy of player 2, a player using an Hannan-consistent strategy achieves lower losses than the value of the game V_1. Clearly using an Hannan-consistent strategy means that if player 2 was not adversarial, then player 1 could potentially earn a significantly higher average payoff than the value V of the game. By symmetry, if both players play an Hannan-consistent strategy then they will asymptotically reach the value of the game $V = V_1 = V_2$.

In Figure 2.2 we present the path of the randomization probabilities of the Rock Paper Scissor game represented in the Δ_2 simplex, obtained by the Exponentially Weighted Majority algorithm against an adversarial opponent which plays the best response at each turn, knowing the probabilities of the learner. Note that the algorithm learns to play the optimal strategy which is the randomization probabilities of $(1/3, 1/3, 1/3)$ over the action space. In general the dynamic of policies learned with Hanna consistent strategies are very complex and not well understood [Bailey and Piliouras, 2018].

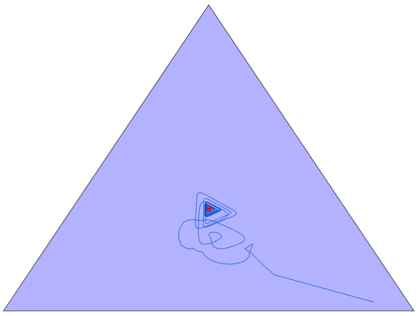


Figure 2.2: Rock Paper Scissor Dynamics in the Δ_2 simplex, generated by the Exponentially Weighted Majority algorithm against an adversarial opponent.

2.5 Online Convex Optimization for Regret Minimization

Let us compare this framework to an apparently unrelated problem, namely optimization, that will turn out to be the most suited framework to embed the Online Portfolio Optimization Problem. In online optimization an agent $\mathcal{A}$ is designed to optimize a sequence of functions $f_t(x)$ where usually $f_t : \mathcal{D} \to \mathbb{R}$ is a real valued function from the set $\mathcal{D} \subset \mathbb{R}^n$. As a remark on the notation, in Online Convex Optimization literature, the loss functions are written as $f(x, y_t) \equiv f_t(x)$, dropping the explicit dependence on the outcome y_t. The decision space $\mathcal{D}$ is assumed to be convex, as the functions $f_t : \mathcal{D} \to \mathbb{R}$. This framework was first devised in [Zinkevich, 2003], and has been later wildly used in the machine learning community to engineer optimization procedures [Shalev-Shwartz et al., 2012].

Convexity plays a central role in most of the analysis made in Online Learning, and Online Convex Optimization. Convexity of the domain $\mathcal{D}$ and of the loss functions $f(\cdot, y)$ bounds the problem geometry and let us derive simple and efficient learning procedures.

In this chapter the decision space $\mathcal{D}$ is a convex subset of $\mathbb{R}^N$. As in the case of uncountable experts discussed in Section 2.2.1, the best expert is the one who plays at each round a fixed point $x \in \mathcal{D}$. In Section 2.6 we will discuss how this framework is well suited to optimize complex functions, as Neural Networks, where we can think as $x \in \mathcal{D}$ as the set of parameters we are trying to optimize. Indeed many state of the art optimization techniques in the field of machine learning have been taking inspiration from the field of Online Optimization [Duchi et al., 2011].

2.5.1 A General Algorithm for Online Convex Optimization

In this Section we will see an algorithm called *Online Mirror Descent* (OMD), that generalizes many Online Convex Optimization algorithms. It is a first order method (*i.e.* it uses only information from the gradient of the loss function) that works in the dual space defined by the choice of some regularizator. The OMD algorithm is general and optimal in the sense that every Online Convex problem can be learned online nearly optimally with OMD, the precise definition of the optimality of the OMD algorithm is quite complex to be summarized here and can be found in [Srebro et al., 2011].

OMD works with a class of regularitators called Bregman Divergences, [Banerjee et al., 2005].

In this section we assume that $\mathcal{D} \subset \mathbb{R}^N$.

Definition 2.5.1. *(Bregman divergence). Given a differentiable convex function* $\psi : \mathcal{D} \to \mathbb{R}$ *the Bregman divergence is defined as an operator* $d_\psi : \mathcal{D} \times \mathcal{D} \to \mathbb{R}^+$ *defined for* $\mathbf{x}, \mathbf{y} \in \mathcal{D} \times \mathcal{D}$ *as:*

$$d_\psi(\mathbf{x}, \mathbf{y}) = \psi(\mathbf{x}) - \psi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\psi(\mathbf{y})\rangle. \tag{2.19}$$

Since ψ is convex we have that $d_\psi(x, y) \geq 0$. We can see that by linearization of $\psi(x)$ around $y \in \mathcal{D}$ and thanks to convexity the other terms are positive. However note that, since the operator defined in Equation (2.19) is not symmetric in its arguments, it does not formally define a metric in the space $\mathcal{D}$.

Now we will present two example of Bregman divergences that we will use to define specifications of the OMD algorithm in Chapter 5. For $\mathbf{x}, \mathbf{y} \in \Delta_{N-1} \subset \mathbb{R}^N$, consider $\psi(\mathbf{x}) = ||\mathbf{x}||_2^2$ then the Bregman divergence becomes $d_\psi(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2^2$, which is the Euclidean norm. For $\psi(\mathbf{x}) = \sum_{i=1}^{N} x_i \log(x_i)$ then $d_\psi(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} x_i \log(x_i / y_i)$, which is the well know Kullback–Leibler divergence [Van Erven and Harremos, 2014].

The OMD algorithm for Online Convex Optimization uses the regularization given by a Bregman divergence to follow the best point in the convex set $\mathcal{D}$ up to now, but it is kept close to the current one by the divergence operator. Formally:

Definition 2.5.2. *(Online Mirror Descent). OMD for a Bregamn Divergence induced by the differentiable, convex real values function* ψ*, and for a set of learning rates* $\{\eta_0, \ldots, \eta_T\}$ *has the following update rule:*

$$\mathbf{x}_{t+1} = \operatorname*{arg\,inf}_{x \in \mathcal{D}} \left\{ d_\psi(\mathbf{x}, \mathbf{x}_t) + \eta_t \langle \nabla f_t(\mathbf{x}_t), \mathbf{x} - \mathbf{x}_t \rangle \right\}. \tag{2.20}$$

Next we will show the idea for a general bound for the OMD algorithm, which explains the geometric ideas behind the OMD algorithm. Note that, in general, the analysis can be refined by fixing the loss function f_t or the convex function ψ.

F: questa frase invece non mi piace....direi invece It is possible to show (citazione di dove c'è il teorema) TEOREMA

The convex function ψ is assumed to be differentiable in the domain $\mathcal{D}$.

Lemma 2.5.1. *Let $d_\psi : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ the Bregman divergence associated to the convex smooth function ψ. Moreover, assume ψ is α-strong convex w.r.t. $||\cdot||$. Then $\forall \mathbf{x} \in \mathcal{D}$ we have*

$$\eta_t(f_t(\mathbf{x}_t) - f_t(\mathbf{x})) \le d_\psi(\mathbf{x}, \mathbf{x}_t) - d_\psi(\mathbf{x}, \mathbf{x}_{t+1}) + \frac{\eta_t^2}{2}||\nabla f_t(\mathbf{x}_t)||_*^2,$$

where we defined the dual norm $||\cdot||_*$ with respect to the norm $||\cdot||$.

Definition 2.5.3. *(Dual Norm). Let $\mathbf{x} \in X$, the dual norm $||\cdot||_*$ of a norm $||\cdot||$ is defined as:*

$$||\mathbf{x}||_* = \sup_{\mathbf{y}:||\mathbf{y}||\le 1} \langle \mathbf{x}, \mathbf{y} \rangle.$$

The specific norm $||\cdot||$ in Theorem 2.5.1 can be chosen depending on the specific Bregman divergence, in order to simplify the analysis. Indeed, Theorem 2.5.1 can be used to prove a regret bound for the general OMD algorithm.

Theorem 2.5.1. *(Regret Bound for Online Mirror Descent). Together with the assumptions of Theorem 2.5.1 and if $\eta_t \ge 0$ is a decreasing sequence of learning rates, then we have:*

$$R_T \le \max_{t \le T} \frac{d_\psi(\mathbf{x}, \mathbf{x}_t)}{\eta_T} + \frac{1}{2\alpha} \sum_{t=1}^{T} \eta_t ||\nabla f_t(\mathbf{x}_t)||_*^2.$$

By choosing $\eta_t = \frac{D\sqrt{\alpha}}{\sqrt{\sum_{t=1}^{T} ||\nabla f_t(\mathbf{x}_t)||_*^2}}$, where $D = \max_{t \le T} d_\psi(\mathbf{x}, \mathbf{x}_t)$, we have a bound for the OMD algorithm of

$$R_T \le \frac{2D}{\sqrt{\alpha}} \sqrt{\sum_{t=1}^{T} ||\nabla f_t(\mathbf{x}_t)||_*^2}. \tag{2.21}$$

Notice that, if the gradient under the dual norm is bounded by $||\nabla f_t(\mathbf{x}_t)||_* \le G \; \forall t \le T$, then we have that

$$R_T \le \frac{2DG}{\sqrt{\alpha}} \sqrt{T}, \tag{2.22}$$

Algorithm 1 OMD for Online Convex Optimization

```
Require: learning rate sequence $\{\eta_1, \dots, \eta_T\}$
Set $\mathbf{x}_1 \leftarrow \frac{1}{M}\mathbf{1}$
for $t \in \{1, \dots, T\}$ do
    Observe $f_t(\mathbf{x}_t)$ decided by the adversary
    $\mathbf{x}_{t+1} = \arg\inf_{\mathbf{x}\in\mathcal{D}} \{d_\psi(\mathbf{x}, \mathbf{x}_t) + \eta_t \langle \nabla f_t(\mathbf{x}_t), \mathbf{x} - \mathbf{x}_t \rangle\}$
end for
```

which is sub-linear in T.

If $\eta_t = \eta > 0$ is a constant sequence then Theorem 2.5.1 can be simplified to give:

$$R_T \leq \frac{d_\psi(\mathbf{x}, \mathbf{x}_1)}{\eta} + \frac{\eta}{2\alpha} \sum_{t=1}^{T} ||\nabla f_t(\mathbf{x}_t)||_*^2. \tag{2.23}$$

The OMD algorithm is a general technique to exploit the geometric convexity of the problem and gives rise to Hannan-consistent strategies in the case of uncountable convex decision spaces. By specializing the loss function and the Bregman divergences we can generate many algorithms that are state of the art in the Online Convex optimization problem, and achieve better theoretical guarantees than the general analysis we saw for the OMD algorithm, in fact we will show in Chapter 5 that the Online Newton Step algorithm, which can be seen as an instance of the OMD algorithm, can achieve $\mathcal{O}(\log T)$ regret rather then $\mathcal{O}(\sqrt{T})$ regret.

F: manca riferimento nel testo e spiega

2.5.2 Mirror Version of the Online Mirror Descent Algorithm

The reason why OMD works is not that we are following the gradient, that points to the minimum of the function; indeed, the sub-gradient (Definition 2.5.5) of a loss function does not point to the minimum in general (Figure 2.3). In practice the reason why OMD and other first order methods are effective is because of the convexity of the loss function and because of the following inequality for the instantaneous regret of convex loss functions:

$$f_t(\mathbf{x}_t) - f_t(\mathbf{x}) \leq \langle \nabla f_t(\mathbf{x}_t), \mathbf{x}_t - \mathbf{x} \rangle, \tag{2.24}$$

and so to minimize the left hand side of Equation (2.24) we can minimize the right hand side of Equation (2.24). Minimizing strictly a linear approximation of the instantaneous regret is not ideal since the environment

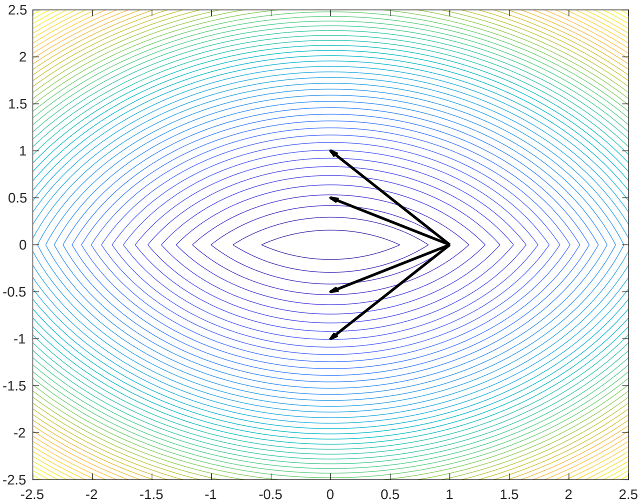


Figure 2.3: Contour lines of $f(x,y) = \max\{x^2+(y-1)^2, x^2+(y+1)^2\}$*, together with sub-gradient directions from* $(1,0)$*. All sub-gradient will point to increasing values of the function*

is adversarial. Instead we minimize the linear approximation together with a regularization term which is given by the Bregman divergence d_ψ.

In order to understand more formally the inner workings of the OMD algorithm we have to introduce some concept from optimization theory:

Definition 2.5.4. *(Fenchel Conjugate). For a function* $f : \mathbb{R}^N \to \mathbb{R}$ *we can define the Fenchel conjugate as:*

$$f^*(\theta) = \sup_{\mathbf{x}\in\mathbb{R}^N} \langle \mathbf{x}, \theta \rangle - f(\mathbf{x}). \tag{2.25}$$

Definition 2.5.4 can be interpreted as a generalized inf function as $f^*(\mathbf{0})$ is the classical inf function. For $\mathbf{x} \neq \mathbf{0}$ then we are looking for the infimum of the function f when the axis of the function are rotated w.r.t. the hyperplane $H(\mathbf{x}) = \langle \theta, \mathbf{x} \rangle$, as illustrated in Figure 2.4.

Definition 2.5.5. *(Sub-Gradient). For a function* $f : \mathbb{R}^N \to \mathbb{R}$ *we can define the set of sub-differentials at* x_0 *as:*

$$\partial f(x_0) = \{g : f(x) - f(x_0) \geq \langle g, x - x_0 \rangle, \forall x\}. \tag{2.26}$$

For a differentiable at $\mathbf{x}_0$ function we have $\partial f(\mathbf{x}_0) = \{\nabla f(\mathbf{x}_0)\}$.

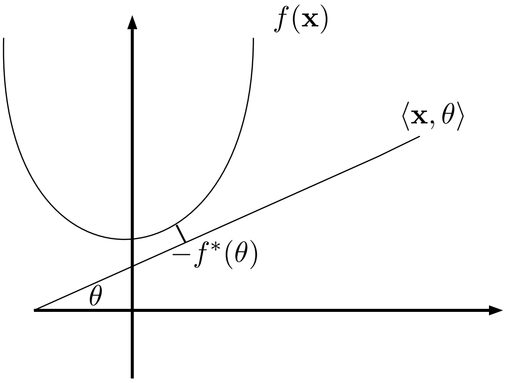


Figure 2.4: Fenchel Conjugate Function.

Finally, the following theorem explains the name of the OMD algorithm and its real more interesting formulation:

Theorem 2.5.2. *Let d_ψ be a Bregman divergence operator then we have the following equality:*

$$\arg\inf_{\mathbf{x}\in\mathcal{D}}\{d_\psi(\mathbf{x},\mathbf{x}_t)+\eta_t\langle\nabla f_t(x_t),\mathbf{x}-\mathbf{x}_t\rangle\} = \nabla\psi^*_{\mathcal{D}}(\nabla\psi(\mathbf{x}_t)-\eta_t\nabla f_t(\mathbf{x}_t)), \quad (2.27)$$

where $\psi_{\mathcal{D}}$ is the restriction of ψ to the convex set $\mathcal{D}$, i.e., $\psi_{\mathcal{D}}(\mathbf{x}) = \psi(\mathbf{x}) + \mathbb{I}^{\infty}_{\mathcal{D}}(\mathbf{x})$*, where we defined*

$$\mathbb{I}^{\infty}_{\mathcal{D}}(\mathbf{x}) = \begin{cases} 0 & \textit{if } \mathbf{x}\in\mathcal{D} \\ +\infty & \textit{otherwise.} \end{cases}$$

.

Theorem 2.5.2 shows the real nature of the OMD algorithm, which is to update predictions using the gradient of the loss function, in the dual space defined by the function ψ. For example if $\psi(\mathbf{x}) = \frac{1}{2}||\mathbf{x}||_2^2$ then we have $\nabla\psi(\mathbf{x}_t) = \mathbf{x}_t$ and $\nabla\psi^*(\mathbf{x}) = \Pi_{\mathcal{D}}(\mathbf{x})$, and we obtain the Online Gradient Descent algorithm, $\mathbf{x}_{t+1} = \Pi_{\mathcal{D}}(\mathbf{x}_t - \eta_t\nabla f_t(\mathbf{x}_t))$, that we will explore with detail in Chapter 6.

2.6 From Online Learning to Statistical Learning

Now we explore the connection between the Online Optimization framework and classical concepts of classical Statistical Learning techniques. More concretely we can prove and design a whole class of algorithms that are

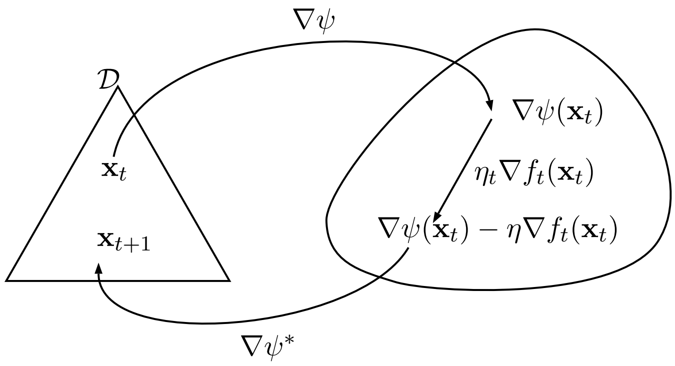

$\nabla\psi$
$\mathcal{D}$
$\mathbf{x}_t$
$\mathbf{x}_{t+1}$
$\nabla\psi(\mathbf{x}_t)$
$\eta_t\nabla f_t(\mathbf{x}_t)$
$\nabla\psi(\mathbf{x}_t) - \eta\nabla f_t(\mathbf{x}_t)$
$\nabla\psi^*$

Figure 2.5: Online Mirror Descent as Mirror Updates.

define PAC

Agnostically PAC Learnable with Online Learning Techniques. Classical statistical learning theory deals with examples (or observations) and models of the phenomena. Then it uses the model to predict the future observations [Bousquet et al., 2003]. Quite informally one could say that we are trying to infer concept from examples. A concept is a map $\mathcal{C} : \mathcal{D} \to \mathcal{Y}$, where $\mathcal{D}$ is the domain space and $\mathcal{Y}$ is the set of labels for the examples. We then observe a sample from an unknown distribution $\mathcal{X}$ such that $(x, y) \sim \mathcal{X}$. What we need to achieve is to learn a mapping $y : \mathcal{D} \to \mathcal{Y}$ such that the error under the distribution $\mathcal{X}$ is small. The loss function needed to define this error is not specific to the problem and can be decided by the user. This is called generalization error and, for a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, it is defined as:

$$e(h) = \mathbb{E}_{(x,y)\sim\mathcal{X}}[l(h(x), y)]. \tag{2.28}$$

The goal for an algorithm $\mathcal{A}$ is to produce a hypothesis h with small generalization error. In general, it is difficult to obtain small generalization error and how difficult it is clarified by the following theorem called the *No free lunch theorem* [Mitchell et al., 1997]. This restriction gives raise to the concept of Probably Approximately Correct (PAC) learnability.

Definition 2.6.1. *(PAC learnable). An hypothesis class $\mathcal{H}$ is PAC learnable w.r.t. the loss l if there exists a learner $\mathcal{A}$ that given a sample S_N of examples learns an hypothesis $h \in \mathcal{H}$ s.t. for all ϵ, δ there exists $N_{\epsilon,\delta}$ such that for any distribution $\mathcal{D}$ we have a generalization error s.t. $\mathbb{P}[e(h) < \epsilon] \geq 1 - \delta$.*

Usually, we also require that the algorithm $\mathcal{A}$ learns the concept h in polynomial time w.r.t. the parameter of the problem.

An example of such learning problems could be the classification of spam emails. In this case $\mathcal{D}$ is the vectorial representation of the text and $\mathcal{Y} = \{0, 1\}$, indicating weather or not the email is a spam or not. If we choose as a model a linear classifier then the hypothesis space is $\mathcal{H} = \{h = \mathbb{I}[\langle x, w\rangle \geq 1/2]\}$ and the loss could be chosen as $l(y_1, y_2) = |y_1 - y_2|$.

PAC learnability intuitively requires the existence of an hypothesis $h \in \mathcal{H}$ with near zero generalization error, otherwise the class $\mathcal{H}$ is not PAC learnable. But we can weaken the concept of PAC learnability by addressing directly this issue.

Definition 2.6.2. *(PAC agnostic learnable). Given the same definitions of Definition 2.6.1, an hypothesis class $\mathcal{H}$ is PAC agnostic learnable if we have a generalization error s.t.* $\mathbb{P}\left[e(h) < \inf_{\tilde{h}\in\mathcal{H}} e(\tilde{h}) + \epsilon\right] \geq 1 - \delta$.

Determining which hypothesis spaces $\mathcal{H}$ are PAC learnable (agnostically or not) for specific spaces is an open and complex issue, but the case for convex hypothesis class $\mathcal{H} \subset \mathcal{R}$ can be solved by Online Learning techniques, showing the versatility of the methods. Moreover, the approach to prove such theorem gives a constructive methodology to solve agnostic PAC learnable problems.

cit theorem

Theorem 2.6.1. *For every hypothesis class $\mathcal{H}$ and bounded loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, for which does exists a low regret algorithm $\mathcal{A}$, the problem is agnostic PAC learnable. In particular, these conditions are satisfied if the hypothesis space $\mathcal{H}$ and the loss function l are convex.*

Proof. (Sketch). Initialize the learner with the hypothesis $h_0 = \mathcal{H}$. For every iteration $t \leq T$: observe a sample $(x_t, y_t) \sim \mathcal{X}$ and a loss function $l_t := l(h_t(x_t), y_t)$. Then update the hypothesis $h_{t+1} = \mathcal{A}(l_1, \ldots, l_t)$.

At $t = T$ return $\bar{h} = \frac{1}{T}\sum_{t=1}^{T} h_t \in \mathcal{H}$.

The proof then continues by defining the random variable $X_T^{(1)} = \sum_{t=1}^{T} e(h_t) - l(h_t(x_t), y_t)$. This is a martingale and $\mathbb{E}[X_T^{(1)}] = 0$. Moreover $|X_T^{(1)} - X_{T-1}^{(1)}| < K$ since the loss function f is bounded. We can normalize the losses so that $K = 1$, and then apply the Azuma martingale inequality $\mathbb{P}[X_T^{(1)} > c] \leq e^{-\frac{c^2}{2T}}$ [Azuma, 1967].

For an appropriate choice of c we get

$$\mathbb{P}\left[\frac{1}{T}\left[\sum_{t=1}^{T} e(h_t) - l(h_t(x_t), y_t)\right) > \sqrt{\frac{2\log(\delta/2)}{T}}\right] \leq \delta/2, \tag{2.29}$$

defining $h^* = \arg\inf_{h \in \mathcal{H}} e(h)$ and $X_T^{(2)} = \sum_{t=1}^{T} e(h^*) - l(h^*(x_t), y_t)$ we can obtain

$$\mathbb{P}\left[\frac{1}{T}\left(\sum_{t=1}^{T} e(h^*) - l(h^*(x_t), y_t)\right) < -\sqrt{\frac{2\log(\delta/2)}{T}}\right] \leq \delta/2. \tag{2.30}$$

By the definition of regret R_T we obtain

$$\frac{1}{T}\sum_{t=1}^{T} e(h_t) - e(h^*) = R_T/T + X_T^{(1)} - X_T^{(2)}, \tag{2.31}$$

and from inequalities in Equations (2.29), (2.30) and from Equation (2.31) we have:

$$\mathbb{P}\left[\frac{1}{T}\sum_{t=1}^{T} e(h_t) - e(h^*) > \frac{R_T}{T} + 2\sqrt{\frac{2\log(\delta/2)}{T}}\right] \leq \delta. \tag{2.32}$$

Now simply thanks to the linearity of the error operator $e : \mathcal{H} \to \mathbb{R}$ we have that

$$\mathbb{P}\left[e(\bar{h}) < e(h^*) + R_T/T + 2\sqrt{\frac{2\log(\delta/2)}{T}}\right] \leq 1 - \delta,$$

and since $R_T/T \to 0$ we can find $\tilde{T}$ large enough such that the thesis is verified. □

This result has been presented since it is useful to prove the general behavior of Hannan-consistent strategies in environments driven by a stationary distribution.

Chapter 3

Information, Prediction and Investing

In Chapter 2 we described at a high level the framework for Online Learning in Adversarial environment. Now we draw its connections with predictions and investments. It surely seems counter-intuitive to speak about predictions in an adversarial framework, since we are used to think about predictions only of stochastic processes, but the way to think about predictions in adversarial environments is to think about probability assignment and empirical frequencies. The roots of this formulation are to be traced back to the Bell Laboratories in the 50s, from works of Kelly [Kelly Jr, 2011], linking sequential betting and concept from information theory [Cover and Thomas, 2012]. This connection is of paramount importance to understand sequential investing as an instance of sequential decision problem. We will first draw the parallelism between probability assignment over discrete events and Online Learning, and then extend the discussion to sequential investments.

3.1 Probability assignment

In this section we will draw the parallelism between assigning probabilities to outcomes, predictions, information theory and investments. In the case of N possible bets the decision space $\mathcal{D}$ is the $\Delta_{N-1} \subset \mathbb{R}^N$ probability simplex while the outcome $\mathcal{Y}$ space is the set $\{1, \ldots, N\}$, representing the winning bet at each turn. The loss function $f(x, y)$ should have these natural properties: low when $x_y \approx 1$ and high when $x_y \approx 0$, where x_y is the probability assigned to the outcome y. The inverse log-likelihood seems a reasonable proposal, not only because of the multiplicative additive property of the logarithm, (we need the loss to be an additive quantity) but has also a deeper connection to information that we will discuss more formally in Section 3.1.2.

Definition 3.1.1. *(Self Information Loss). In the sequential probability assignment problem the loss function $f(\mathbf{x}, y)$, $\mathbf{x} \in \Delta_{N-1}$ and $y \in [1, \dots, N]$ is defined as*

$$f(\mathbf{x}, y) = -\log\left(x^{(y)}\right),$$

where $x^{(y)}$ is the probability assigned to outcome $y \in \mathcal{Y}$.

In the case of simulable experts, the prediction $\mathbf{x}_t$ of the agent is a function of the history of outcomes $y^{t-1} := \{y_1, y_2, \dots, y_{t-1}\}$.

The cumulative loss for the agent $\mathcal{A}$ is then given by

$$L_T = \prod_{t=1}^{T} f(\mathbf{x}_t, y_t), \tag{3.1}$$

and can be interpreted as the log-likelihood assigned to the outcome sequence y^T since

$$L_T = \sum_{t=1}^{T} f(\mathbf{x}_t, y_t) = -\log\left(\prod_{t=1}^{T} x_t^{(y_t)}\right), \tag{3.2}$$

where we can interpret $\prod\limits_{t=1}^{T} x^{(y_t)}$ as the probability assigned to the entire outcome sequence y^T. This is already very similar to the compression-entropy rate one encounters in a classical lossless encoder, such as the arithmetic encoder [Langdon, 1984]. We will explore the connections to information theory later on in Chapter 3.1.2.

Similarly we can define the loss for an expert $e \in \mathcal{E}$ as:

$$L_{e,T} = \sum_{t=1}^{T} f(\mathbf{x}_{e,t}, y_t) = -\log\left(\prod_{t=1}^{T} x_{e,t}^{(y_t)}\right), \tag{3.3}$$

and the regret for each expert $e \in \mathcal{E}$ is defined as

$$R_{e,T} = L_T - L_{e,T} = \log\left(\prod_{t=1}^{T} x_{t,e}^{(y_t)} \Big/ \prod_{t=1}^{T} x_t^{(y_t)}\right), \tag{3.4}$$

and the regret w.r.t. a generic class $\mathcal{E}$ of experts is defined as:

$$R_T = \sup_{e \in \mathcal{E}} \log\left(\prod_{t=1}^{T} x_{t,e}^{(y_t)} \Big/ \prod_{t=1}^{T} x_t^{(y_t)}\right), \tag{3.5}$$

where the class of experts $\mathcal{E}$ can be finite or uncountable.

Moreover, the self information loss defined in Definition 3.1.1, is clearly exp concave with coefficient $\nu = 1$ as defined in Chapter 2, and we know that we have $R_T \leq \log(N)$ in the case of finite experts and $R_T \leq N(\log(T/N)+1)$ in the case of uncountable experts, by Theorem 2.3.2. The case of the expert class being identified with the simplex Δ_{N-1} can be interpreted as a convex hull of experts and so the Theorem 2.3.2 gives a $R_T = \mathcal{O}(\log T)$ regret bound on the problem of probability assignment described in the previous section.

3.1.1 Laplace Mixture Forecaster

Fixing the log-loss we can show better regret bounds on the Mixture Forecaster for uncountable experts, introduced in Theorem 2.3.2. The Mixture Forecaster with log-loss has regret bound ([Cesa-Bianchi and Lugosi, 2006] Theorem 9.3)

$$R_T \leq (N-1)\log(T+1), \tag{3.6}$$

and it is called Laplace Mixture Forecaster [Weinberger et al., 1994]. The improved constants for the Laplace Mixture Forecaster results from exploiting both exp-concavity and the additive property of the log-loss.

3.1.2 Connection to Information Theory

The link between sequential predictions and information theory has been observed in [Kelly Jr, 2011], and connects the concept of sequential betting (or predictions) and entropy.

Kelly put himself in a setting where the bettor has to predict the outcomes of binary events, given private information from an *information channel* prone to errors. The binary bet pays double for a correct prediction and zero for an incorrect one. The input bits of the information channel are the correct outcomes of the binary sequential event, but they reach the end of the private channel with probability p of being correct and $q = 1 - p$ of being wrong. Clearly the optimal strategy with $p = 1$ is to bet everything on each turn reaching a final wealth, after T rounds, of $V_T = 2^T$. In case $p < 1$ it is not clear which strategy is the best to follow, this is clearly related and still under philosophical debate as the St. Petersburg paradox [Samuelson, 1977]. Kelly proposed to maximize the grow rate of the wealth by investing a constant fraction of the current wealth. The growth rate G of the wealth V_T is defined as

$$G = \lim_{T \to +\infty} \frac{1}{T} \log_2(V_T).$$

Calling $l \in [0,1]$ the fraction of the wealth invested in the bet we have a capital after T rounds of

$$V_T = (1+l)^W (1-l)^{T-W},$$

and the associated growth rate becomes:

$$G = p \log_2(1+l) + q \log_2(1-l),$$

which is maximized for $l = p - q$ giving $G_{\max} = 1 + p\log_2(p) + q\log_2(q)$ which is the rate of transmission for the communication channel, *i.e.* the number of bits transferred for unit of time. This is the trivial case and can be extended to arbitrary odds and number of bets.

The equivalent formulation in Online Learning can be obtained by observing that $\mathcal{D} = \Delta_0$ and that we are betting a fraction l_t on the event being 0 and a fraction $1 - l_t$ on the the outcome being 1. In that case the wealth at time t will be $V_t = V_{t-1} l_t^{1-y_t}(1-l_t)^{y_t}$ and hence:

$$log(V_T) = \sum_{t=1}^{T} \log(l_t(y_t - 1) + (1 - l_t)y_t), \tag{3.7}$$

which defines the cumulative loss

$$L_T = -log(V_T) = \sum_{t=1}^{T} - \log(l_t(1 - y_t) + (1 - l_t)y_t),$$

which is equivalent to the loss defined in Equation (3.2).

By defining the growth rate at T as $G_T = \frac{1}{T}\log_2(V_T)$ we can observe that $L_T = TG_T \log(2)$ and so a learner $\mathcal{A}$ that obtains sub-linear regret $R_T/T \to 0$, where the expert class is composed of constant experts for which $l_t = const$, is equivalent to obtaining a growth rate $G_T \to G_{\max}$.

This draws the connection to information rate as defined by Shannon in terms of information bits and growth rate of a betting strategy, and the fact that an Hannan-consistent strategy is able to converge to the highest growth rate.

3.1.3 Horse Races

In this section we will see how sequential investment is equivalent to the problem of sequential betting discussed in the previous section. In the previous chapter we saw how to formalize sequential betting in the simple case of doubling odds and binary outcomes into the Online Learning formulation.

Now we will extend the model to account variable odds and multiple bets, and how this is connected to investing.

Let us model horse races as a sequential betting process, in which we have N horses each paying a payoff of $o_{t,i}\ \forall i \in 1,\dots,N$. The agent $\mathcal{A}$ splits its wealth into N separate betting by choosing an element of the simplex Δ_{N-1}.

The wealth of the agent $\mathcal{A}$ at time t will be the $V_t = V_{t-1}\langle \mathbf{x}_t, \mathbf{y}_t\rangle$, where $\mathbf{y}_t = o_{y_t}\mathbf{e}_{y_t} \in \mathbb{R}^N$, *i.e.* the basis vector with 1 as the $y_t \in 1,\dots,N$ component, which represents the winning horse for the turn, and o_{y_t} is the payout of the bet at time t, on the t_y horse winning. As we did in the previous section we can apply $-\log(\cdot)$ so that we can embed the problem into an Online Learning framework. By defining

$$L_T = -\log(V_T) = -\log(V_{T-1}) - \log(\langle \mathbf{x}_t, \mathbf{y}_t\rangle),$$

that implies

$$L_T = \sum_{t=1}^{T} -\log(\langle \mathbf{x}_t, \mathbf{y}_t\rangle), \tag{3.8}$$

we obtain exactly the same formulation presented at the beginning of the chapter. Moreover, we note that the regret R_T does not depend on the value of the payout o_{y_t}.

We saw in Section 2.2.1 that Theorem 2.3.2 assures that we have a sublinear regret $R_T = \mathcal{O}(\log T)$ in case that the expert class $\mathcal{E}$ is being generated by the convex hull of finite basic experts $\mathcal{E}_N$, which in this case can be taken as the N experts always predicting $\mathbf{x}_{t,j} = \mathbf{e}_j, \forall j \in 1,\dots,N$. The convex hull generated by $\mathcal{E}_N$ is then composed by experts predicting a constant element of the simplex $\mathbf{x}_{t,e} = \mathbf{x}_e \in \Delta_{N-1}$.

Theorem 2.3.2 is stating that we can obtain asymptotic wealth equivalent to the one obtained by the best expert in hindsight, for all sequences of outcomes.

A very similar formulation can be obtained for the case of sequential investments. In the case of horse races we have just one winner for each day, while in the case of stock investing we have a different payout for each stock. In the following section we will present how to model sequential decision problems in the Online Learning formulation.

3.2 From Horse Races to Online Portfolio Optimization

We can formulate the portfolio allocation as a sequential betting problem. Let us imagine that there are no real life issues associated with trading costs and liquidity (this issues will be discussed in the following Chapter 4). Then the best strategy would be to invest at each round t the entire capital on one single stock, knowing that will be the best performance stock at round t. But assuming an adversarial environment we have to randomize our bet, or equivalently distribute our wealth into the N horses accordingly to our randomization probabilities, as they are equivalent under Equation (3.8).

3.2.1 The Online Portfolio Optimization Model

The model consists in a sequential wealth allocation in $N \in \mathbb{N}$ stocks for discrete rounds $t \in \{1, \ldots, T\}$, where T is the investment horizon. Note that the set of times is arbitrary, and could also be non-homogeneous, in practice in the Online Portfolio Optimization case, it is usually thought to be in days. The evolution of the stock $i \in 1, \ldots, N$ prices at time t, $P_{t,i}$, define the price relatives $y_{i,t} = \frac{P_{i,t+1}}{P_{i,t}}$, and we can define the price relative vector at time t as $\mathbf{y}_t = (y_{1,t}, \ldots, y_{N,t}) \in \mathbb{R}^N$.

An investor dividing, at round t, its wealth W_t into a fraction $\mathbf{x}_t \in \Delta_{N-1}$ for each asset will get a wealth $W_{t+1} = W_t \langle \mathbf{x}_t, \mathbf{y}_t \rangle$ at round $t+1$. As in Section 3.1.2 we can define the growth rate

$$G_T = \log(W_T) = \sum_{t=1}^{T} \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle).$$

As in the case of binary outcomes, *i.e.* horse races, we can redefine the problem in an Online Learning framework, by defining the loss $f(\mathbf{x}, \mathbf{y}) = -\log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)$ and a cumulative loss as

$$L_T = -G_T = \sum_{t=1}^{T} -\log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle).$$

Exactly as in the previous Section, the expert class is generated by the convex hull of the base class $\mathcal{E}_N$, composed by the experts always betting on the win of the same horse $i \in 1, \ldots, N$, or, equivalently, allocating all the portfolio on the same asset, at every round. The convex hull of the class is the class of experts $\mathcal{E}$, so that at every turn t, the expert is allocating all the wealth in a specific element $\mathbf{x} \in \Delta_{N-1}$. In the Online Portfolio literature this class of allocations is called *Constant Rebalancing Portfolio* (CRP), and we will define its wealth as $W_T(\mathbf{x}) = W_0 \prod_{t=1}^{T} \langle \mathbf{x}, \mathbf{y}_t \rangle$.

As in every adversarial environment, we have to compare our losses with the best expert in the expert class through the concept of regret:

$$\begin{align} R_T &= L_T - \inf_{e \in \mathcal{E}} L_{T,e} \tag{3.9} \\ &= \sum_{t=1}^{T} - \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle) - \inf_{\mathbf{x} \in \Delta_{N-1}} \sum_{t=1}^{T} - \log(\langle \mathbf{x}, \mathbf{y}_t \rangle). \tag{3.10} \end{align}$$

The CRP attaining the minimum loss

$$\mathbf{x}^* = \inf_{\mathbf{x} \in \Delta_{N-1}} \sum_{t=1}^{T} - \log(\langle \mathbf{x}, \mathbf{y}_t \rangle).$$

is called *Best Constant Rebalancing Portfolio* (BCRP).

As we shall see in the next section, Constant Rebalancing Portfolios (CRP) are a very powerful class of strategies and being competitive (in terms of sub-linear regret) with respect to this class assures good theoretical guarantees.

3.2.2 Effectiveness of Constant Rebalancing Portfolios

The CRP is a strategy that each round t redistributes its wealth into the same distribution $\mathbf{x} \in \Delta_{N-1}$. As we saw in the previous Section this strategies can be identified as the ones generated by expert class $\mathcal{E}$ defined previously. The Buy and Hold (BAH) is a strategy that holds on an allocation at the start of the investment period and hold on to it to the end of the investment horizon T. The wealth of an BHA strategy can be written as $W_T = \langle \mathbf{x}, \prod_{t=1}^{T} \mathbf{y}_t \rangle$.

A simple example can illustrate the effectiveness of the CRP strategies, and the inherently difference that exists between the Modern Portfolio Theory and the Online Portfolio Optimization techniques. Imagine to have two stocks, and the adversary can choose the value of the price relatives in the set: $y_{1,t}, y_{2,t} \in \left\{\frac{3}{5}, \frac{8}{5}\right\}$. Imagine that the adversary picks a price relative in the set $\left\{\frac{3}{5}, \frac{8}{5}\right\}$ with equal probability. Every BHA allocation is exponentially decaying $\mathbb{E}[W_T] = \langle \mathbf{x}, (\frac{24}{25}, \frac{24}{25}) \rangle = \frac{24}{25}$ and hence has decaying growth rate $G_T < 0$. Conversely, the equally allocated CRP $\mathbf{x} = (\frac{1}{2}, \frac{1}{2})$ has positive growth rate and exponentially increasing wealth: $\mathbb{E}[W_T] = (11/10)^T$ and $G_T = T \log(11/10) > 0$.

Historically, this example has been called Shannon Demon [Poundstone, 2010] and being compared to the Maxwell's Demon since, as in the thermodynamics case, Shannon's Demon is generating wealth (energy in the case on Maxwell) from nothing since both stocks are martingales, and opponents to the Capital Growth Theory, used this argument to invalidate this ideas. In reality there is nothing strange about this example, and it is just one of the many techniques that exploits the existence of volatility and converts it into wealth, as theoretically does a delta-hedged option in the Black and Scholes model [Black and Scholes, 1973].

Chapter 4

Problem Formulation

In this section we will extend the Online Portfolio Optimization model to consider transaction costs. The resulting framework will be central in our contributions. Indeed, the importance of the trading mechanism is usually not taken into account in the models of Online Portfolio Optimization, notably, the most important aspect left out of the analysis is transaction costs. Including transaction costs into the Online Portfolio Optimization model is non-trivial and complex. The reason why transaction costs are more difficult to include into the model is that the inclusion of transaction costs change significantly the loss function and, as we shall see, the theoretical guarantees of the algorithms do relay heavily on very strict conditions on the loss function, such as convexity and exp-concavity. Notice that an algorithm that guarantees sub-linear regret without transaction costs, is not guaranteed to have sub-linear regret in the more realistic scenario in which trading costs are present.

Very few works include transaction costs in the Online Portfolio Optimization model. There exist a wide variety of heuristic methods that tried to overcome this problem [Li et al., 2018a, Yang et al., 2018], but they do not provide any guarantee on the regret in the presence of transaction costs. To the best of our knowledge, there are only two studies that analyze transaction costs theoretically: Universal Portfolio with Costs (U$_C$P) [Blum and Kalai, 1999], and Online Lazy Updates (OLU) [Das et al., 2013], but only OLU gives an algorithm to implement. We will present the algorithm designed in these works in Chapter 5. The principal contribution of this thesis is to give an algorithm that has sub-linear regret in the Online Portfolio Optimization problem with transaction costs.

4.1 Online Portfolio Optimization with Transaction Costs

Transaction costs are notably difficult to model or even define. In order to model trading costs correctly, one would have to take into account many aspects of the trading mechanism and explore the mechanism of trading in its minutiae, this field of research is called *market micro-structure*. Great starting references can be found in [Harris, 2003] and [O'hara, 1997], in which the authors explore the practical implementation of a trade and its theoretical formulation, respectively. Our model of transaction costs is much simpler in order to recover regret bounds in the presence of trading costs.

F: mi sembra un po' riduttivo. M: si. però è vero. Come lo giustifico meglio?

At the end of this section we will then perform some approximations in order to define a new concept of regret to be used in the framework of Online Portfolio Optimization. The final model for Online Portfolio Optimization with transaction costs, that will be derived in this section is the same used in [Das et al., 2013].

Following the approach previously used in the OPO literature [Blum and Kalai, 1999], we use an approximation of the real transaction costs considering them proportional to the difference in portfolio allocation. Formally, the transaction costs, at round t, are implicitly determined by the solution of the following equation:

$$\begin{aligned} W_{t-1} = \tilde{W}_{t-1} &+ \gamma_s \sum_{i=1}^{N} \left(\frac{x_{i,t-1} y_{i,t-1}}{\langle \mathbf{x}_{t-1}, \mathbf{y}_{t-1} \rangle} - x_{i,t} \tilde{W}_{t-1} \right)^{+} \\ &+ \gamma_b \sum_{i=1}^{N} \left(x_{i,t} \tilde{W}_{t-1} - \frac{x_{i,t-1} y_{i,t-1}}{\langle \mathbf{x}_{t-1}, \mathbf{y}_{t-1} \rangle} \right)^{+}, \end{aligned} \tag{4.1}$$

where $\gamma_s, \gamma_b > 0$ are the proportional transaction fees for selling and buying respectively, W_{t-1} is the wealth before the trading costs are taken into account, $\tilde{W}_{t-1}$ is the wealth remaining after the trading costs, and $(x)^+$ is defined as the positive part of x as $(x)^+ := \max(x, 0)$. This model for transaction costs is called *proportional transaction costs*. There is no work in the scientific literature able to bound theoretically the wealth of an online learning algorithm when this model of costs is adopted.

If we assume that in Equation (4.1) we have $\gamma = \gamma_s = \gamma_b > 0$ equal and fixed for buying ans selling and defining $\alpha_t := \frac{\tilde{W}_{t-1}}{W_{t-1}}$, we can rewrite Equation (4.1) as:

$$\alpha_t = 1 - \gamma ||\mathbf{x}'_{t-1} - \mathbf{x}_t \alpha_t||_1, \tag{4.2}$$

where $\mathbf{x}'_{t-1} = \frac{\mathbf{x}_{t-1} \otimes \mathbf{y}_{t-1}}{\langle \mathbf{x}_{t-1}, \mathbf{y}_{t-1} \rangle}$ is the portfolio composition after the market movement $\mathbf{y}_{t-1}$. With $\mathbf{a} \otimes \mathbf{b}$ we denote the element-wise product between the two vectors $\mathbf{a}$ and $\mathbf{b}$.

With this model, the wealth that takes into account transaction costs can be written as:

$$\tilde{W}_T = \prod_{t=1}^{T} \alpha_t \langle \mathbf{x}_t, \mathbf{y}_t \rangle, \tag{4.3}$$

where α_t is the solution of Equation (4.2). We simplify further Equation (4.2) to avoid having to work with a non-linear equation. Indeed, if we assume that the components of $\mathbf{y}_t$ are small, we can assume $\mathbf{x}'_t \approx \mathbf{x}_t$ and $\alpha_t \mathbf{x}_t \approx \mathbf{x}_t$. Therefore, the ratio of the wealth remaining after the trading costs can be approximated by:

$$\alpha_t = 1 - \gamma ||\mathbf{x}_t - \mathbf{x}_{t-1}||_1. \tag{4.4}$$

We will now define a new concept of regret for the Online Portfolio Optimization, compared to the one originated from the log-loss of Section 3.2. Formally, using the approximation of the cost provided in Equation (4.4) we have:

$$\begin{aligned} \log(\tilde{W}_T) &= \log\left(\prod_{t=1}^{T} \langle \mathbf{x}_t, \mathbf{y}_t \alpha_t \rangle\right) && (4.5) \\ &= \log(W_T) + \log\left(\prod_{t=1}^{T} \alpha_t\right) && (4.6) \\ &\approx \log(W_T) - \sum_{t=1}^{T} \gamma ||\mathbf{x}_t - \mathbf{x}_{t-1}||_1. && (4.7) \end{aligned}$$

The approximation in Equation (4.7) is because $\gamma \ll 1$ and $\log(1-x) \approx -x$.

Hence we can define quantity

$$W_T^C := W_T \exp\left\{-\gamma \sum_{t=1}^{T} ||\mathbf{x}_t - \mathbf{x}_{t-1}||_1\right\}, \tag{4.8}$$

which is the wealth obtained by an algorithm assuming transaction costs given by Equation (4.4).

Note that by using Equation (4.4) we have that the BCRP pays zero transaction costs, and this observation justifies further the use of the following definitions:

Definition 4.1.1. *(Regret on the costs) For the Online Portfolio Optimization with transaction costs problem we define:*

$$C_T := \gamma \sum_{t=1}^{T} ||\mathbf{x}_t - \mathbf{x}_{t-1}||_1, \tag{4.9}$$

as the regret on the costs paid by a learner predicting the sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$ *of portfolio vectors.*

Hence, under this model, the quantity C_T can be interpreted either as the costs paid by the learner or as the gap between the costs paid by the learner and the best expert in the class, which is zero.

Using the previous definition we are now able to introduce the concept of total regret.

Definition 4.1.2. *(Total Regret) For the Online Portfolio Optimization with transaction costs problem we define for an online learning algorithm* $\mathcal{A}$*:*

$$R_T^C(\mathcal{A}) := R_T(\mathcal{A}) + C_T(\mathcal{A}), \tag{4.10}$$

where $R_T(\mathcal{A})$ *is defined as the log-loss regret introduced in Section 3.2 and* $C_T(\mathcal{A})$ *is the regret on the costs defined in Definition 4.1.1.*

We are mainly interested in algorithms that bound the total regret R_T^C, as we believe that this line of research might potentially start to close the bridge between theory and practice in the Online Portfolio Optimization framework.

4.2 Related Works in Online Learning and Switching Costs

Finally, it is worth noting that the problem of dealing with transaction costs has also been tackled in sequential decision-making settings similar to the Online Portfolio Optimization one, *i.e.*, in the expert and bandit learning [Li et al., 2018b, Cesa-Bianchi et al., 2013, Trovò et al., 2016] and the Metrical Task Systems literature [Lin et al., 2012], where the notion of regret has been extended to include the cost of changing the prediction of the algorithm over time. This algorithms cannot be applied directly to the problem of Online Portfolio Optimization, because their setting is essentially notably different. For example in [Li et al., 2018b] the authors are concerned with Online Learning in applications where the outcomes are very predictable (*e.g.*, energy consumption) and hence they assume to have knowledge of the n future outcomes. This assumption is clearly not met in general in financial assets. On the other hand, the Multi Arm Bandit framework assumes that the learning agent has knowledge of the loss function only for the the action taken at the previous step, and cannot compute the loss associated with the other actions. Nonetheless, in [Ito et al., 2018] the authors suggested that the partial observability of the Multi Armed Bandit framework could be used to model illiquid assets such as the real estate market.

Chapter 5

Algorithms for the Online Portfolio Optimization Problem

In this section we will review the state of the art algorithms for the Online Portfolio Optimization problem and discuss their theoretical guarantees, and how these algorithms can be generated by the theoretical framework of Online Learning with expert advice and Online Optimization we described in Chapter 2.

The setting is the one described in Section 3.2.2, in particular $\Delta = \Delta_{N-1} \subset \mathbb{R}^N$ is the N-simplex, and an element $\mathbf{x}_t \in \Delta$ describes the allocation over N stocks for the t-th period.

As is commonly done in the portfolio allocation literature [Agarwal et al., 2006], we assume that the price of the assets does not change too much during two consecutive rounds, or, formally:

Assumption 1. *There exist two finite constants* $\epsilon_l, \epsilon_u \in \mathbb{R}^+$ *s.t. the price relatives* $y_{j,t} \in [\epsilon_l, \epsilon_u]$, *with* $0 < \epsilon_l \leq \epsilon_u < +\infty$, *for each round* $t \in \{1, \dots, T\}$ *and each asset* $j \in \{1, \dots, N\}$.

Notice that under Assumption 1 it is possible to bound the L_1, L_2 and the L_∞ gradient of the loss as follows:

check

$$||\nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)||_1 \leq \frac{N\epsilon_u}{\epsilon_l} := G_1, \tag{5.1}$$

$$||\nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)||_2 \leq \frac{\epsilon_u \sqrt{N}}{\epsilon_l} := G_2, \tag{5.2}$$

$$||\nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)||_\infty \leq \frac{\epsilon_u}{\epsilon_l} := G_\infty. \tag{5.3}$$

Since we will compare multiple algorithms, we introduce the notation $R_T(\mathcal{A})$ when speaking about the regret at time T of an online learner $\mathcal{A}$. The same notation applies with the total regret R_T^C or the regret on the costs C_T, defined in Section 4.

5.1 Algorithm with regret bound

As already pointed out, most algorithms in the Online Portfolio Optimization literature do not consider transaction costs and have guarantees only on the standard regret R_T. In this section we will summarize the most relevant algorithms for the Online Portfolio Optimization problem that have been proven to have only bounded regret R_T.

5.1.1 Universal Portfolios

The Universal Portfolios (UP) [Cover and Ordentlich, 1996] algorithm has been one of the first algorithms that have been introduced in the framework of Online Portfolio Optimization. The UP algorithm has the best theoretical guarantees among the algorithm for Online Portfolio Optimization, as it can reach the minmax value of the game between the adversarial environment and the learning agent (Theorem 10.2 [Cesa-Bianchi and Lugosi, 2006]).

Definition 5.1.1. *(Universal Portfolios). The prediction of the UP algorithm is the following:*

$$\mathbf{x}_{t+1} = \frac{\int_\Delta \mathbf{x} W_t(\mathbf{x}) d\mathbf{x}}{\int_\Delta W_t(\mathbf{x}) d\mathbf{x}}. \tag{5.4}$$

Note that this algorithm is the Continuous Mixture Forecaster for exp-concave losses, described in Section 2.3, since the logarithmic loss is exp-concave with $\nu = 1$. as described in the analysis of Section 3.1.1.

Hence, we have that:

$$R_T(UP) \leq (N-1)\log(T+1). \tag{5.5}$$

Clearly the UP algorithm is computationally hard as it involves integration over a N-simplex. Indeed, there is an extensive research that looks into efficient implementations of the UP algorithm [Kalai and Vempala, 2002].

Moreover, the update rule in Equation (5.4) can be generalized as follows:

$$\mathbf{x}_{t+1} = \frac{\int_\Delta \mathbf{x} W_t(\mathbf{x})\mu(\mathbf{x})d\mathbf{x}}{\int_\Delta W_t(\mathbf{x})\mu(\mathbf{x})d\mathbf{x}}, \tag{5.6}$$

where $\mu(\mathbf{x})$ is a distribution over Δ_{N-1}, the standard UP algorithm is obtained by choosing μ as the uniform distribution over the probability simplex, but there are choices of $\mu(\mathbf{x})$ for which we can obtain slightly better constants for the regret bound.

5.1.2 Exponential Gradient

The Exponential Gradient (EG) algorithm is a specification of the OMD algorithm described in Section 2.5.1, by using the Kullback–Leibler divergence $d_\psi(\mathbf{x}, \mathbf{y}) = KL(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} x_i \log(x_i/y_i)$ as the Bregman divergence the, and $\eta_t = \eta$ as the constant sequence of learning rates. The update rule for EG in this case becomes:

Definition 5.1.2. *(Exponential Gradient). The EG algorithm is defined by the following update rule:*

$$\mathbf{x}_{t+1} = \operatorname*{arg\,inf}_{\mathbf{x}\in\Delta_{N-1}} \left\{ KL(\mathbf{x}, \mathbf{x}_t) - \eta_t \left\langle \frac{\mathbf{x}_t}{\langle \mathbf{x}_t, \mathbf{y}_t \rangle}, \mathbf{x} - \mathbf{x}_t \right\rangle \right\}. \tag{5.7}$$

The update rule in Equation (5.7) can be solved analytically [Helmbold et al., 1998], giving the following closed update

$$x_{i,t+1} = \frac{x_{j,t} \exp\left(\eta_t y_{j,t} / \langle \mathbf{x}_t, \mathbf{y}_t \rangle\right)}{\sum_{j=1}^{N} x_{j,t} \exp\left(\eta_t y_{j,t} / \langle \mathbf{x}_t, \mathbf{y}_t \rangle\right)}, \forall i \in 1, \ldots, N. \tag{5.8}$$

This update rule is also a Weighted Average Forecaster described in Section 2.1.2, and in particular, it is a special case of Exponentially Weighted Forecaster of Definition 2.1.3. This is useful for proving the following theorem.

Theorem 5.1.1. *(Regret Bound for the Exponential Gradient Algorithm). The EG algorithm defined by the update rule in Equation* (5.8) *has the following regret bound:*

$$R_T(EG) \leq \frac{\epsilon_u}{\epsilon_l} \sqrt{\frac{T \log N}{2}}. \tag{5.9}$$

Proof. We know that $\psi(\mathbf{x}) = \sum_{i=1}^{N} x_i \log(x_i)$ is 1-strong convex with respect to the L_1 norm $||\cdot||_1$ [Shalev-Shwartz and Singer, 2007], and so we have that $KL(\mathbf{x}, \mathbf{x}_t) \geq \frac{1}{2}||\mathbf{x} - \mathbf{x}_t||_1$.

Moreover, we can bound the L_1 diameter D_1 of the simplex Δ_{N-1} as:

$$D_1 = \sup_{\mathbf{x},\mathbf{y}\in\Delta_{N-1}} ||\mathbf{x}-\mathbf{y}||_1 \leq 2.$$

Therefore, we can apply Theorem 2.5.1 with $\eta = \frac{1}{G_\infty}\sqrt{\frac{2\log N}{T}}$ and $\mathbf{x}_1 = (1/N, \dots, 1/N)$ giving as a result the thesis. □

Note that one could also obtain a regret bound by using the fact that the EG algorithm is a specialization of OMD with $\psi(\mathbf{x}) = \sum_{i=1}^{N} x_i \log(x_i)$.

5.1.3 Online Newton Step

The Online Newton Step (ONS) [Hazan et al., 2007] algorithm is one of the few algorithms other than the UP one, that guarantees a logarithmic bound $R_T(ONS) = \mathcal{O}(\log T)$. The method uses second order information of the loss function, but it can nonetheless be stated into first order method such as OMD, as we will discuss at the end of this section.

Definition 5.1.3. *(Online Newton Step). The ONS algorithm is defined by the following update rule:*

$$\mathbf{x}_{t+1} = \Pi^{A_t}_{\Delta_{N-1}}\left(\mathbf{x}_t + \frac{1}{\beta}A_t^{-1}\frac{\mathbf{y}_t}{\langle \mathbf{x}_t, \mathbf{y}_t\rangle}\right), \tag{5.10}$$

where $\prod^{A_t}_{\Delta_{N-1}}(\cdot)$ *is the non-standard projection onto the simplex* Δ_{N-1} *defined as*

$$\Pi^{A_t}_{\Delta_{N-1}}(\mathbf{x}_0) := \operatorname*{arg\,inf}_{\mathbf{x}\in\Delta_{N-1}} \langle \mathbf{x}-\mathbf{x}_0, A_t(\mathbf{x}-\mathbf{x}_0)\rangle, \tag{5.11}$$

and the matrix $A_t \in \mathbb{R}^{N\times N}$ *is defined as*

$$A_t = \sum_{s=1}^{t} \nabla f_t(\mathbf{x}_t)\nabla f_t(\mathbf{x}_t)^T + \epsilon\mathbb{I}_N, \tag{5.12}$$

, where $\mathbb{I}_N$ *si the identity matrix in* $\mathbb{R}^N$*.*

The idea for the ONS algorithm is originated from the concept of strong convexity, that is defined as follow:

Definition 5.1.4. *(Strong Convexity). A function* $f : \mathcal{D} \to \mathbb{R}$ *is said to be* μ*-strong convex w.r.t. the norm* $||\cdot||$ *if:*

$$f(y) - f(x) \geq \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}||y-x||^2, \forall x, y \in \mathcal{D}, \forall x, y \in \mathcal{D}.$$

Usually there is the correspondence of convex-loss $R_T = \mathcal{O}(\sqrt{T})$ and strong-convex loss $R_T = \mathcal{O}(\log T)$. The idea of the ONS algorithm is to recover a weaker concept of strong convexity for exp-concave losses:

Definition 5.1.5. *(Weak Strong Convexity). A function $f : \mathcal{D} \subset \mathbb{R}^N \to \mathbb{R}$ is said to be weak-strong convex if $\forall x \in \mathcal{D} \exists A \in \mathbb{R}^{N\times N}$ such that:*

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

for a positive defined matrix A that defines the norm $||x||_A^2 = \langle x, Ax\rangle$.

Indeed, for any ν exp-concave function $f : \mathcal{D} \to \mathbb{R}$ with bounded gradient, *i.e.* $||\nabla f(\mathbf{x})||_2 \leq G \ \forall \mathbf{x} \in \mathcal{D}$, with $D = \sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} ||\mathbf{x} - \mathbf{y}||_2$, $\beta = \frac{1}{2}\min\{\nu, \frac{1}{4GD}\}$ and $A = \nabla f(\mathbf{x})\nabla f(\mathbf{x})^T$, we have that:

$$f(\mathbf{y}) - f(\mathbf{x}) \geq \langle \nabla f(x), \mathbf{y} - \mathbf{x}\rangle + \frac{\beta}{2}||\mathbf{y} - \mathbf{x}||_A \forall x, y \in \mathcal{D}. \tag{5.13}$$

The main idea of ONS is exploiting the weak-strong convexity of exp-concave functions to recover $\mathcal{O}(\log T)$ regret bounds. The complete proof can be found in [Hazan et al., 2007].

From Equation (5.13) we can see that the matrix A used by the ONS algorithm is just a lower bound on the Hessian of the loss function. This is also the reason why the projection onto the simplex of the ONS algorithm is the non standard projection defined by the matrix A_t defined in Equation (5.12).

Theorem 5.1.2. *(Regret Bound for the Online Newton Step Algorithm).*

By choosing $\beta = \frac{\alpha}{8\sqrt{N}}$ in Equation (5.10), the regret bound for the ONS algorithm becomes:

$$R_T(ONS) \leq \frac{10N^{3/2}}{\epsilon_l}\log\left(\frac{NT}{\epsilon_l^2}\right) \tag{5.14}$$

ONS can also be seen as a specification of OMD [Luo et al., 2018] by choosing an adaptive regularizer $\psi(\mathbf{x}) = \psi_t(\mathbf{x}) = \frac{1}{2}||\mathbf{x}||_{A_t}^2$ where A_t is defined as $A_t = A_{t-1} + \nabla f_t(\mathbf{x}_t)\nabla f_t(\mathbf{x}_t)^T$, for a positive defined A_0. In this case the gradient of the Fenchel Conjugate becomes $\nabla\psi_t^*(\mathbf{x}) = \Pi_{\Delta_{N-1}}^{A_t}(\mathbf{x})$, defined in Equation (5.11).

5.2 Algorithm with total regret bound

To the best of our knowledge there are only two works that bound the total regret R_T^C defined in Chapter 4. We will present the works and discuss their limitations, that we tried to solve with our approach.

5.2.1 Online Lazy Updates

Online Lazy Updates (OLU) [Das et al., 2013] is an algorithm designed to minimize explicitly the total regret R_T^C. The origin of this algorithm has to be traced back to a generalization of the OMD algorithm discussed in Section 2.5.1. Namely, the generalization of the OMD algorithm that we are referring to is the Composite Objective Mirror Descent (COMID) algorithm [Duchi et al., 2010]. The idea behind the COMID algorithm is to have a composite loss function of the kind $g_t(\mathbf{x}) = f_t(\mathbf{x}) + r(\mathbf{x})$, then the algorithm linearizes the first term $f_t(\mathbf{x})$ of the composite loss (as in OMD) but does not linearize the second term $r(\mathbf{x})$ of the composite loss $g_t(\mathbf{x})$. Both terms of the loss function, f_t and r, are assumed to be convex.

Definition 5.2.1. *(Composite Objective Mirror Descent). The COMID algorithm is defined with the following update equation:*

$$\mathbf{x}_{t+1} = \underset{\mathbf{x}\in\Delta_{M-1}}{\arg\inf}\left\{\eta\langle\nabla f_t(\mathbf{x}_t), \mathbf{x}\rangle + \eta\ r(\mathbf{x}) + d_\psi(\mathbf{x}, \mathbf{x}_t)\right\}, \tag{5.15}$$

where d_ψ is the Bregman divergence for a convex function ψ.

A lemma similar to Lemma 2.5.1 gives the following guarantees to the regret of a learner using COMID:

Lemma 5.2.1. *([Duchi et al., 2010] Theorem 2.2) $\forall\mathbf{x} \in \Delta_{N-1}$ and for a sequence $\{\mathbf{x}_t\}_{t=1}^T$ defined by the update rule (5.15), we have:*

$$\eta\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}) + r(\mathbf{x}_t) - r(\mathbf{x})] \leq d_\psi(\mathbf{x}, \mathbf{x}_t) + \eta r(\mathbf{x}_1) + \frac{\eta^2}{2\alpha}\sum_{t=1}^{T}||\nabla f_t(\mathbf{x}_t)||_*^2, \tag{5.16}$$

where α is the parameter that ensures $d_\psi(\mathbf{x}, \mathbf{y}) \geq \frac{\alpha}{2}||\mathbf{x} - \mathbf{y}||^2$.

This lemma implies a regret bound on R_T. If we assume that the losses f_t have bounded gradient by G_* under the norm $||\cdot||_*$ then we have that:

$$\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}) + r(\mathbf{x}_t) - r(\mathbf{x})] \leq \frac{1}{\eta}d_\psi(\mathbf{x}, \mathbf{x}_t) + r(\mathbf{x}_1) + \frac{T\eta}{2\alpha}G_*^2. \tag{5.17}$$

Consequently, taking $\eta = \frac{K}{\sqrt{T}}$, and assuming $d_\psi(\mathbf{x}, \mathbf{y}) \leq D\ \forall\mathbf{x}, \mathbf{y} \in \Delta_{N-1}$, and $r(\mathbf{x}_1) \leq D_1$, we obtain:

$$\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}) + r(\mathbf{x}_t) - r(\mathbf{x})] \leq KD\sqrt{T} + D_1 + \frac{\sqrt{T}}{2\alpha}G_*^2. \tag{5.18}$$

The idea of OLU is to take $r = r_t(\mathbf{x}) = \gamma||\mathbf{x} - \mathbf{x}_{t-1}||_1$ [Das, 2014], $\psi = ||\mathbf{x}||_2^2$ generating the following update rule:

Definition 5.2.2. *(Online Lazy Update). The OLU algorithm is defined by the following update rule:*

$$\mathbf{x}_{t+1} = \underset{\mathbf{x}\in\Delta_{M-1}}{\arg\inf}\left\{-\eta\log(\langle\mathbf{x},\mathbf{y}_t\rangle) + \eta\gamma||\mathbf{x}_t - \mathbf{x}||_1 + \frac{1}{2}||\mathbf{x} - \mathbf{x}_t||_2^2\right\}. \tag{5.19}$$

Note that there are multiple definitions of the OLU algorithm, and we reported a version in which the first term of the loss has not been linearized. Linearization of the first term of the loss with $\langle\nabla f_t(\mathbf{x}_t), \mathbf{x}\rangle$ would result in the same update rule and same analysis (since the loss f_t is convex).

With these specifications we obtain the result from Equation (5.18):

$$\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}) + \gamma||\mathbf{x}_t - \mathbf{x}_{t-1}||_1 - \gamma||\mathbf{x} - \mathbf{x}_{t-1}||_1] \leq \left(\frac{1}{K} + \frac{NK\epsilon_u^2}{2\epsilon_l^2}\right)\sqrt{T}. \tag{5.20}$$

Then, taking to the left hand side the terms $\gamma||\mathbf{x} - \mathbf{x}_{t-1}||_1$, and specializing $f_t(\mathbf{x})$ as the log-loss defined for the Online Portfolio Optimization framework, we obtain:

$$\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}) + \gamma||\mathbf{x}_t - \mathbf{x}_{t-1}||_1] \leq \sum_{t=1}^{T}\gamma||\mathbf{x} - \mathbf{x}_{t-1}||_1 + \left(\frac{1}{K} + \frac{NK\epsilon_u^2}{2\epsilon_l^2}\right)\sqrt{T}. \tag{5.21}$$

Now the left hand side is equivalent to our Definition 4.1.2 of total regret R_T^C. Note that we do not have a sub-linear bound for the total regret yet. In order to recover the sub-linear bound on the total regret R_T^C in [Das, 2014] (Theorem 1) the authors assume $\gamma = \frac{\gamma_0}{\sqrt{T}}$. With this assumption we can recover the following bound on the total regret for the OLU algorithm:

Theorem 5.2.1 (Total Regret of OLU [Das, 2014])**.** *If Assumption 1 holds, the OLU algorithm with* $\eta = \frac{K}{\sqrt{T}}$, $\forall K \in \mathbb{R}^+$, *and* $\gamma = \frac{\gamma_0}{\sqrt{T}}$ *has a total regret of:*

$$R_T^C(OLU) \leq 2\gamma_0\sqrt{T} + \left(\frac{1}{K} + \frac{NK\epsilon_u^2}{2\epsilon_l^2}\right)\sqrt{T}. \tag{5.22}$$

It is clear from our discussion on the model for Online Portfolio Optimization with transaction costs described in Chapter 4 that $\gamma > 0$ is fixed and independent on the time horizon T of the investment process.

F: Scriviamola prima questa cosa di γ costante. M: mi smebra ovvia e quinid non saprei come gistificare di scriverla.

5.2.2 Implementation of Online Lazy Update

Due to the non-smooth L_1 term in Equation (5.19), we need a special optimization procedure. The authors proposed the Alternating Direction Method of Multipliers (ADMM) scheme [Boyd et al., 2011], by decoupling the non-smooth term $||\mathbf{x}_t - \mathbf{x}||_1$ from the rest of the objective function. Indeed, Equation (5.19) is equivalent to

$$\mathbf{x}_{t+1} = \underset{\mathbf{x}\in\Delta,\mathbf{x}-\mathbf{x}_t=\mathbf{z}}{\arg\min} \left\{ -\eta \log(\langle \mathbf{x}, \mathbf{y}_t \rangle) + \eta\gamma||\mathbf{z}||_1 + \frac{1}{2}||\mathbf{x} - \mathbf{x}_t||_2^2 \right\} \tag{5.23}$$

The ADMM method is concerned with optimization problems of the kind

$$\begin{cases} \inf f(\mathbf{x}) + g(\mathbf{z}) \\ s.t.\ A\mathbf{x} + B\mathbf{z} = \mathbf{c}, \end{cases} \tag{5.24}$$

where $\mathbf{x}, \mathbf{z}, \mathbf{c} \in \mathbb{R}^N$ and $A, B \in \mathbb{R}^N$.

Problem (5.24) has augmented Lagrangian:

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{x}) + \langle \mathbf{y}, A\mathbf{x} + B\mathbf{z} - \mathbf{c} \rangle + \frac{\rho^2}{2}||A\mathbf{x} + B\mathbf{z} - \mathbf{c}||_2^2. \tag{5.25}$$

Now ADMM solves the Lagrangian problem by iterating over minimization on the primal variables and then doing a dual update (this justifies the name *alternating direction* in ADMM) with the following update rules:

$$\begin{cases} (\mathbf{x}^{k+1}, \mathbf{z}^{k+1}) = \underset{\mathbf{x},\mathbf{z}}{\arg\inf}\, \mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}^{(k)}) \\ \mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \rho(A\mathbf{x}^{k+1} + B\mathbf{z}^{(k+1)} - \mathbf{c}) \end{cases} \tag{5.26}$$

If we define the residual $\mathbf{r} = A\mathbf{x} + B\mathbf{z} - \mathbf{c}$ and $\mathbf{u} = \frac{1}{\rho}\mathbf{y}$ as the scaled dual variable, then the Lagrangian in Equation (5.25) turns into

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = f(\mathbf{x}) + g(\mathbf{x}) + \frac{\rho}{2}||\mathbf{r} + \mathbf{u}||_2^2 - \frac{\rho}{2}||\mathbf{u}||_2^2 \tag{5.27}$$

So we can rewrite the update Equations (5.26) as reported in Algorithm 2.

Algorithm 2 is know to converge (see [Boyd et al., 2011] Appendix A).

In order to use ADMM to solve the optimization of OLU in Equation (5.19), we have do define the elements in the ADMM algorithm at each time t as:

Algorithm 2 Alternating Direction Method of Multipliers

```
Require: $f, g, A, B, \mathbf{c}, \mathbf{x}^0, \mathbf{z}^0, \mathbf{u}^0, \rho$
while Stopping condition not met: do
    Update the primal variables:
        $$\mathbf{x}^{k+1} = \arg\inf_{\mathbf{x}} \left\{ f(\mathbf{x}) + \frac{\rho}{2} ||A\mathbf{x}^{(k)} + B\mathbf{z}^{(k)} - \mathbf{c} + \mathbf{u}^{(k)}|| \right\}$$
        $$\mathbf{z}^{k+1} = \arg\inf_{\mathbf{z}} \left\{ g(\mathbf{z}) + \frac{\rho}{2} ||A\mathbf{x}^{(k)} + B\mathbf{z}^{(k)} - \mathbf{c} + \mathbf{u}^{(k)}|| \right\}$$
    Update the dual variable:
        $$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + A\mathbf{x}^{k+1} + B\mathbf{z}^{(k+1)} - \mathbf{c}$$
end while
return $\mathbf{x}^k, \mathbf{z}^k$.
```

$$\begin{cases} f(\mathbf{x}) & = -\eta \nabla f_t(\mathbf{x}_t) \\ g(\mathbf{z}) & = \gamma\eta ||\mathbf{z}||_1 \\ \mathbf{z} & = \mathbf{x} - \mathbf{x}_t \\ A & = \mathbb{I}_N \\ B & = -\mathbb{I}_N \\ \mathbf{c} & = \mathbf{x}_t \end{cases}, \tag{5.28}$$

and then use Algorithm 2 to solve Equation (5.19).

5.3 Heuristic Algorithms without Regret Bound

There are also heuristic algorithms designed to exploit some known phenomena in markets. Among these heuristic algorithm we can find Anticor [Borodin et al., 2004], PAMR [Li et al., 2012], OLMAR [Li et al., 2015], and MRTC [Yang et al., 2018], which in some cases outperform the algorithms described above in terms of empirical performance. Remarkably, none of the above algorithms provide guarantees on the regret, and so we will avoid an in-depth description of their mechanism, since we are currently concerned with algorithms that provide theoretical guarantees without assumptions on the distribution of the marker vectors.

Chapter 6

Online Gradient Descent for Online Portfolio Optimization with Transaction Costs

The Online Gradient Descent (OGD) algorithm is one of the first algorithms developed in the field of Online Convex Optimization [Zinkevich, 2003]. We extended its use to the Online Portfolio Optimization framework and proved that the OGD algorithm has many interesting properties, among which a bound on the total regret R_T^C.

Algorithm 3 OGD in Online Portfolio Optimization with Transaction Costs

```
Require: learning rate sequence ${\eta_1, \dots, \eta_T}$
Set $\mathbf{x}_1 \leftarrow \frac{1}{N}\mathbf{1}$
for $t \in \{1, \dots, T\}$ do
    $\mathbf{z}_{t+1} \leftarrow \mathbf{x}_t + \eta_t \frac{\mathbf{y}_t}{\langle \mathbf{y}_t, \mathbf{x}_t \rangle}$
    Select Portfolio $\mathbf{x}_{t+1} = \Pi_{\Delta_{N-1}}(\mathbf{z}_t)$
    Observe $\mathbf{y}_{t+1}$ from the market
    Get wealth $\log(\langle \mathbf{y}_{t+1}, \mathbf{x}_{t+1} \rangle) - \gamma ||\mathbf{x}_{t+1} - \mathbf{x}_t||_1$
end for
```

6.1 Using OGD for Portfolio Optimization

This section describes the adaptation of the OGD algorithm to the Online Portfolio Optimization framework and provides a theoretical analysis of such an algorithm in the presence of transaction costs.

6.1.1 The OGD Algorithm

The definition of the OGD update rule for a generic convex loss function $f_t(\mathbf{x}_t)$ over a generic convex set $\mathcal{D}$ is the following:

$$\mathbf{x}_{t+1} = \Pi_{\mathcal{D}} \left(\mathbf{x}_t - \eta_t \nabla f_t(\mathbf{x}_t) \right), \tag{6.1}$$

where $\Pi_{\mathcal{D}}(y) := \underset{x \in \mathcal{D}}{\arg\inf} \, ||y - x||_2^2$ is the standard projection of the vector y onto $\mathcal{D}$, $\eta_t > 0$ is the learning rate at round t, and ∇ denotes the gradient operator. Recalling that in the Online Portfolio Optimization framework the function to be minimized is the loss $f_t(\mathbf{x}_t) = -\log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)$, the portfolio update rule becomes:

$$\mathbf{x}_{t+1} = \Pi_{\Delta_{N-1}} \left(\mathbf{x}_t + \eta_t \frac{\mathbf{y}_t}{\langle \mathbf{x}_t, \mathbf{y}_t \rangle} \right). \tag{6.2}$$

The pseudo-code corresponding to the OGD algorithm is presented in Algorithm 3. The algorithm starts with a portfolio $\mathbf{x}_1$ equally allocated among the N available assets (Line 1). Then, for each round $t \in \{1, \dots, T\}$ it rebalances the assets according to Equation (6.2), observes the market outcomes $\mathbf{y}_{t+1}$ (Line 3), and gains a per-round wealth, including costs, of $\log(\langle \mathbf{y}_{t+1}, \mathbf{x}_{t+1} \rangle) - \gamma ||\mathbf{x}_{t+1} - \mathbf{x}_t||_1$ (Line 6). The projection in Line 4, can be implemented very efficiently as we will discuss in Section 6.3 with Algorithm 4.

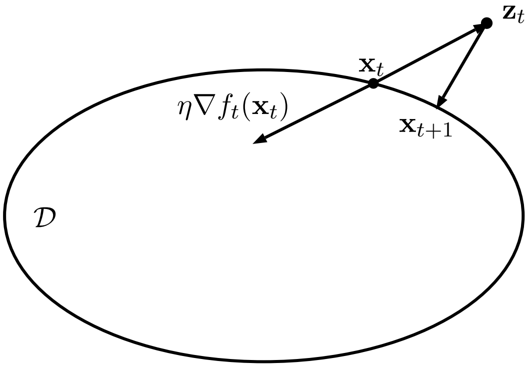


Figure 6.1: Online Gradient Descent.

Note that OGD is an instance of the OMD algorithm described in Section 2.5.1, with $\psi(\mathbf{x}) = ||\mathbf{x}||_2^2$. Indeed the general update Equation (6.1) is equivalent to:

$$\mathbf{x}_{t+1} = \operatorname*{arg\,inf}_{\mathbf{x}\in\Delta} ||\mathbf{x} - \mathbf{x}_t + \eta_t \nabla f_t(\mathbf{x}_t)||_2^2 \tag{6.3}$$

$$= \operatorname*{arg\,inf}_{\mathbf{x}\in\Delta} \left(||\mathbf{x} - \mathbf{x}_t||_2^2 + \eta_t^2 ||\nabla f_t(\mathbf{x}_t)||_2^2 + 2\langle \nabla f_t(\mathbf{x}_t), \mathbf{x} - \mathbf{x}_t\rangle\right) \tag{6.4}$$

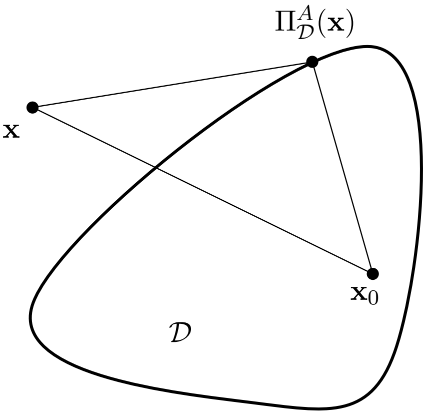


Figure 6.2: Generalized Pythagorean Theorem.

Moreover the following lemma is paramount to prove the regret bound for OGD. This lemma establishes the non expansiveness of the projection operator Π_Δ:

Lemma 6.1.1. *(Generalized Pythagorean Theorem.) Let $\mathcal{D} \in \mathbb{R}^N$ a convex set and $A \in \mathbb{R}^{N\times N}$ a semi-positive defined matrix. Then for any point $\mathbf{x} \in \mathbb{R}^N$ we have:*

$$\langle \mathbf{x} - \mathbf{x}_0, A(\mathbf{x} - \mathbf{x}_0)\rangle \geq \langle \mathbf{z} - \mathbf{x}_0, A(\mathbf{z} - \mathbf{x}_0)\rangle, \forall \mathbf{x}_0 \in \mathcal{D}, \tag{6.5}$$

where $\mathbf{z} = \Pi_{\mathcal{D}}^{A}(\mathbf{x}) = \operatorname*{arg\,inf}_{\mathbf{y}\in\mathcal{D}} \langle \mathbf{y} - \mathbf{x}, A(\mathbf{y} - \mathbf{x})\rangle$.

In the case of $A = \mathbb{I}_{N\times N}$, being $\mathbb{I}_{N\times N}$ the identity matrix in $\mathbb{R}^{N\times N}$, we have that $||\Pi_{\mathcal{D}}(\mathbf{x}) - \mathbf{x}_0||_2^2 \leq ||\mathbf{x} - \mathbf{x}_0||_2^2$. Hence, that the operator $\Pi_\Delta : \mathbb{R}^N \to \mathcal{D}$ is non-expansive. Figure 6.2 depicts Lemma 6.1.1.

6.2 Regret Analysis

6.2.1 OGD Regret on the Wealth

We recall that, for a generic convex function $f_t(x)$, it has been shown in [Belmega et al., 2018] that $R_T(OGD) = \mathcal{O}(\sqrt{T})$ if the loss function $f_t(x)$ is convex, as in our case. We follow the proof in [Zinkevich, 2003] to derive the specific result for the regret of OGD in the Online Portfolio Optimization framework:

Theorem 6.2.1. *If Assumption 1 holds, the OGD algorithm with* $\eta_t = \frac{K}{\sqrt{t}}$*,* $\forall K \in \mathbb{R}^+$ *has a regret on the wealth of:*

$$R_T(OGD) \leq \left(\frac{1}{K} + \frac{NK\epsilon_u^2}{\epsilon_l^2}\right)\sqrt{T}.$$

Proof. Notice that the L_2 diameter of a simplex Δ_{N-1} is $D = \sqrt{2}$ for any N and that, under Assumption 1, it is possible to bound the gradient of the loss as follows:

$$||\nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)||_2 \leq \frac{\epsilon_u \sqrt{N}}{\epsilon_l} := G_2. \tag{6.6}$$

Given the update in Equation (6.2) for the OGD algorithm, we have:

$$\begin{aligned} ||\mathbf{x}_{t+1} - \mathbf{x}^*||_2^2 = &||\Pi_{\Delta_{N-1}}(\mathbf{x}_t + \eta_t \nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)) - \mathbf{x}^*||_2^2 \\ \leq &||\mathbf{x}^* - \mathbf{x}_t||_2^2 - 2\eta_t \langle \mathbf{x}_t - \mathbf{x}^*, \nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)\rangle \\ &+ \eta_t^2 ||\nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)||_2^2, \end{aligned} \tag{6.7}$$

where we used the fact that the projection operator $\Pi_{\Delta_{N-1}}(\cdot)$ is non-expansive (Lemma 6.1.1). Rearranging the terms, we have:

$$\langle \mathbf{x}^* - \mathbf{x}_t, \nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)\rangle \leq \frac{1}{2\eta_t}\left(||\mathbf{x}_t - \mathbf{x}^*||_2^2 - ||\mathbf{x}_{t+1} - \mathbf{x}^*||_2^2\right) + \frac{\eta_t}{2} G_2^2.$$

Using the above inequality and the convexity of the logarithm, we bound the regret $R_T(OGD)$ as follows:

$$\begin{aligned}
R_T(ODG) &= \sum_{t=1}^{T} \log(\langle \mathbf{x}^*, \mathbf{y}_t \rangle) - \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle) \\
&\leq \sum_{t=1}^{T} \langle \mathbf{x}^* - \mathbf{x}_t, \nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle) \rangle \\
&\leq \sum_{t=1}^{T} \left[\frac{1}{2\eta_t} \left(||\mathbf{x}_t - \mathbf{x}^*||_2^2 - ||\mathbf{x}_{t+1} - \mathbf{x}^*||_2^2 \right) + \frac{\eta_t}{2} G_2^2 \right] \\
&\leq \frac{D^2}{2\eta_1} + \frac{D^2}{2} \sum_{t=2}^{T} \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) + \sum_{t=1}^{T} \frac{\eta_t G_2^2}{2} \\
&= \frac{D^2}{2\eta_T} + \sum_{t=1}^{T} \frac{\eta_t G_2^2}{2} \leq \left(\frac{D^2}{2K} + G_2^2 K \right) \sqrt{T},
\end{aligned}$$

where, for the last inequality, we used that $\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leq 2\sqrt{T}$. By plugging the expression of the diameter D and the L_2 bound on the gradient G_2, we conclude the proof. $\square$

6.2.2 OGD Regret on the Costs

In the following theorem, using techniques similar to the ones in [Andrew et al., 2013], we bound the transaction costs $C_T(OGD)$ of the OGD algorithm in the Online Portfolio Optimization framework:

Theorem 6.2.2. *If Assumption 1 holds, the OGD algorithm with* $\eta_t = \frac{K}{\sqrt{t}}$, $\forall K \in \mathbb{R}^+$ *has a regret on the costs of:*

$$C_T(OGD) \leq \frac{2NK\gamma\epsilon_u}{\epsilon_l} \sqrt{T}.$$

Proof. Recall that, in this setting, the regret on the costs $C_T(OGD)$ is equivalent to the sum of the costs incurred by the OGD algorithm, since the best CRP incurs in no costs. Therefore, we have:

$$
\begin{aligned}
C_T(OGD) &= \gamma \sum_{t=1}^{T-1} ||\mathbf{x}_{t+1} - \mathbf{x}_t||_1 && (6.8)\\
&\leq \gamma \sum_{t=1}^{T-1} \sqrt{N} ||\mathbf{x}_{t+1} - \mathbf{x}_t||_2 && (6.9)\\
&\leq \gamma \sum_{t=1}^{T-1} \sqrt{N} ||\eta_t \nabla \log(\langle \mathbf{x}_t, \mathbf{y}_t \rangle)||_2 && (6.10)\\
&\leq \gamma \sqrt{N} G_2 \sum_{t=1}^{T-1} \eta_t && (6.11)\\
&\leq 2\gamma G_2 K \sqrt{NT}, && (6.12)
\end{aligned}
$$

where we used the equivalence of the norms in $\mathbb{R}^N$ for the inequality in Equation (6.9), the fact that the projection operator $\Pi_\Delta(\cdot)$ is non-expansive and the update formula for OGD to derive Equation (6.10), and the fact that the gradient of the loss is bounded by G_2 in Equation (6.11). Finally, we conclude the proof by substituting the bound on the gradient in Equation (6.6) into Equation (6.12). □

6.2.3 Total Regret

Summarizing the bounds derived in Theorems 6.2.1 and 6.2.2, we obtain the following:

Theorem 6.2.3. *If Assumption 1 holds, the OGD algorithm with* $\eta_t = \frac{K}{\sqrt{t}}$, $\forall K \in \mathbb{R}^+$ *has a total regret of:*

$$R_T^C(OGD) \leq \left[\frac{1}{K} + \frac{NK\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right)\right]\sqrt{T}.$$

If the investment horizon T is known in advance, the learning rate η_t can be tuned to obtain a slightly better upper bound on the total regret:

Corollary 6.2.1. *If Assumption 1 holds, the OGD algorithm with* $\eta_t = \frac{K}{\sqrt{T}}$, $\forall K \in \mathbb{R}^+$ *has a total regret of:*

$$R_T^C(OGD) \leq \left(\frac{1}{K} + \frac{NK\epsilon_u^2}{2\epsilon_l^2}\right)\sqrt{T} + 2\gamma\frac{\epsilon_u}{\epsilon_l}\sqrt{T}. \tag{6.13}$$

Finally, knowing the value of ϵ_l and ϵ_u in Assumption 1, the parameter K can be chosen to minimize the bound in Theorem 6.2.3, giving the following result:

Corollary 6.2.2. *If Assumption 1 holds, the OGD algorithm with* $\eta_t = \frac{1}{\sqrt{t}}\left[\frac{N\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right)\right]^{-\frac{1}{2}}$ *has a total regret of:*

$$R_T^C(OGD) \leq 2\sqrt{\frac{N\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right)T}.$$

In what follows, we compare the theoretical guarantees of OGD in terms of computational complexity and regret with OLU and U_CP, the only algorithms that provide upper bounds to total regret.

6.3 Implementation of the Online Gradient Descent Algorithm

The OGD algorithm can be implemented very efficiently, indeed all computations of Algorithm 3 are trivial and lightweight, except for the projection operator $\Pi_{\Delta_{N-1}}$ onto the simplex Δ_{N-1}. In [Duchi et al., 2008] the authors propose the following algorithm to solve the following optimization problem:

$$\Pi_{\Delta_{N-1}}(\mathbf{x}_0) = \underset{\mathbf{x}\in\Delta_{N-1}}{\arg\inf}\ \frac{1}{2}||\mathbf{x} - \mathbf{x}_0||_2^2 \tag{6.14}$$

Algorithm 4 Near Linear Time Projection Onto The Probability Simplex

```
Require: $\mathbf{z} \in \mathbb{R}^N$.
1: Sort $\mathbf{z}$ into $z_1 \geq z_2 \geq \ldots \geq z_N$.
2: Set $K \leftarrow \max\left\{ j = 1, \ldots, N \,\middle|\, z_j - \frac{1}{j}\left(\sum_{k=1}^{j} z_k - 1\right) > 0 \right\}$.
3: Set $\theta = \frac{1}{K}\left(\sum_{i=1}^{K} z_i - 1\right)$.
4: Set $w_i \leftarrow (z_i - 1)^+,\ \forall i \in 1, \ldots, N$.
5: return $\mathbf{w} = (w_1, \ldots, w_N)$.
```

The procedure is Near-Linear since in Line 1, we need to sort the input vector, that is known to be of $O(N \log N)$ complexity. Hence Algorithm 4 is a $O(N \log N)$ procedure of projecting a $\mathbb{R}^N$ vector onto the probability simplex Δ_{N-1}. Note that Algorithm 4 can be refined to be of $O(N)$ complexity if we avoid to sort the vector, this can be one as shown in [Duchi et al., 2008]. This shows that, OGD is able to handle data streams that come at higher frequencies, *e.g.*, the ones required by some specific financial applications [Abernethy and Kale, 2013].

6.4 Discussion on the Regret Bound

In this section we will discuss some advantages of the OGD algorithm among the algorithm that bound the total regret R_T^C defined in Definition 4.1.2.

As discussed in Section 5.2.1, the OLU algorithm is the only algorithm competing with OGD, in terms of theoretical guarantees on the total regret.

Assuming to know *a priori* the time horizon T and under Assumption 1, the authors of OLU provided the following guarantee on the total regret described in Theorem 5.2.1. Notice that the OLU algorithm achieves a regret of $\mathcal{O}(\sqrt{T})$ only if the transaction rate $\gamma \propto \frac{1}{\sqrt{T}}$, *i.e.*, if the transaction rate decreases over time. We observe that the first term of the r.h.s. of Equation (5.22) is the same as the corresponding one in Equation (6.13): these terms correspond to the regret R_T Instead, if we focus on the second term of the r.h.s. of Equation (5.22) and we assume that γ is constant over the investment horizon T, we would have a total regret of the order of $\mathcal{O}(T)$ for the OLU algorithm. This does not happen to OGD, which, even under these assumptions, would provide a total regret of the order of $\mathcal{O}(\sqrt{T})$. Conversely, if we assume $\gamma \propto \frac{1}{\sqrt{T}}$ as in [Das et al., 2013], the last term in Equation (6.13) would have constant regret on the costs, *i.e.*, $C_T(OGD) \leq \frac{2\epsilon_u NK}{\epsilon_l} = \mathcal{O}(1)$, compared to an order of $\mathcal{O}(\sqrt{T})$ obtained by OLU, which makes OGD strictly better then OLU in terms of total regret bound.

Chapter 7

Numerical Experiments

In this section we analyze the empirical performance of OGD, comparing it with the algorithm from the Online Portfolio Optimization literature that provides guarantees on total regret: OLU [Das et al., 2013]. We also consider UP [Cover and Ordentlich, 1996] and ONS [Agarwal et al., 2006], because UP[1] has the best theoretical guarantees on the regret R_T and ONS because has both good theoretical guarantees on the regret on the wealth R_T and is known to provide the best results regarding the regret on the wealth when analyzed empirically.

Datasets				
Name	Market	Year Span	Rebalances	Assets
NYSE(O)	New York Stock Exchange	1962 - 1984	5651	36
TSE	Toronto Stock Exchange	1994 - 1998	1258	88
SP500	Standard Poor's 500	1998 - 2003	1276	25

Table 7.1: Description of the main datasets used commonly in the Online Portfolio Optimization literature.

Table 7.1 summarizes the datasets used for the experiments. All assets in the datasets are being anonymize to ensure to avoid common bias toward specific assets. To compare the algorithms, we used mainly the NYSE(O) dataset, a well-known benchmark that has been previously used in several portfolio optimization research papers, and notably, in all the works which propose the algorithms here considered as baselines. The NYSE(O) dataset spans 22 years (between 1962 and 1984), for a total investment horizon of

---

[1] We used a naïve version of UP since the classic implementation required an unfeasible amount of time for the experiments. Instead, we discretized the simplex with 10^4 points and used the corresponding CRPs to approximate the integrals used by UP.

$T = 5651$ days (≈ 250 working days per year). In each experiment, we sampled a set of $N = 5$ assets randomly chosen among the 36 and ran the algorithms for the entire investment horizon T. We ran 100 independent experiments for the NYSE(O) datasets, 20 and 50 for the TSE and SP500 dataset respectively, and then we averaged the results. The choice of doing a great number of experiments is to stress the point that we are not concerned with the selection of assets to invest in, but only with the behavior of the algorithms with respect to transaction costs. We considered different values for the transaction rate $\gamma \in \{0, 0.0005, 0.001, 0.003, 0.006, 0.01, 0.02, 0.04\}$, including hight values of γ to simulate highly illiquid markets.

To set the parameter K of OGD, we used the learning rate η_t prescribed by Theorem 6.2.3, with $\epsilon_l = 0.8$ and $\epsilon_u = 1.2$, for which Assumption 1 holds in the dataset NYSE(O). For ONS, we used $\eta = 0$, $\beta = 1$, $\delta = 1/8$, as suggested by the authors in [Agarwal et al., 2006]. We used $\alpha = 0.12$ and $\eta = 1.3$ for OLU, which is the best combination of parameters according to [Das et al., 2013]. All algorithms have been initialized with $\mathbf{x}_1 = \frac{1}{N}\mathbf{1}$.

We used the Annual Percentage Yield (APY) as a metric, assuming 250 working days per year and one update per day. Formally, the APY for the wealth W is defined as:

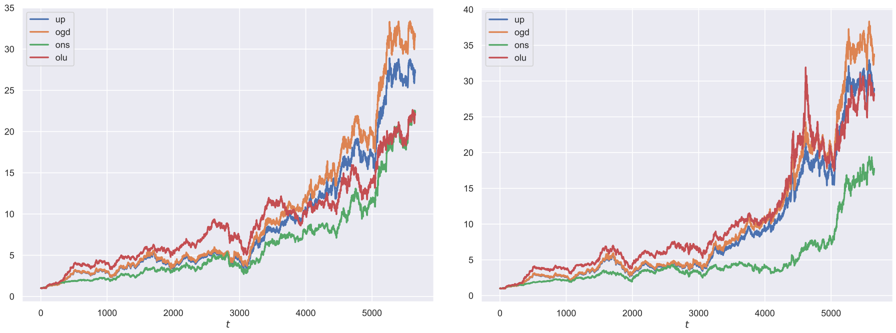
$$A(W) = W^{250/T} - 1,$$

where $W \in \{W_T^C, \tilde{W}_T\}$ and are defined in equation (4.8) and (4.3) respectively. 95% confidence intervals for the mean have been computed with statistical bootstrapping and are depicted as semi-transparent areas.

7.0.1 Results on the NYSE(O) dataset

Figure 7.1 shows the evolution of the total wealth $W_t^C(\mathcal{A})$ of the different algorithms over the investment horizon in two specific runs, one without any cost ($\gamma = 0$) (Figure 7.7a), and one with a transaction rate of $\gamma = 0.001$ (Figure 7.7b). In these two specific runs, OGD obtains a cumulative wealth larger than any other algorithm analyzed, suggesting that, in some settings, it might provide the best performance. The results with $\gamma = 0$ suggest that OGD might be a viable solution even in the absence of costs.

In Figure 7.2, we present the results for the average APY, with the corresponding confidence intervals. In particular, with no transaction costs ($\gamma = 0$), all the analyzed algorithms give similar results. In this setting, ONS is the algorithm with the largest APY. As we increase the transaction rate γ, OGD gets the largest APY, while OLU and ONS seem to be penalized by large transaction costs. Conversely, the fact that the APY decreases from



(a) (b)

Figure 7.1: Wealth $W_T^C(\mathcal{A})$ on two runs of the NYSE(O) for $\gamma = 0$ (a), and $\gamma = 0.001$ (b).

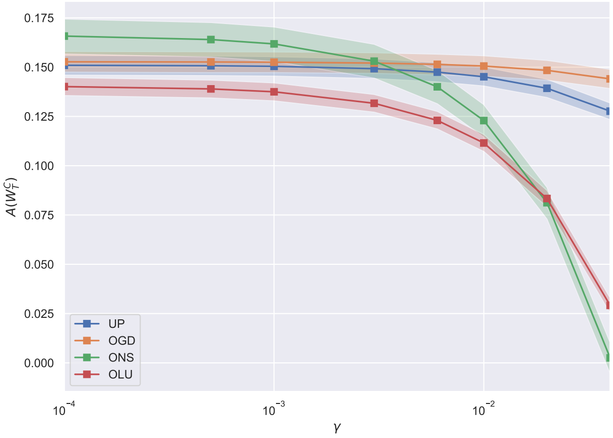


Figure 7.2: Average APY computed on the wealth W_T^C assuming the costs given by $C_T(\mathcal{A})$ for the NYSE(O) dataset.

≈ 0.15 to ≈ 0.14 suggests that OGD is effective at minimizing the costs $C_T(\mathcal{A})$.

Figure 7.3 considers the wealth $\tilde{W}_T(\mathcal{A})$, *i.e.*, the one defined in Equation (4.3). We notice that comparing these results with the ones obtained using W_T^C (Figure 7.2), we have a smaller APY when $\gamma \gg 0$. This suggests that, when applied to real-world cases, they might under-perform w.r.t. what is expected from the theoretical results. In terms of $\tilde{W}_T(\mathcal{A})$, UP seems to perform slightly better than OGD, but the difference is not statistically significant for $\gamma < 0.04$. ONS and OLU provide negative profits ($A(\tilde{W}_T) < 0$) for large values of transaction costs, *e.g.*, for $\gamma = 0.04$ the APY becomes

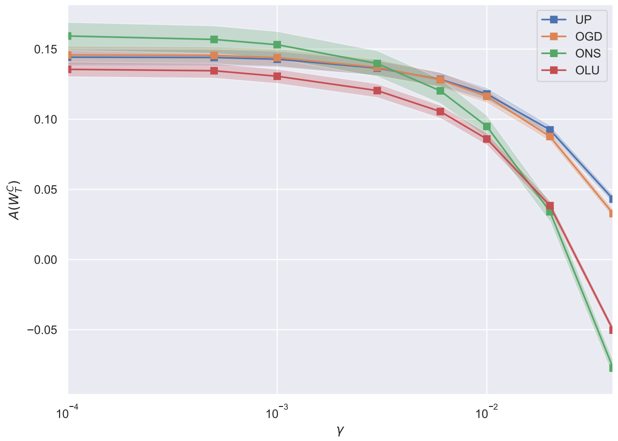


Figure 7.3: Average APY computed on the wealth $\tilde{W}_T(\mathcal{A})$ *assuming the costs given by Equation* (4.2)*, for the NYSE(O) dataset.*

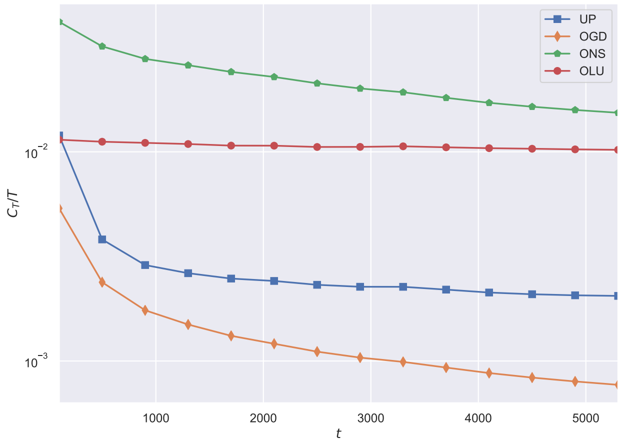


Figure 7.4: Average costs $C_T(\mathcal{A})$ *with* $\gamma = 1$*, for the NYSE(O) dataset.*

negative and, thus, the accumulated wealth is completely canceled out by the transaction costs. From Figure 7.3, we would be inclined to choose ONS for $\gamma \leq 0.003$, and OGD for $\gamma \geq 0.003$.

Figure 7.4 shows the averaged cost per round $C_t(\mathcal{A})/t$ and the corresponding confidence intervals, with $\gamma = 1$ (the value of γ has been chosen to easily interpret how the regret on the costs behaves over time). OGD is the algorithm that provides the lowest cost per round, which strengthens the claim of this work that OGD keeps transaction costs low. The costs per round for OLU are approximately linear, as expected from the theory (see Section 5.2.1). Conversely, the results for ONS, while not having any theoretical guarantee on $C_T(\mathcal{A})/T$, suggest that it has a cost per round of order $\mathcal{O}(\sqrt{T})$, but with a larger constant than OGD. Finally, the costs of UP decrease slower than those of ONS and OGD.

7.0.2 Results on the TSE and SP500 dataset

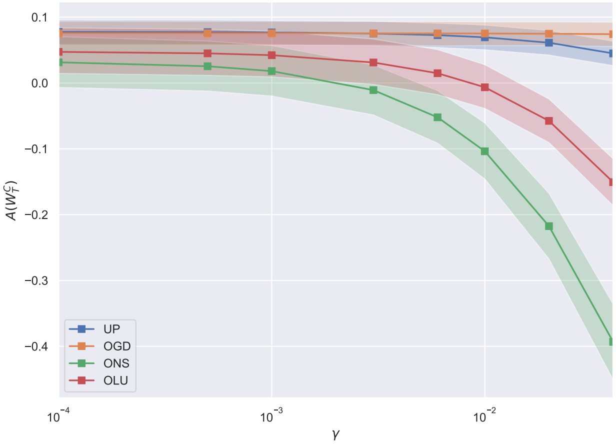


Figure 7.5: Average APY computed on the wealth W_T^C assuming the costs given by $C_T(\mathcal{A})$ for the TSE dataset.

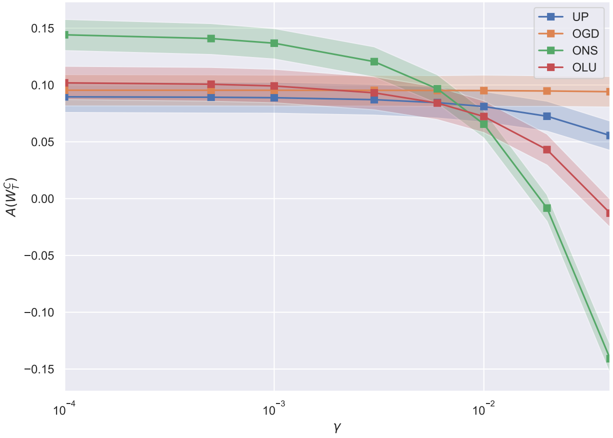


Figure 7.6: Average APY computed on the wealth W_T^C assuming the costs given by $C_T(\mathcal{A})$ for the SP500 dataset.

In Figure 7.5 and 7.6 present the results obtained on the TSE and SP500 datasets respectively, using the same approach we used for the NYSE(O) dataset. The results obtained are in line with the one presented with the NYSE(O) dataset, i.e, the OGD algorithm performs better than the others for transaction rate greater than 0.003, while it presents similar performance, in terms of APY, for smaller values of the transaction rate. Notably, in the SP500 dataset, ONS outperforms the other algorithms form small transaction rate γ, while in the TSE dataset, even for small values of the transaction rate γ, it is out-performed by the other algorithms.

7.0.3 Result on Custom Dataset

In this section we used a custom dataset to test the algorithms. The assets and time period has been choose to reflect a variety of market regimes. We are well aware that a back-testing procedure not rigid enough can lead to over-fitting and biased results, which is a important problem in the financial literature ([Bailey et al., 2016], [Harvey and Liu, 2015]). Conversely to the results presented int eh previous sections, we do not claim the experiments of this section to have any scientific insight other than being illustrative of the behavior of the algorithms on the specific run presented. Nonetheless, we found interesting to present these results.

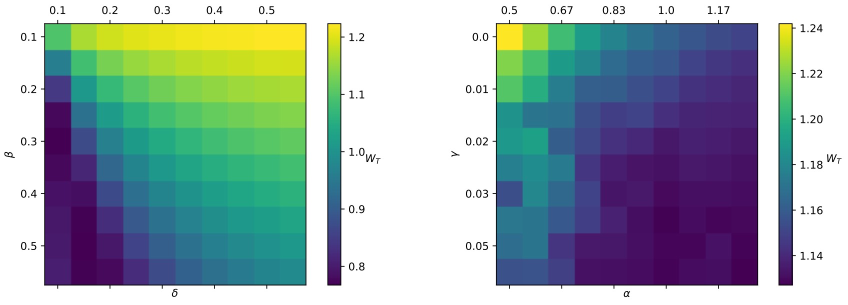
Custom Data (03/29/2019 - 03/29/2020)		
Ticker	Description	Market Category
SPY	SPDR S&P 500 ETF Trust (SPY)	Equity
BNDX	Vanguard Bond Index Fund ETF	Fixed Income
DAX	Global X DAX Germany ETF	Equity
VIX	CBOE Volatility Index	Derivatives

Table 7.2: Detailed description of the custom dataset.

Table 7.2 gives a description of the tailored dataset. We used data for one year (from April 2019 to April 2020), including the period of December 2019 - March 2020 that shows a global decline in the global financial markets. We included two Equity indices (SPY and DAX) as a proxy for the USA markets and European markets, then we included a Bond index (BNDX) and a volatility index (VIX) that simulates a Variance Swap [Bossu, 2006].

To set the parameters of ONS and OLU for the experiments performed on this dataset we used as a validation the first 1/4 of the dataset (corresponding approximately to the first 3 months of the dataset), performed grid search algorithm and found picked the parameter with the highest wealth W_T on the validation set. The results of the grid search are presented in

Figure 7.7. For the OGD algorithm we set the parameter K to minimize the bound according to Theorem 6.2.3.



(a) (b)

Figure 7.7: Grid search for the parameters of ONS (a) and OLU (b) on the validation set of the custom dataset.

Figure 7.8 show the results for the algorithms run on the custom dataset. The transaction rate was set to $\gamma = 0.001$ for all the algorithms. It is interesting to observe the similarity between the wealth of UP and OGD. This can be due to the particular naïve implementation we used for the UP algorithm. It would be interesting to observe if analytically the used approximation of UP would result in the OGD algorithm.

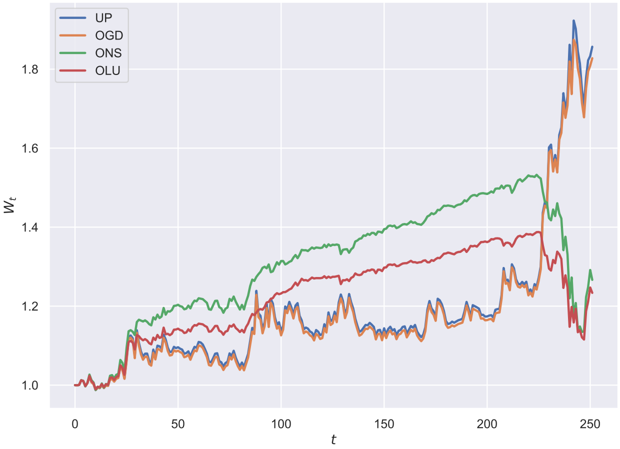


Figure 7.8: Wealth W_t achieved on the custom dataset described in table 7.2, with $\gamma = 0.001$.

Bibliography

[Abernethy and Kale, 2013] Abernethy, J. and Kale, S. (2013). Adaptive market making via online learning. In *NeurIPS*, pages 2058–2066.

[Agarwal et al., 2010] Agarwal, A., Bartlett, P., and Dama, M. (2010). Optimal allocation strategies for the dark pool problem. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 9–16.

[Agarwal et al., 2006] Agarwal, A., Hazan, E., Kale, S., and Schapire, R. (2006). Algorithms for portfolio management based on the newton method. In *ICML*.

[Andrew et al., 2013] 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 *COLT*, pages 741–763.

[Azuma, 1967] Azuma, K. (1967). Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367.

[Bailey et al., 2016] Bailey, D. H., Borwein, J., Lopez de Prado, M., and Zhu, Q. J. (2016). The probability of backtest overfitting. *Journal of Computational Finance, forthcoming.*

[Bailey and Piliouras, 2018] Bailey, J. P. and Piliouras, G. (2018). Multiplicative weights update in zero-sum games. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 321–338.

[Banerjee et al., 2005] Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005). Clustering with bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749.

[Belmega et al., 2018] Belmega, E. V., Mertikopoulos, P., Negrel, R., and Sanguinetti, L. (2018). Online convex optimization and no-regret learning: Algorithms, guarantees and applications. *arXiv preprint arXiv:1804.04529*.

[Black and Scholes, 1973] Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654.

[Blackwell et al., 1956] Blackwell, D. et al. (1956). An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8.

[Blum and Kalai, 1999] Blum, A. and Kalai, A. (1999). Universal portfolios with and without transaction costs. *MACH LEARN*, 35(3):193–205.

[Borodin et al., 2004] Borodin, A. et al. (2004). Can we learn to beat the best stock. In *NeurIPS*, pages 345–352.

[Bossu, 2006] Bossu, S. (2006). Introduction to variance swaps. *Wilmott Magazine*, pages 50–55.

[Bousquet et al., 2003] Bousquet, O., Boucheron, S., and Lugosi, G. (2003). Introduction to statistical learning theory. In *Summer School on Machine Learning*, pages 169–207. Springer.

[Boyd et al., 2011] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122.

[Bubeck et al., 2012] Bubeck, S., Cesa-Bianchi, N., et al. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122.

[Cesa-Bianchi et al., 2013] Cesa-Bianchi, N., Dekel, O., and Shamir, O. (2013). Online learning with switching costs and other adaptive adversaries. In *NeurIPS*, pages 1160–1168.

[Cesa-Bianchi and Lugosi, 2003] Cesa-Bianchi, N. and Lugosi, G. (2003). Potential-based algorithms in on-line prediction and game theory. *Machine Learning*, 51(3):239–261.

[Cesa-Bianchi and Lugosi, 2006] Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games.* Cambridge university press.

[Cover and Ordentlich, 1996] Cover, T. and Ordentlich, E. (1996). Universal portfolios with side information. *IEEE Transactions on Information Theory*, 42(2):348–363.

[Cover, 1966] Cover, T. M. (1966). Behavior of sequential predictors of binary sequences. Technical report, STANFORD UNIV CALIF STANFORD ELECTRONICS LABS.

[Cover and Thomas, 2012] Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

[Das, 2014] Das, P. (2014). *Online convex optimization and its application to online portfolio selection*. PhD thesis, The University of Minnesota.

[Das et al., 2013] Das, P., Johnson, N., and Banerjee, A. (2013). Online lazy updates for portfolio selection with transaction costs. In *AAAI*, pages 202–208.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159.

[Duchi et al., 2008] Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279.

[Duchi et al., 2010] Duchi, J., Shalev-Shwartz, S., Singer, Y., and Tewari, A. (2010). Composite objective mirror descent. In *COLT*, pages 14–26.

[Hakansson et al., 1995] Hakansson, N. H., Ziemba, W. T., et al. (1995). Capital growth theory. *Handbooks in operations research and management science*, 9:65–86.

[Hannan, 1957] Hannan, J. (1957). Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139.

[Harris, 2003] Harris, L. (2003). *Trading and exchanges: Market microstructure for practitioners*. OUP USA.

[Hart and Mas-Colell, 2001] Hart, S. and Mas-Colell, A. (2001). A general class of adaptive strategies. *Journal of Economic Theory*, 98(1):26–54.

[Harvey and Liu, 2015] Harvey, C. R. and Liu, Y. (2015). Backtesting. *The Journal of Portfolio Management*, 42(1):13–28.

[Hazan et al., 2007] Hazan, E., Agarwal, A., and Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192.

[Hazan et al., 2016] Hazan, E. et al. (2016). Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325.

[Helmbold et al., 1998] Helmbold, D. P., Schapire, R. E., Singer, Y., and Warmuth, M. K. (1998). On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347.

[Ito et al., 2018] Ito, S., , Hatano, D., Sumita, H., Yabe, A., Fukunaga, T., Kakimura, N., and Kawarabayashi, K. (2018). Regret bounds for online portfolio selection with a cardinality constraint. In *NeurIPS*, pages 1–10.

[Jiang et al., 2016] Jiang, Y., Yuan, J., and Zeng, M. (2016). A Game Theoretic Study of Enterprise Mergers and Acquisitions: The Case of RJR Nabisco Being Acquired by KKR. *Business and Management Studies*, 2(2):21–33.

[Kalai and Vempala, 2002] Kalai, A. and Vempala, S. (2002). Efficient algorithms for universal portfolios. *Journal of Machine Learning Research*, 3(Nov):423–440.

[Kelly Jr, 2011] Kelly Jr, J. L. (2011). A new interpretation of information rate. In *The Kelly Capital Growth Investment Criterion: Theory and Practice*, pages 25–34. World Scientific.

[Langdon, 1984] Langdon, G. G. (1984). An introduction to arithmetic coding. *IBM Journal of Research and Development*, 28(2):135–149.

[Li et al., 2015] Li, B., Hoi, S., Sahoo, D., and Liu, Z. (2015). Moving average reversion strategy for on-line portfolio selection. *ARTIF INTELL*, 222:104–123.

[Li et al., 2018a] Li, B., Wang, J., Huang, D., and Hoi, S. (2018a). Transaction cost optimization for online portfolio selection. *QUANT FINANC*, 18(8):1411–1424.

[Li et al., 2012] Li, B., Zhao, P., Hoi, S., and Gopalkrishnan, V. (2012). Pamr: Passive aggressive mean reversion strategy for portfolio selection. *MACH LEARN*, 87(2):221–258.

[Li et al., 2018b] Li, Y., Qu, G., and Li, N. (2018b). Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit. *arXiv:1801.07780*.

[Lin et al., 2012] Lin, M., Wierman, A., Roytman, A., Meyerson, A., and Andrew, L. (2012). Online optimization with switching cost. *PERF E R SI*, 40(3):98–100.

[Luo et al., 2018] Luo, H., Wei, C.-Y., and Zheng, K. (2018). Efficient online portfolio with logarithmic regret. In *Advances in Neural Information Processing Systems*, pages 8235–8245.

[Mitchell et al., 1997] Mitchell, T. M. et al. (1997). Machine learning.

[Nash, 1951] Nash, J. (1951). Non-cooperative games. *Annals of mathematics*, pages 286–295.

[O'hara, 1997] O'hara, M. (1997). *Market microstructure theory*. Wiley.

[Poundstone, 2010] Poundstone, W. (2010). *Fortune's Formula: The untold story of the scientific betting system that beat the casinos and wall street*. Hill and Wang.

[Samuelson, 1977] Samuelson, P. A. (1977). St. petersburg paradoxes: Defanged, dissected, and historically described. *Journal of Economic Literature*, 15(1):24–55.

[Shalev-Shwartz et al., 2012] Shalev-Shwartz, S. et al. (2012). Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194.

[Shalev-Shwartz and Singer, 2007] Shalev-Shwartz, S. and Singer, Y. (2007). Online learning: Theory, algorithms, and applications.

[Srebro et al., 2011] Srebro, N., Sridharan, K., and Tewari, A. (2011). On the universality of online mirror descent. In *Advances in neural information processing systems*, pages 2645–2653.

[Sutton et al.,] Sutton, R. S. et al. *Introduction to reinforcement learning*, volume 135.

[Trovò et al., 2016] Trovò, F., Paladino, S., Restelli, M., and Gatti, N. (2016). Budgeted multi-armed bandit in continuous action space. In *ECAI*, pages 560–568.

[Van Erven and Harremos, 2014] Van Erven, T. and Harremos, P. (2014). Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820.

[Weinberger et al., 1994] Weinberger, M. J., Merhav, N., and Feder, M. (1994). Optimal sequential probability assignment for individual sequences. *IEEE Transactions on Information Theory*, 40(2):384–396.

[Yang et al., 2018] Yang, X., Li, H., Zhang, Y., and He, J. (2018). Reversion strategy for online portfolio selection with transaction costs. *INT J APP DECIS SCI*, 11(1):79–99.

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