Introduction to Online Convex Optimization

Second Edition

Elad Hazan

וְהָגִיתָ בּוֹ יוֹמָם וָלָיְלָה
יְהוֹשֻׁעַ א ח

"...you shall research and meditate therein day and night..."
—*Yehoshua 1:8*

To my family: Dana, Hadar, Yoav, Oded and Deluca,
—EH

Contents

Preface

This book serves as an introduction to the expanding theory of online convex optimization (OCO). It was written as an advanced textbook to serve as a basis for a graduate course, and/or as a reference to researchers diving into this fascinating world at the intersection of optimization and machine learning.

Such a course was given at the Technion in 2010–2014, with slight variations from year to year, and later at Princeton University in 2015–2020. The core material in these courses is fully covered in this book, along with exercises that allow students to complete parts of proofs, or that were found illuminating and thought-provoking by those taking the course. Most of the material is given with examples of applications, which are interlaced throughout various topics. These include prediction from expert advice, portfolio selection, matrix completion and recommendation systems, and support vector machine training.

Our hope is that this compendium of material and exercises will be useful to you; the researcher and/or educator.

Placing this Book in the Machine Learning Library

The broad field of machine learning, as in the sub-disciplines of online learning, boosting, regret minimization in games, universal prediction, and other related topics, have seen a plethora of introductory books in recent years. With this note, we can hardly do justice to all of these, but perhaps point to the most related books on the topics of machine learning, learning in games, and optimization, whose intersection is our main focus.

The most closely related book, which served as an inspiration to the current, and indeed an inspiration to the entire field of learning in games, is the wonderful text of Cesa-Bianchi and Lugosi [2006]. From the literature on mathematical optimization theory, there are the numerous introductory

essays to convex optimization and convex analysis, to name only a few [Boyd and Vandenberghe, 2004, Nesterov, 2004, Nemirovski and Yudin, 1983, Nemirovskii, 2004, Borwein and Lewis, 2006, Rockafellar, 1997]. The author fondly recommends the text from which he has learned about mathematical optimization theory [Nemirovskii, 2004]. The more broad texts on machine learning are too numerous to state here.

The primary purpose of this is to serve as an educational textbook for a *dedicated* course on OCO and the convex optimization approach to machine learning. Online convex optimization has already had enough impact to appear in several surveys and introductory texts [Hazan, 2011, Shalev-Shwartz, 2011, Rakhlin, 2009, Rakhlin and Sridharan, 2014]. We hope this compilation of material and exercises will further enrich the literature.

Book's tructure

This book is intended to serve as a reference for a self-contained course for graduate students in computer science/electrical engineering/operations research/statistics and related fields. As such, its organization follows the structure of the course "Decision Analysis", taught at the Technion, and later "Theoretical Machine Learning", taught at Princeton University.

Each chapter should take one or two weeks of classes, depending on the depth and breadth of the intended course. Chapter 1 is designed to be a teaser for the field, and it is less rigorous than the rest of the book.

Roughly speaking, the book can be conceived as three units. The first, from chapter 2 through 4, contains the basic definitions, framework and core algorithms for OCO. Chapters 5 to 7 contain more advanced algorithms and in-depth analysis of the framework and its extensions to other computational and information access models. The rest of the book deals with more advanced algorithms, more difficult settings, and relationships to well-known machine learning paradigms.

This book can assist educators in designing a complete course on the topic of online convex optimization, or it can serve as a component in a comprehensive course on machine learning. A accompanying manual of solutions to selected exercises given in the book is available for educators only.

New in the Second Edition

The main additions to the second edition of this book include the following:

- Expanded coverage of optimization in chapter 2, with a unified gradient descent analysis of the Polyak stepsize.
- Expanded coverage of learning theory in chapter 9, with an introduction to compression and its use in generalization theory.
- An expanded chapter 4, with addition of the exponential weighted optimizer for exp-concave loss functions.
- A revised chapter 5, with the addition of mirror descent analysis, as well as a revised section on adaptive gradient methods.
- New chapter 10 on the notion of adaptive regret and algorithms for OCO with near-optimal adaptive regret bounds.
- New chapter 11 on boosting and its relationship to OCO. Derivation of boosting algorithms from regret minimization.
- New chapter 12 on online boosting.
- New chapter 13 on Blackwell approachability and its strong connection to OCO.

In addition, numerous typos are fixed, exercises are corrected, and solutions to several questions have been made available in a separate manual for educators.

Acknowledgements

The First Version

First of all, I gratefully acknowledge the numerous contributions and insight of the students of the course "decision analysis" given at the Technion during 2010-2014, as well as the students of "theoretical machine learning" taught at Princeton University during 2015-2016.

I would like to thank the friends, colleagues and students that have contributed many suggestions and corrections. A partial list includes: Sanjeev Arora, Shai Shalev-Shwartz, Aleksander Madry, Yoram Singer, Satyen Kale, Alon Gonen, Roi Livni, Gal Lavee, Maayan Harel, Daniel Khasabi, Shuang Liu, Jason Altschuler, Haipeng Luo, Zeyuan Allen-Zhu, Mehrdad Mahdavi, Jaehyun Park, Baris Ungun, Maria Gregori, Tengyu Ma, Kayla McCue, Esther Rolf, Jeremy Cohen, Daniel Suo, Lydia Liu, Fermi Ma, Mert Al, Amir Reza Asadi, Carl Gabel, Nati Srebro, Abbas Mehrabian, Chris Liaw, Nikhil Bansal, Naman Agarwal, Raunak Kumar, Zhize Li, Sheng Zhang, Swati Gupta, Xinyi Chen, Liang Zeng and Kunal Mittal.

I thank Udi Aharoni for his artwork and illustrations depicting algorithms in this book.

I am forever indebted to my teacher and mentor, Sanjeev Arora, without him this book would not be possible.

Finally, I am grateful for the love and support of my wife and children: Dana, Hadar, Yoav, and Oded.

The Second Version

I am most thankful to my students and colleagues that have collaborated with me on research, some of which appears in the second version of this book. Notably my collaborators on boosting methods including Nataly Brukhim, Xinyi Chen, Shay Moran, Naman Agarwal and Karan Singh.

Thanks to my students that helped me proof check new and existing sections: Edgar Minasyan, Paula Gradu, Karan Singh, Nataly Brukhim, Xinyi Chen, Naman Agarwal and Udaya Ghai.

I am thankful to Shay Moran for explaining compression schemes and how they simplify generalization for boosting.

I gratefully acknowledges the help of Ahmed Farah, Charlie Cowen-Breen and the students of "COS 597C: Computational Control Theory" for many helpful suggestions, corrections and solutions to many of the problem sets.

I am very thankful to Wouter Koolen-Wijkstra for a helpful suggestion in the analysis of the Online Newton Step algorithm.

I thank the extremely helpful and rigorous reviewers of this book found by MIT press who gave fantastic suggestions and improve the final manuscript.

As in the first version and even more so, I am grateful for the love and support of my wife and children: Dana, Hadar, Yoav, and Oded.

Elad Hazan
Princeton University

List of Figures

List of Symbols

General

$\stackrel{\text{def}}{=}$	definition
$\arg\min\{\}$	the argument minimizing the expression in braces
$[n]$	the set of integers $\{1, 2, \ldots, n\}$

Geometry and Calculus

$\mathbb{R}^d$	d dimensional Euclidean space
Δ_d	d dimensional simplex, $\{\sum_i \mathbf{x}_i = 1, \mathbf{x}_i \geq 0\}$
$\mathbb{S}$	d dimensional sphere, $\{\|\mathbf{x}\| = 1\}$
$\mathbb{B}$	d dimensional ball, $\{\|\mathbf{x}\| \leq 1\}$
$\mathbb{R}$	real numbers
$\mathbb{C}$	complex numbers
$\|A\|$	determinant of matrix A

Learning Theory

$\mathcal{X}, \mathcal{Y}$	feature/label sets
$\mathcal{D}$	distribution over examples $(\mathbf{x}, y)$
$\mathcal{H}$	hypothesis class in $\mathcal{X} \mapsto Y$
h	single hypothesis $h \in \mathcal{H}$
m	training set size
$\text{error}(h)$	generalization error of hypothesis $h \in \mathcal{H}$

Optimization

$\mathbf{x}$	vectors in the decision set
$\mathcal{K}$	decision set
$\nabla^k f$	the k'th differential of f; note $\nabla^k f \in \mathbb{R}^{d^k}$
$\nabla^{-2} f$	the inverse Hessian of f
∇f	the gradient of f
∇_t	the gradient of f at point $\mathbf{x}_t$
$\mathbf{x}^\star$	the global or local optima of objective f
h_t	objective value distance to optimality, $h_t = f(\mathbf{x}_t) - f(\mathbf{x}^\star)$
d_t	Euclidean distance to optimality $d_t = \|\mathbf{x}_t - \mathbf{x}^\star\|$
G	upper bound on norm of subgradients
D	upper bound on Euclidean diameter
D_p, G_p	upper bound on the p-norm of the subgradients/diameter

Regularization

R	strongly convex and smooth regularization function
$B_R(\mathbf{x}\|\|\mathbf{y})$	R-Bregman-divergence $R(\mathbf{x}) - R(\mathbf{y}) - \nabla R(\mathbf{y})^\top(\mathbf{x} - \mathbf{y})$
G_R	upper bound on norm of (sub)gradients
D_R^2	squared R diameter $\max_{\mathbf{x},\mathbf{y}\in\mathcal{K}}\{R(\mathbf{x}) - R(\mathbf{y})\}$
$\|\mathbf{x}\|_A^2$	squared matrix norm $\mathbf{x}^\top A\mathbf{x}$
$\|\mathbf{x}\|_{\mathbf{y}}^2$	local norm according to local regularization $\mathbf{x}^\top\nabla^2 R(\mathbf{y})\mathbf{x}$
$\|\mathbf{x}\|^*$	dual norm to $\|\mathbf{x}\|$

Chapter 1

Introduction

This book considers *optimization as a process*. In many practical applications, the environment is so complex that it is not feasible to lay out a comprehensive theoretical model and use classical algorithmic theory and mathematical optimization. It is necessary, as well as beneficial, to take a robust approach, by applying an optimization method that learns as more aspects of the problem are observed. This view of optimization as a process has become prominent in various fields, which has led to spectacular successes in modeling and systems that are now part of our daily lives.

The growing body of literature of machine learning, statistics, decision science, and mathematical optimization blurs the classical distinctions between deterministic modeling, stochastic modeling, and optimization methodology. We continue this trend in this book, studying a prominent optimization framework whose precise location in the mathematical sciences is unclear: the framework of *online convex optimization* (OCO), which was first defined in the machine learning literature (see section 1.4, later in this chapter). The metric of success is borrowed from game theory, and the framework is closely tied to statistical learning theory and convex optimization.

We embrace these fruitful connections and, on purpose, do not try to use any particular jargon in the discussion. Rather, this book will start with actual problems that can be modeled and solved via OCO. We will proceed to present rigorous definitions, backgrounds, and algorithms. Throughout, we provide connections to the literature in other fields. It is our hope that you, the reader, will contribute to our understanding of these connections from your domain of expertise, and expand the growing amount of literature on this fascinating subject.

1.1 The Online Convex Optimization Setting

In OCO, an online player iteratively makes decisions. At the time of each decision, the outcome or outcomes associated with it are unknown to the player.

After committing to a decision, the decision maker suffers a loss: every possible decision incurs a (possibly different) loss. These losses are unknown to the decision maker beforehand. The losses can be adversarially chosen, and even depend on the action taken by the decision maker.

Already at this point, several restrictions are necessary in order for this framework to make any sense at all:

- The losses determined by an adversary should not be allowed to be unbounded [1]. Otherwise, the adversary could keep decreasing the scale of the loss at each step, and never allow the algorithm to recover from the loss of the first step. Thus, we assume that the losses lie in some bounded region.

- The decision set must be somehow bounded and/or structured, though not necessarily finite.

  To see why this is necessary, consider decision making with an infinite set of possible decisions. An adversary can assign high loss to all the strategies chosen by the player indefinitely, while setting apart some strategies with zero loss. This precludes any meaningful performance metric.

Surprisingly, interesting statements and algorithms can be derived with not much more than these two restrictions. The online convex optimization (OCO) framework models the decision set as a convex set in Euclidean space denoted as $\mathcal{K} \subseteq \mathbb{R}^n$. The costs are modeled as bounded convex functions over $\mathcal{K}$.

The OCO framework can be seen as a structured repeated game. The protocol of this learning framework is as follows.

At iteration t, the online player chooses $\mathbf{x}_t \in \mathcal{K}$. After the player has committed to this choice, a convex cost function $f_t \in \mathcal{F} : \mathcal{K} \mapsto \mathbb{R}$ is revealed. Here, $\mathcal{F}$ is the bounded family of cost functions available to the adversary. The cost incurred by the online player is $f_t(\mathbf{x}_t)$, the value of the cost function for the choice $\mathbf{x}_t$. Let T denote the total number of game iterations.

What would make an algorithm a good OCO algorithm? As the framework is game-theoretic and adversarial in nature, the appropriate performance metric also comes from game theory: define the *regret* of the decision

maker to be the difference between the total cost she has incurred and that of the best fixed decision in hindsight. In OCO, we are usually interested in an upper bound on the worst-case regret of an algorithm.

Let $\mathcal{A}$ be an algorithm for OCO, which maps a certain game history to a decision in the decision set:

$$\mathbf{x}_t^{\mathcal{A}} = \mathcal{A}(f_1, ..., f_{t-1}) \in \mathcal{K}.$$

We formally define the regret of $\mathcal{A}$ after T iterations as:

$$\text{Regret}_T(\mathcal{A}) = \sup_{\{f_1,...,f_T\} \subseteq \mathcal{F}} \left\{ \sum_{t=1}^{T} f_t(\mathbf{x}_t^{\mathcal{A}}) - \min_{\mathbf{x} \in \mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}) \right\}. \tag{1.1}$$

If the algorithm is clear from the context, we henceforth omit the superscript and denote the algorithm's decision at time t simply as $\mathbf{x}_t$. Intuitively, an algorithm performs well if its regret is sublinear as a function of T (i.e. $\text{Regret}_T(\mathcal{A}) = o(T)$), since this implies that on average, the algorithm performs as well as the best fixed strategy in hindsight.

The running time of an algorithm for OCO is defined to be the worst-case expected time to produce $\mathbf{x}_t$, for an iteration $t \in [T]^2$ in a T-iteration repeated game. Typically, the running time will depend on n (the dimensionality of the decision set $\mathcal{K}$), T (the total number of game iterations), and the parameters of the cost functions and underlying convex set.

1.2 Examples of Problems That Can Be Modeled via Online Convex Optimization

Perhaps the main reason that OCO has become a leading online learning framework in recent years is its powerful modeling capability: problems from diverse domains such as online routing, ad selection for search engines, and spam filtering can all be modeled as special cases. In this section, we briefly survey a few special cases and how they fit into the OCO framework.

1.2.1 Prediction from expert advice

Perhaps the most well known problem in prediction theory is the *experts problem*. The decision maker has to choose among the advice of n given experts. After making her choice, a loss between zero and 1 is incurred. This scenario is repeated iteratively, and at each iteration, the costs of the various experts are arbitrary (and possibly even adversarial, trying to mislead the

decision maker). The goal of the decision maker is to do as well as the best expert in hindsight.

The OCO setting captures this as a special case: the set of decisions is the set of all distributions over n elements (experts); that is, the n-dimensional simplex $\mathcal{K} = \Delta_n = \{\mathbf{x} \in \mathbb{R}^n \ , \ \sum_i \mathbf{x}_i = 1 \ , \ \mathbf{x}_i \geq 0\}$. Let the cost of the ith expert at iteration t be $\mathbf{g}_t(i)$, and let $\mathbf{g}_t$ be the cost vector of all n experts. Then the cost function is the expected cost of choosing an expert according to distribution $\mathbf{x}$, and it is given by the linear function $f_t(\mathbf{x}) = \mathbf{g}_t^\top \mathbf{x}$.

Thus, prediction from expert advice is a special case of OCO, in which the decision set is the simplex and the cost functions are linear and bounded, in the ℓ_∞ norm, to be at most 1. The bound on the cost functions is derived from the bound on the elements of the cost vector $\mathbf{g}_t$.

The fundamental importance of the experts problem in machine learning warrants special attention, and we shall return to it and analyze it in detail at the end of this chapter.

1.2.2 Online spam filtering

Consider an online spam-filtering system. Repeatedly, emails arrive in the system and are classified as spam or valid. Obviously, such a system has to cope with adversarially generated data and dynamically change with the varying input—a hallmark of the OCO model.

The linear variant of this model is captured by representing the emails as vectors according to the "bag-of-words" representation. Each email is represented as a vector $\mathbf{a} \in \mathbb{R}^d$, where d is the number of words in the dictionary. The entries of this vector are all zero, except for those coordinates that correspond to words appearing in the email, which are assigned the value one.

To predict whether an email is spam, we learn a filter, for example a vector $\mathbf{x} \in \mathbb{R}^d$. Usually a bound on the Euclidean norm of this vector is decided upon a priori, and is a parameter of great importance in practice.

Classification of an email $\mathbf{a} \in \mathbb{R}^d$ by a filter $\mathbf{x} \in \mathbb{R}^d$ is given by the sign of the inner product between these two vectors, i.e., $\hat{b} = \text{sign}(\mathbf{x}^\top \mathbf{a})$ (with, for example, $+1$ meaning valid and -1 meaning spam).

In the OCO model of online spam filtering, the decision set is taken to be the set of all such norm-bounded linear filters, i.e., the Euclidean ball of a certain radius. The cost functions are determined according to a stream of incoming emails arriving into the system, and their labels (which may be known by the system, partially known, or not known at all). Let $(\mathbf{a}, b)$ be an email/label pair. Then the corresponding cost function over filters is given

by $f(\mathbf{x}) = \ell(\hat{b}, b)$. Here $\hat{b}$ is the classification given by the filter $\mathbf{x}$, b is the true label, and ℓ is a convex loss function, for example, the scaled square loss $\ell(\hat{b}, b) = \frac{1}{4}(\hat{b} - b)^2$.

At this point the reader may wonder - why use a square loss rather than any other function? The most natural choice being perhaps a loss of one if $b = \hat{b}$ and zero otherwise.

To answer this, notice first that if both b and $\hat{b}$ are binary and in $\{-1, 1\}$, then the square loss is indeed one or zero. However, moving to a continuous function allows us much more flexibility in the decision making process. We can allow, for example, the algorithm to return a number in the interval $[-1, 1]$ depending on its confidence.

Another reason has to do with the algorithmic efficiency of finding a a good solution. This will be the subject of future chapters.

1.2.3 Online shortest paths

In the online shortest path problem, the decision maker is given a directed graph $G = (V, E)$ and a source-sink pair $u, v \in V$. At each iteration $t \in [T]$, the decision maker chooses a path $p_t \in \mathcal{P}_{u,v}$, where $\mathcal{P}_{u,v} \subseteq E^{|V|}$ is the set of all u-v-paths in the graph. The adversary independently chooses weights (lengths) on the edges of the graph, given by a function from the edges to the real numbers $\mathbf{w}_t : E \mapsto \mathbb{R}$, which can be represented as a vector $\mathbf{w}_t \in \mathbb{R}^m$, where $m = |E|$. The decision maker suffers and observes a loss, which is the weighted length of the chosen path $\sum_{e \in p_t} \mathbf{w}_t(e)$.

The discrete description of this problem as an experts problem, where we have an expert for each path, presents an efficiency challenge. There are potentially exponentially many paths in terms of the graph representation size.

Alternatively, the online shortest path problem can be cast in the online convex optimization framework as follows. Recall the standard description of the set of all distributions over paths (flows) in a graph as a convex set in $\mathbb{R}^m$, with $O(m + |V|)$ constraints (figure 1.1). Denote this flow polytope by $\mathcal{K}$. The expected cost of a given flow $\mathbf{x} \in \mathcal{K}$ (distribution over paths) is then a linear function, given by $f_t(\mathbf{x}) = \mathbf{w}_t^\top \mathbf{x}$, where, as defined above, $\mathbf{w}_t(e)$ is the length of the edge $e \in E$. This inherently succinct formulation leads to computationally efficient algorithms.

$$\sum_{e=(u,w),w\in V} \mathbf{x}_e = 1 = \sum_{e=(w,v),w\in V} \mathbf{x}_e \qquad \text{flow value is one}$$

$$\forall w \in V \setminus \{u,v\} \quad \sum_{e=(v,x)\in E} \mathbf{x}_e = \sum_{e=(x,v)\in E} \mathbf{x}_e \qquad \text{flow conservation}$$

$$\forall e \in E \;\; 0 \le \mathbf{x}_e \le 1 \qquad \text{capacity constraints}$$

Figure 1.1: Linear equalities and inequalities that define the flow polytope, which is the convex hull of all u-v paths

1.2.4 Portfolio selection

In this section we consider a portfolio selection model that does not make any statistical assumptions about the stock market (as opposed to the standard geometric Brownian motion model for stock prices), and is called the "universal portfolio selection" model.

At each iteration $t \in [T]$, the decision maker chooses a distribution of her wealth over n assets $\mathbf{x}_t \in \Delta_n$. The adversary independently chooses market returns for the assets, i.e., a vector $\mathbf{r}_t \in \mathbb{R}^n$ with strictly positive entries such that each coordinate $\mathbf{r}_t(i)$ is the price ratio for the i'th asset between the iterations t and $t+1$. The ratio between the wealth of the investor at iterations $t+1$ and t is $\mathbf{r}_t^\top \mathbf{x}_t$, and hence the gain in this setting is defined to be the logarithm of this change ratio in wealth $\log(\mathbf{r}_t^\top \mathbf{x}_t)$. Notice that since $\mathbf{x}_t$ is the distribution of the investor's wealth, even if $\mathbf{x}_{t+1} = \mathbf{x}_t$, the investor may still need to trade to adjust for price changes.

The goal of regret minimization, which in this case corresponds to minimizing the difference $\max_{\mathbf{x}^\star\in\Delta_n} \sum_{t=1}^T \log(\mathbf{r}_t^\top \mathbf{x}^\star) - \sum_{t=1}^T \log(\mathbf{r}_t^\top \mathbf{x}_t)$, has an intuitive interpretation. The first term is the logarithm of the wealth accumulated by the best possible in-hindsight distribution $\mathbf{x}^\star$. Since this distribution is fixed, it corresponds to a strategy of rebalancing the position after every trading period, and hence, is called a *constant rebalanced portfolio*. The second term is the logarithm of the wealth accumulated by the online decision maker. Hence regret minimization corresponds to maximizing the ratio of the investor's wealth to the wealth of the best benchmark from a pool of investing strategies.

A *universal* portfolio selection algorithm is defined to be one that, in this setting, attains regret converging to zero. Such an algorithm, albeit

requiring exponential time, was first described by Cover (see bibliographic notes at the end of this chapter). The online convex optimization framework has given rise to much more efficient algorithms based on Newton's method. We shall return to study these in detail in chapter 4.

1.2.5 Matrix completion and recommendation systems

The prevalence of large-scale media delivery systems such as the Netflix online video library, Spotify music service and many others, give rise to very large scale recommendation systems. One of the most popular and successful models for automated recommendation is the matrix completion model.

In this mathematical model, recommendations are thought of as composing a matrix. The customers are represented by the rows, the different media are the columns, and at the entry corresponding to a particular user/media pair we have a value scoring the preference of the user for that particular media.

For example, for the case of binary recommendations for music, we have a matrix $X \in \{0,1\}^{n\times m}$ where n is the number of persons considered, m is the number of songs in our library, and 0/1 signifies dislike/like respectively:

$$X_{ij} = \begin{cases} 0, & \text{person } i \text{ dislikes song } j \\ \\ 1, & \text{person } i \text{ likes song } j \end{cases} .$$

In the online setting, for each iteration the decision maker outputs a preference matrix $X_t \in \mathcal{K}$, where $\mathcal{K} \subseteq \{0,1\}^{n\times m}$ is a subset of all possible zero/one matrices. An adversary then chooses a user/song pair (i_t, j_t) along with a "real" preference for this pair $y_t \in \{0,1\}$. Thus, the loss experienced by the decision maker can be described by the convex loss function,

$$f_t(X) = (X_{i_t,j_t} - y_t)^2.$$

The natural comparator in this scenario is a low-rank matrix, which corresponds to the intuitive assumption that preference is determined by few unknown factors. Regret with respect to this comparator means performing, on the average, as few preference-prediction errors as the best low-rank matrix.

We return to this problem and explore efficient algorithms for it in chapter 7.

1.3 A Gentle Start: Learning from Expert Advice

Consider the following fundamental iterative decision making problem:

At each time step $t = 1, 2, \ldots, T$, the decision maker faces a choice between two actions A or B (i.e., buy or sell a certain stock). The decision maker has assistance in the form of N "experts" that offer their advice. After a choice between the two actions has been made, the decision maker receives feedback in the form of a loss associated with each decision. For simplicity one of the actions receives a loss of zero (i.e., the "correct" decision) and the other a loss of one.

We make the following elementary observations:

1. A decision maker that chooses an action uniformly at random each iteration, trivially attains a loss of $\frac{T}{2}$ and is "correct" 50% of the time.

2. In terms of the number of mistakes, no algorithm can do better in the worst case! In a later exercise, we will devise a randomized setting in which the expected number of mistakes of any algorithm is at least $\frac{T}{2}$.

We are thus motivated to consider a *relative performance metric*: can the decision maker make as few mistakes as the best expert in hindsight? The next theorem shows that the answer in the worst case is negative for a deterministic decision maker.

Theorem 1.1. *Let $L \leq \frac{T}{2}$ denote the number of mistakes made by the best expert in hindsight. Then there does not exist a deterministic algorithm that can guarantee less than $2L$ mistakes.*

Proof. Assume that there are only two experts and one always chooses option A while the other always chooses option B. Consider the setting in which an adversary always chooses the opposite of our prediction (she can do so, since our algorithm is deterministic). Then, the total number of mistakes the algorithm makes is T. However, the best expert makes no more than $\frac{T}{2}$ mistakes (at every iteration exactly one of the two experts is mistaken). Therefore, there is no algorithm that can always guarantee less than $2L$ mistakes.

□

This observation motivates the design of random decision making algorithms, and indeed, the OCO framework gracefully models decisions on a continuous probability space. Henceforth we prove Lemmas 1.3 and 1.4 that show the following:

Theorem 1.2. *Let $\varepsilon \in (0, \frac{1}{2})$. Suppose the best expert makes L mistakes. Then:*

1. *There is an efficient deterministic algorithm that can guarantee less than $2(1+\varepsilon)L + \frac{2\log N}{\varepsilon}$ mistakes;*
2. *There is an efficient randomized algorithm for which the expected number of mistakes is at most $(1+\varepsilon)L + \frac{\log N}{\varepsilon}$.*

1.3.1 The weighted majority algorithm

The weighted majority (WM) algorithm is intuitive to describe: each expert i is assigned a weight $W_t(i)$ at every iteration t. Initially, we set $W_1(i) = 1$ for all experts $i \in [N]$. For all $t \in [T]$ let $S_t(A), S_t(B) \subseteq [N]$ be the set of experts that choose A (and respectively B) at time t. Define,

$$W_t(A) = \sum_{i \in S_t(A)} W_t(i) \qquad W_t(B) = \sum_{i \in S_t(B)} W_t(i)$$

and predict according to

$$a_t = \begin{cases} A & \text{if } W_t(A) \geq W_t(B) \\ B & \text{otherwise.} \end{cases}$$

Next, update the weights $W_t(i)$ as follows:

$$W_{t+1}(i) = \begin{cases} W_t(i) & \text{if expert } i \text{ was correct} \\ W_t(i)(1-\varepsilon) & \text{if expert } i \text{ was wrong} \end{cases},$$

where ε is a parameter of the algorithm that will affect its performance. This concludes the description of the WM algorithm. We proceed to bound the number of mistakes it makes.

Lemma 1.3. *Denote by M_t the number of mistakes the algorithm makes until time t, and by $M_t(i)$ the number of mistakes made by expert i until time t. Then, for any expert $i \in [N]$ we have*

$$M_T \leq 2(1+\varepsilon)M_T(i) + \frac{2\log N}{\varepsilon}.$$

We can optimize ε to minimize the above bound. The expression on the right hand side is of the form $f(x) = ax + b/x$, that reaches its minimum

at $x = \sqrt{b/a}$. Therefore the bound is minimized at $\varepsilon^\star = \sqrt{\log N / M_T(i)}$. Using this optimal value of ε, we get that for the best expert $i^\star$

$$M_T \leq 2M_T(i^\star) + O\left(\sqrt{M_T(i^\star)\log N}\right).$$

Of course, this value of $\varepsilon^\star$ cannot be used in advance since we do not know which expert is the best one ahead of time (and therefore we do not know the value of $M_T(i^\star)$). However, we shall see later on that the same asymptotic bound can be obtained even without this prior knowledge.

Let us now prove Lemma 1.3.

Proof. Let $\Phi_t = \sum_{i=1}^N W_t(i)$ for all $t \in [T]$, and note that $\Phi_1 = N$.

Notice that $\Phi_{t+1} \leq \Phi_t$. However, on iterations in which the WM algorithm erred, we have

$$\Phi_{t+1} \leq \Phi_t(1 - \frac{\varepsilon}{2}),$$

the reason being that experts with at least half of total weight were wrong (else WM would not have erred), and therefore

$$\Phi_{t+1} \leq \frac{1}{2}\Phi_t(1-\varepsilon) + \frac{1}{2}\Phi_t = \Phi_t(1 - \frac{\varepsilon}{2}).$$

From both observations,

$$\Phi_t \leq \Phi_1(1 - \frac{\varepsilon}{2})^{M_t} = N(1 - \frac{\varepsilon}{2})^{M_t}.$$

On the other hand, by definition we have for any expert i that

$$W_T(i) = (1-\varepsilon)^{M_T(i)}.$$

Since the value of $W_T(i)$ is always less than the sum of all weights Φ_T, we conclude that

$$(1-\varepsilon)^{M_T(i)} = W_T(i) \leq \Phi_T \leq N(1 - \frac{\varepsilon}{2})^{M_T}.$$

Taking the logarithm of both sides we get

$$M_T(i)\log(1-\varepsilon) \leq \log N + M_T \log(1 - \frac{\varepsilon}{2}).$$

Next, we use the approximations

$$-x - x^2 \leq \log(1-x) \leq -x \qquad 0 < x < \frac{1}{2},$$

which follow from the Taylor series of the logarithm function, to obtain that

$$-M_T(i)(\varepsilon + \varepsilon^2) \leq \log N - M_T\frac{\varepsilon}{2},$$

and the lemma follows. □

1.3.2 Randomized weighted majority

In the randomized version of the WM algorithm, denoted RWM, we choose expert i w.p. $p_t(i) = W_t(i)/\sum_{j=1}^{N} W_t(j)$ at time t.

Lemma 1.4. *Let M_t denote the number of mistakes made by RWM until iteration t. Then, for any expert $i \in [N]$ we have*

$$\mathbf{E}[M_T] \leq (1+\varepsilon)M_T(i) + \frac{\log N}{\varepsilon}.$$

The proof of this lemma is very similar to the previous one, where the factor of two is saved by the use of randomness:

Proof. As before, let $\Phi_t = \sum_{i=1}^{N} W_t(i)$ for all $t \in [T]$, and note that $\Phi_1 = N$. Let $\tilde{m}_t = M_t - M_{t-1}$ be the indicator variable that equals one if the RWM algorithm makes a mistake on iteration t. Let $m_t(i)$ equal one if the i'th expert makes a mistake on iteration t and zero otherwise. Inspecting the sum of the weights:

$$\begin{aligned}
\Phi_{t+1} &= \sum_i W_t(i)(1-\varepsilon m_t(i)) \\
&= \Phi_t(1-\varepsilon \sum_i p_t(i)m_t(i)) && p_t(i) = \tfrac{W_t(i)}{\sum_j W_t(j)} \\
&= \Phi_t(1-\varepsilon\,\mathbf{E}[\tilde{m}_t]) \\
&\leq \Phi_t e^{-\varepsilon\,\mathbf{E}[\tilde{m}_t]}. && 1+x \leq e^x
\end{aligned}$$

On the other hand, by definition we have for any expert i that

$$W_T(i) = (1-\varepsilon)^{M_T(i)}$$

Since the value of $W_T(i)$ is always less than the sum of all weights Φ_T, we conclude that

$$(1-\varepsilon)^{M_T(i)} = W_T(i) \leq \Phi_T \leq N e^{-\varepsilon\,\mathbf{E}[M_T]}.$$

Taking the logarithm of both sides we get

$$M_T(i)\log(1-\varepsilon) \leq \log N - \varepsilon\,\mathbf{E}[M_T]$$

Next, we use the approximation

$$-x - x^2 \leq \log(1-x) \leq -x \qquad , \quad 0 < x < \frac{1}{2}$$

to obtain

$$-M_T(i)(\varepsilon+\varepsilon^2) \leq \log N - \varepsilon\,\mathbf{E}[M_T],$$

and the lemma follows. □

1.3.3 Hedge

The RWM algorithm is in fact more general: instead of considering a discrete number of mistakes, we can consider measuring the performance of an expert by a non-negative real number $\ell_t(i)$, which we refer to as the *loss* of the expert i at iteration t. The randomized weighted majority algorithm guarantees that a decision maker following its advice will incur an average expected loss approaching that of the best expert in hindsight.

Historically, this was observed by a different and closely related algorithm called Hedge, whose total loss bound will be of interest to us later on in the book.

Algorithm 1 Hedge

Initialize: $\forall i \in [N],\ W_1(i) = 1$
for $t = 1$ to T **do**
    Pick $i_t \sim_R W_t$, i.e., $i_t = i$ with probability $\mathbf{x}_t(i) = \frac{W_t(i)}{\sum_j W_t(j)}$
    Incur loss $\ell_t(i_t)$.
    Update weights $W_{t+1}(i) = W_t(i)e^{-\varepsilon \ell_t(i)}$
end for

Henceforth, denote in vector notation the expected loss of the algorithm by

$$\mathbf{E}[\ell_t(i_t)] = \sum_{i=1}^{N} \mathbf{x}_t(i)\ell_t(i) = \mathbf{x}_t^\top \ell_t$$

Theorem 1.5. *Let ℓ_t^2 denote the N-dimensional vector of square losses, i.e., $\ell_t^2(i) = \ell_t(i)^2$, let $\varepsilon > 0$, and assume all losses to be non-negative. The Hedge algorithm satisfies for any expert $i^\star \in [N]$:*

$$\sum_{t=1}^{T} \mathbf{x}_t^\top \ell_t \leq \sum_{t=1}^{T} \ell_t(i^\star) + \varepsilon \sum_{t=1}^{T} \mathbf{x}_t^\top \ell_t^2 + \frac{\log N}{\varepsilon}$$

Proof. As before, let $\Phi_t = \sum_{i=1}^{N} W_t(i)$ for all $t \in [T]$, and note that $\Phi_1 = N$.

Inspecting the sum of weights:

$$\begin{aligned}
\Phi_{t+1} &= \textstyle\sum_i W_t(i) e^{-\varepsilon \ell_t(i)} \\
&= \Phi_t \textstyle\sum_i \mathbf{x}_t(i) e^{-\varepsilon \ell_t(i)} && \mathbf{x}_t(i) = \frac{W_t(i)}{\sum_j W_t(j)} \\
&\leq \Phi_t \textstyle\sum_i \mathbf{x}_t(i)(1 - \varepsilon \ell_t(i) + \varepsilon^2 \ell_t(i)^2)) && \text{for } x \geq 0, \\
& && e^{-x} \leq 1 - x + x^2 \\
&= \Phi_t (1 - \varepsilon \mathbf{x}_t^\top \ell_t + \varepsilon^2 \mathbf{x}_t^\top \ell_t^2) \\
&\leq \Phi_t e^{-\varepsilon \mathbf{x}_t^\top \ell_t + \varepsilon^2 \mathbf{x}_t^\top \ell_t^2}. && 1 + x \leq e^x
\end{aligned}$$

On the other hand, by definition, for expert $i^\star$ we have that

$$W_{T+1}(i^\star) = e^{-\varepsilon \sum_{t=1}^T \ell_t(i^\star)}$$

Since the value of $W_T(i^\star)$ is always less than the sum of all weights Φ_t, we conclude that

$$W_{T+1}(i^\star) \leq \Phi_{T+1} \leq N e^{-\varepsilon \sum_t \mathbf{x}_t^\top \ell_t + \varepsilon^2 \sum_t \mathbf{x}_t^\top \ell_t^2}.$$

Taking the logarithm of both sides we get

$$-\varepsilon \sum_{t=1}^T \ell_t(i^\star) \leq \log N - \varepsilon \sum_{t=1}^T \mathbf{x}_t^\top \ell_t + \varepsilon^2 \sum_{t=1}^T \mathbf{x}_t^\top \ell_t^2$$

and the theorem follows by simplifying. □

1.4 Bibliographic Remarks

The OCO model was first defined by Zinkevich [2003] and has since become widely influential in the learning community and significantly extended since (see thesis and surveys [Hazan, 2006, 2011, Shalev-Shwartz, 2011]).

The problem of prediction from expert advice and the Weighted Majority algorithm were devised in [Littlestone and Warmuth, 1989, 1994]. This seminal work was one of the first uses of the multiplicative updates method—a ubiquitous meta-algorithm in computation and learning, see the survey [Arora et al., 2012] for more details. The Hedge algorithm was introduced by Freund and Schapire [1997].

The Universal Portfolios model was put forth in [Cover, 1991], and is one of the first examples of a worst-case online learning model. Cover gave an optimal-regret algorithm for universal portfolio selection that runs in exponential time. A polynomial time algorithm was given in [Kalai and Vempala, 2003], which was further sped up in [Agarwal et al., 2006, Hazan et al., 2007]. Numerous extensions to the model also appeared in the literature, including addition of transaction costs [Blum and Kalai, 1999] and relation to the Geometric Brownian Motion model for stock prices [Hazan and Kale, 2009].

In their influential paper, Awerbuch and Kleinberg [2008] put forth the application of online convex optimization to online routing. A great deal of work has been devoted since then to improve the initial bounds, and generalize it into a complete framework for decision making with limited feedback. This framework is an extension of OCO, called Bandit Convex Optimization (BCO). We defer further bibliographic remarks to chapter 6 which is devoted to the BCO framework.

1.5 Exercises

1. (Attributed to Claude Shannon)
Construct market returns over two stocks for which the wealth accumulated over any single stock decreases exponentially, whereas the best constant rebalanced portfolio increases wealth exponentially. More precisely, construct two sequences of numbers in the range $(0, \infty)$, that represent returns, such that:

(a) Investing in any of the individual stocks results in exponential decrease in wealth. This means that the product of the prefix of numbers in each of these sequences decreases exponentially.

(b) Investing evenly on the two assets and rebalancing after every iteration increases wealth exponentially.

2.

(a) Consider the experts problem in which the losses are between zero and a positive real number $G > 0$. Give an algorithm that attains expected loss upper bounded by:

$$\sum_{t=1}^{T} \mathbf{E}[\ell_t(i_t)] \leq \max_{i^\star \in [N]} \sum_{t=1}^{T} \ell_t(i^\star) + c\sqrt{T \log N}$$

for the best constant c you can (the constant c should be independent of the number of game iterations T, and the number of experts N. Assume that T is known in advance).

(b) Suppose the upper bound G is not known in advance. Give an algorithm whose performance is asymptotically as good as your algorithm in part (a), up to an additive and/or multiplicative constant which is independent of T, N, G. Prove your claim.

3. Consider the experts problem in which the losses can be negative and are real numbers in the range $[-1, 1]$. Give an algorithm with regret guarantee of $O(\sqrt{T \log N})$ and prove your claim.

Chapter 2

Basic Concepts in Convex Optimization

In this chapter we give a gentle introduction to convex optimization and present some basic algorithms for solving convex mathematical programs. Although offline convex optimization is not our main topic, it is useful to recall the basic definitions and results before we move on to OCO. This will help in assessing the advantages and limitations of OCO. Furthermore, we describe some tools that will be our bread-and-butter later on.

The material in this chapter is far from being new. A broad and significantly more detailed literature exists, and the reader is deferred to the bibliography at the end of this chapter for references. We give here only the most elementary analysis, and focus on the techniques that will be of use to us later on.

2.1 Basic Definitions and Setup

The goal in this chapter is to minimize a continuous and convex function over a convex subset of Euclidean space. Henceforth, let $\mathcal{K} \subseteq \mathbb{R}^d$ be a bounded convex and closed set in Euclidean space. We denote by D an upper bound on the diameter of $\mathcal{K}$:

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{K},\ \|\mathbf{x} - \mathbf{y}\| \leq D.$$

A set $\mathcal{K}$ is convex if for any $\mathbf{x}, \mathbf{y} \in \mathcal{K}$, all the points on the line segment connecting $\mathbf{x}$ and $\mathbf{y}$ also belong to $\mathcal{K}$, i.e.,

$$\forall \alpha \in [0,1],\ \alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in \mathcal{K}.$$

A function $f : \mathcal{K} \mapsto \mathbb{R}$ is convex if for any $\mathbf{x}, \mathbf{y} \in \mathcal{K}$

$$\forall \alpha \in [0,1],\ f((1-\alpha)\mathbf{x} + \alpha\mathbf{y}) \leq (1-\alpha)f(\mathbf{x}) + \alpha f(\mathbf{y}).$$

This inequality, and generalizations thereof, is also known as Jensen's inequality. Equivalently, if f is differentiable, that is, its gradient $\nabla f(\mathbf{x})$ exists for all $\mathbf{x} \in \mathcal{K}$, then it is convex if and only if $\forall \mathbf{x}, \mathbf{y} \in \mathcal{K}$

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}).$$

For convex and non-differentiable functions f, the subgradient at $\mathbf{x}$ is *defined* to be any member of the set of vectors $\{\nabla f(\mathbf{x})\}$ that satisfies the above for all $\mathbf{y} \in \mathcal{K}$.

We denote by $G > 0$ an upper bound on the norm of the subgradients of f over $\mathcal{K}$, i.e., $\|\nabla f(\mathbf{x})\| \leq G$ for all $\mathbf{x} \in \mathcal{K}$. Such an upper bound implies that the function f is Lipschitz continuous with parameter G, that is, for all $\mathbf{x}, \mathbf{y} \in \mathcal{K}$

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq G\|\mathbf{x} - \mathbf{y}\|.$$

The optimization and machine learning literature studies special types of convex functions that admit useful properties, which in turn allow for more efficient optimization. Notably, we say that a function is α-strongly convex if

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\alpha}{2}\|\mathbf{y} - \mathbf{x}\|^2.$$

A function is β-smooth if

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\beta}{2}\|\mathbf{y} - \mathbf{x}\|^2.$$

The latter condition is equivalent to a Lipschitz condition over the gradients, i.e.,

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq \beta\|\mathbf{x} - \mathbf{y}\|.$$

If the function is twice differentiable and admits a second derivative, known as a Hessian for a function of several variables, the above conditions are equivalent to the following condition on the Hessian, denoted $\nabla^2 f(\mathbf{x})$:

$$\alpha I \preccurlyeq \nabla^2 f(\mathbf{x}) \preccurlyeq \beta I,$$

where $A \preccurlyeq B$ if the matrix $B - A$ is positive semidefinite.

When the function f is both α-strongly convex and β-smooth, we say that it is γ-well-conditioned where γ is the ratio between strong convexity and smoothness, also called the *condition number* of f

$$\gamma = \frac{\alpha}{\beta} \leq 1$$

2.1.1 Projections onto convex sets

In the following algorithms we shall make use of a projection operation onto a convex set, which is defined as the closest point in terms of Euclidean distance [3] inside the convex set to a given point. Formally,

$$\prod_{\mathcal{K}}(\mathbf{y}) \stackrel{\text{def}}{=} \arg\min_{\mathbf{x}\in\mathcal{K}} \|\mathbf{x}-\mathbf{y}\|.$$

When clear from the context, we shall remove the $\mathcal{K}$ subscript. It is left as an exercise to the reader to prove that the projection of a given point over a closed, bounded and convex set exists and is unique.

The computational complexity of projections is a subtle issue that depends much on the characterization of $\mathcal{K}$ itself. Most generally, $\mathcal{K}$ can be represented by a membership oracle—an efficient procedure that is capable of deciding whether a given $\mathbf{x}$ belongs to $\mathcal{K}$ or not. In this case, projections can be computed in polynomial time. In certain special cases, projections can be computed very efficiently in near-linear time. The computational cost of projections, as well as optimization algorithms that avoid them altogether, is discussed in chapter 7.

A crucial property of projections that we shall make extensive use of is the Pythagorean theorem, which we state here for completeness:

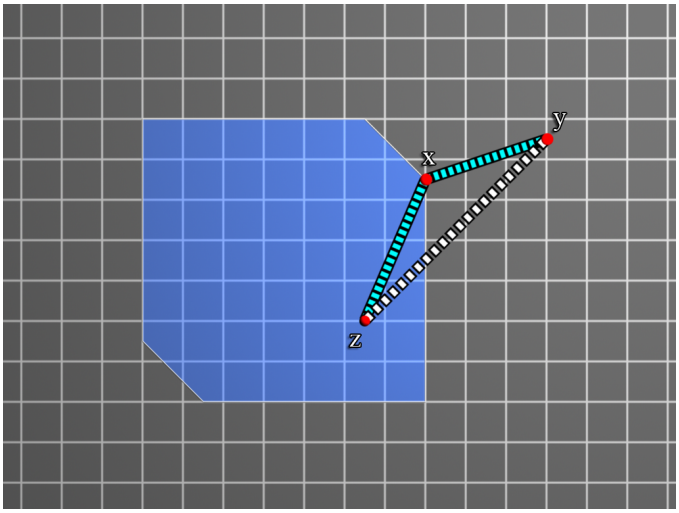

Figure 2.1: Pythagorean theorem

Theorem 2.1 (Pythagoras, circa 500 BC)**.** *Let $\mathcal{K} \subseteq \mathbb{R}^d$ be a convex set, $\mathbf{y} \in \mathbb{R}^d$ and $\mathbf{x} = \Pi_{\mathcal{K}}(\mathbf{y})$. Then for any $\mathbf{z} \in \mathcal{K}$ we have*

$$\|\mathbf{y} - \mathbf{z}\| \geq \|\mathbf{x} - \mathbf{z}\|.$$

We note that there exists a more general version of the Pythagorean theorem. The above theorem and the definition of projections are true and valid not only for Euclidean norms, but for projections according to other distances that are not norms. In particular, an analogue of the Pythagorean theorem remains valid with respect to Bregman divergences (see chapter 5).

2.1.2 Introduction to optimality conditions

The standard curriculum of high school mathematics contains the basic facts concerning when a function (usually in one dimension) attains a local optimum or saddle point. The generalization of these conditions to more than one dimension is called the KKT (Karush-Kuhn-Tucker) conditions, and the reader is referred to the bibliographic material at the end of this chapter for an in-depth rigorous discussion of optimality conditions in general mathematical programming.

For our purposes, we describe only briefly and intuitively the main facts that we will require henceforth. Naturally, we restrict ourselves to convex programming, and thus a local minimum of a convex function is also a global minimum (see exercises at the end of this chapter). In general there can be many points in which a function is minimized, and thus we refer to the *set* of minima of a given objective function, denoted as $\arg\min_{\mathbf{x}\in\mathbb{R}^n}\{f(\mathbf{x})\}$ [4].

The generalization of the fact that a minimum of a convex differentiable function on $\mathbb{R}$ is a point in which its derivative is equal to zero, is given by the multi-dimensional analogue that its gradient is zero:

$$\nabla f(\mathbf{x}) = 0 \iff \mathbf{x} \in \arg\min_{\mathbf{x}\in\mathbb{R}^n}\{f(\mathbf{x})\}.$$

We will require a slightly more general, but equally intuitive, fact for constrained optimization: at a minimum point of a constrained convex function, the inner product between the negative gradient and direction towards the interior of $\mathcal{K}$ is non-positive. This is depicted in figure 2.2, which shows that $-\nabla f(\mathbf{x}^\star)$ defines a supporting hyperplane to $\mathcal{K}$. The intuition is that if the inner product were positive, one could improve the objective by moving in the direction of the projected negative gradient. This fact is stated formally in the following theorem.

Theorem 2.2 (Karush-Kuhn-Tucker)**.** *Let $\mathcal{K} \subseteq \mathbb{R}^d$ be a convex set, $\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{K}} f(\mathbf{x})$. Then for any $\mathbf{y} \in \mathcal{K}$ we have*

$$\nabla f(\mathbf{x}^\star)^\top (\mathbf{y} - \mathbf{x}^\star) \geq 0.$$

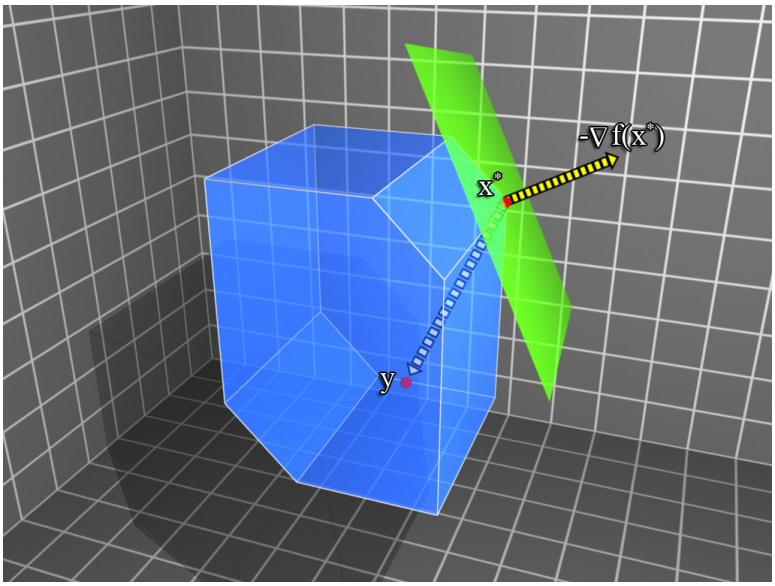

Figure 2.2: Optimality conditions: negative subgradient pointing outwards

2.2 Gradient Descent

Gradient descent (GD) is the simplest and oldest of optimization methods. It is an *iterative method*—the optimization procedure proceeds in iterations, each improving the objective value. The basic method amounts to iteratively moving the current point in the direction of the gradient, which is a linear time operation if the gradient is given explicitly (indeed, for many functions computing the gradient at a certain point is a simple linear-time operation).

The basic template algorithm, for unconstrained optimization, is given in 2, and a depiction of the iterates it produced in figure 2.3.

For a convex function there always exists a choice of step sizes that will cause GD to converge to the optimal solution. The rates of convergence, however, differ greatly and depend on the smoothness and strong convexity properties of the objective function. The following table summarises the

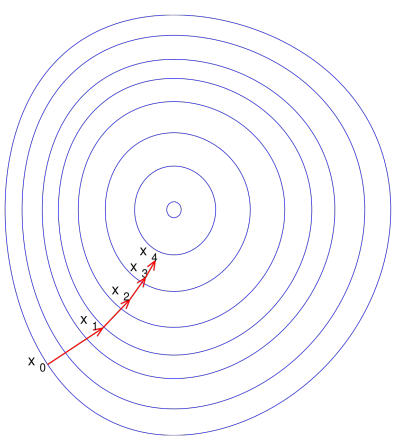

Figure 2.3: Iterates of the GD algorithm

Algorithm 2 Gradient Descent

1: Input: time horizon T, initial point x_0, step sizes $\{\eta_t\}$
2: **for** $t = 0, \dots, T-1$ **do**
3: $\quad \mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_t$
4: **end for**
5: **return** $\bar{\mathbf{x}} = \arg\min_{\mathbf{x}_t} \{f(\mathbf{x}_t)\}$

convergence rates of GD variants for convex functions with different convexity parameters. The rates described omit the (usually small) constants in the bounds—we focus on asymptotic rates.

In this section we address only the first row of Table 2.1. For accelerated methods and their analysis see references at the bibliographic section.

2.2.1 The Polyak stepsize

Luckily, there exists a simple choice of step sizes that yields the optimal convergence rate, called the Polyak stepsize. It has a huge advantage of not depending on the strong convexity and/or smoothness parameters of the objective function.

However, it does depend on the distance in function value to optimality and gradient norm. While the latter can be efficiently estimated, the distance to optimality is not always available if $f(\mathbf{x}^*)$ is not known ahead of

	general	α-strongly convex	β-smooth	γ-well conditioned
Gradient descent	$\frac{1}{\sqrt{T}}$	$\frac{1}{\alpha T}$	$\frac{\beta}{T}$	$e^{-\gamma T}$
Accelerated GD	—	—	$\frac{\beta}{T^2}$	$e^{-\sqrt{\gamma}\ T}$

Table 2.1: Rates of convergence of first order (gradient-based) methods as a function of the number of iterations and the smoothness and strong-convexity of the objective. Dependence on other parameters and constants, namely the Lipchitz constant, diameter of constraint set and initial distance to the objective is omitted. Acceleration for non-smooth functions is not possible in general.

time. This can be remedied, as referred to in the bibliography.

We henceforth denote:

1. Distance to optimality in value: $h_t = h(\mathbf{x}_t) = f(\mathbf{x}_t) - f(\mathbf{x}^*)$
2. Euclidean distance to optimality: $d_t = \|\mathbf{x}_t - \mathbf{x}^*\|$
3. Current gradient norm $\|\nabla_t\| = \|\nabla f(\mathbf{x}_t)\|$

With these notations we can describe the algorithm precisely in Algorithm 3:

Algorithm 3 Gradient Descent with Polyak stepsize

Input: time horizon T, x_0
for $t = 0, \ldots, T-1$ **do**
  Set $\eta_t = \frac{h_t}{\|\nabla_t\|^2}$
  $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_t$
end for
Return $\bar{\mathbf{x}} = \arg\min_{\mathbf{x}_t}\{f(\mathbf{x}_t)\}$

To prove precise convergence bounds, assume $\|\nabla_t\| \leq G$, and define:

$$B_T \quad = \quad \min\left\{\frac{Gd_0}{\sqrt{T}}, \frac{2\beta d_0^2}{T}, \frac{3G^2}{\alpha T}, \beta d_0^2\left(1 - \frac{\gamma}{4}\right)^T\right\}$$

We can now state the main guarantee of GD with the Polyak stepsize:

Theorem 2.3. *(GD with the Polyak Step Size) Algorithm 3 guarantees the following after T steps:*

$$f(\bar{\mathbf{x}}) - f(\mathbf{x}^\star) \leq \min_{0 \leq t \leq T}\{h_t\} \leq B_T$$

2.2.2 Measuring distance to optimality

When analyzing convergence of gradient methods, it is useful to use potential functions in lieu of function distance to optimality, such as gradient norm and/or Euclidean distance. The following relationships hold between these quantities.

Lemma 2.4. *The following properties hold for α-strongly-convex functions and/or β-smooth functions over Euclidean space $\mathbb{R}^d$.*

1. $\frac{\alpha}{2}d_t^2 \leq h_t$
2. $h_t \leq \frac{\beta}{2}d_t^2$
3. $\frac{1}{2\beta}\|\nabla_t\|^2 \leq h_t$
4. $h_t \leq \frac{1}{2\alpha}\|\nabla_t\|^2$

Proof. 1. $h_t \geq \frac{\alpha}{2}d_t^2$:

By strong convexity, we have

$$\begin{aligned} h_t &= f(\mathbf{x}_t) - f(\mathbf{x}^\star) \\ &\geq \nabla f(\mathbf{x}^\star)^\top(\mathbf{x}_t - \mathbf{x}^\star) + \tfrac{\alpha}{2}\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \\ &= \tfrac{\alpha}{2}\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \end{aligned}$$

where the last inequality follows since the gradient at the global optimum is zero.

2. $h_t \leq \frac{\beta}{2}d_t^2$:

By smoothness,

$$\begin{aligned} h_t &= f(\mathbf{x}_t) - f(\mathbf{x}^\star) \\ &\leq \nabla f(\mathbf{x}^\star)^\top(\mathbf{x}_t - \mathbf{x}^\star) + \tfrac{\beta}{2}\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \\ &= \tfrac{\beta}{2}\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \end{aligned}$$

where the last inequality follows since the gradient at the global optimum is zero.

3. $h_t \geq \frac{1}{2\beta}\|\nabla_t\|^2$: Using smoothness, and let $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\nabla_t$ for $\eta = \frac{1}{\beta}$,

$$\begin{aligned} h_t = f(\mathbf{x}_t) - f(\mathbf{x}^\star) &\geq f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \\ &\geq \nabla f(\mathbf{x}_t)^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) - \tfrac{\beta}{2}\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \\ &= \eta\|\nabla_t\|^2 - \tfrac{\beta}{2}\eta^2\|\nabla_t\|^2 \\ &= \tfrac{1}{2\beta}\|\nabla_t\|^2. \end{aligned}$$

4. $h_t \leq \frac{1}{2\alpha}\|\nabla_t\|^2$:

   We have for any pair $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$:

$$\begin{aligned} f(\mathbf{y}) &\geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\alpha}{2}\|\mathbf{x} - \mathbf{y}\|^2 \\ &\geq \min_{\mathbf{z}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{z} - \mathbf{x}) + \frac{\alpha}{2}\|\mathbf{x} - \mathbf{z}\|^2 \right\} \\ &= f(\mathbf{x}) - \frac{1}{2\alpha}\|\nabla f(\mathbf{x})\|^2. \\ &\text{by taking } \mathbf{z} = \mathbf{x} - \frac{1}{\alpha}\nabla f(\mathbf{x}) \end{aligned}$$

   In particular, taking $\mathbf{x} = \mathbf{x}_t$, $\mathbf{y} = \mathbf{x}^\star$, we get

$$h_t = f(\mathbf{x}_t) - f(\mathbf{x}^\star) \leq \frac{1}{2\alpha}\|\nabla_t\|^2. \tag{2.1}$$

□

2.2.3 Analysis of the Polyak stepsize

We are now ready to prove Theorem 2.3, which directly follows from the following lemma.

Lemma 2.5. *Suppose that a sequence $\mathbf{x}_0, \ldots \mathbf{x}_t$ satisfies:*

$$d_{t+1}^2 \leq d_t^2 - \frac{h_t^2}{\|\nabla_t\|^2} \tag{2.2}$$

then for $\bar{\mathbf{x}}$ as defined in the algorithm, we have:

$$f(\bar{\mathbf{x}}) - f(\mathbf{x}^\star) \leq \frac{1}{T}\sum_t h_t \leq B_T\,.$$

Proof. The proof analyzes different cases:

1. For convex functions with gradient bounded by G,

$$d_{t+1}^2 - d_t^2 \;\; \leq -\tfrac{h_t^2}{\|\nabla_t\|^2} \leq -\tfrac{h_t^2}{G^2}$$

   Summing up over T iterations, and using Cauchy-Schwartz on the T-dimensional vectors of $\frac{1}{T}\mathbf{1}$ and $(h_1, ..., h_T)$, we have

$$\begin{aligned} \frac{1}{T}\sum_t h_t &\leq \frac{1}{\sqrt{T}}\sqrt{\sum_t h_t^2} \\ &\leq \frac{G}{\sqrt{T}}\sqrt{\sum_t (d_t^2 - d_{t+1}^2)} \leq \frac{G d_0}{\sqrt{T}}\,. \end{aligned}$$

2. For smooth functions whose gradient is bounded by G, Lemma 2.4 implies:
$$d_{t+1}^2 - d_t^2 \leq -\frac{h_t^2}{\|\nabla_t\|^2} \leq -\frac{h_t}{2\beta}\,.$$
This implies
$$\frac{1}{T}\sum_t h_t \leq \frac{2\beta d_0^2}{T}\,.$$

3. For strongly convex functions, Lemma 2.4 implies:
$$d_{t+1}^2 - d_t^2 \leq -\frac{h_t^2}{\|\nabla_t\|^2} \leq -\frac{h_t^2}{G^2} \leq -\frac{\alpha^2 d_t^4}{4G^2}\,.$$
In other words, $d_{t+1}^2 \leq d_t^2(1 - \frac{\alpha^2 d_t^2}{4G^2})$. Defining $a_t := \frac{\alpha^2 d_t^2}{4G^2}$, we have:
$$a_{t+1} \leq a_t(1 - a_t)\,.$$
This implies that $a_t \leq \frac{1}{t+1}$, which can be seen by induction[5]. The proof is completed as follows[6] :
$$\begin{aligned}\frac{1}{T/2}\sum_{t=T/2}^{T} h_t^2 &\leq \frac{2G^2}{T}\sum_{t=T/2}^{T}(d_t^2 - d_{t+1}^2)\\ &= \frac{2G^2}{T}(d_{T/2}^2 - d_T^2)\\ &= \frac{8G^4}{\alpha^2 T}(a_{T/2} - a_T)\\ &\leq \frac{9G^4}{\alpha^2 T^2}\,.\end{aligned}$$
Thus, there exists a t for which $h_t^2 \leq \frac{9G^4}{\alpha^2 T^2}$. Taking the square root completes the claim.

4. For both strongly convex and smooth functions:
$$d_{t+1}^2 - d_t^2 \leq -\frac{h_t^2}{\|\nabla_t\|^2} \leq -\frac{h_t}{2\beta} \leq -\frac{\alpha}{4\beta}d_t^2$$
Thus,
$$h_T \leq \beta d_T^2 \leq \beta d_0^2\left(1 - \frac{\alpha}{4\beta}\right)^T = \beta d_0^2\left(1 - \frac{\gamma}{4}\right)^T\,.$$

This completes the proof of all cases. □

2.3 Constrained Gradient/Subgradient Descent

The vast majority of the problems considered in this text include constraints. Consider the examples given in section 1.2: a path is a point in the flow polytope, a portfolio is a point in the simplex and so on. In the language of optimization, we require $\mathbf{x}$ not only to minimize a certain objective function, but also to belong to a convex set $\mathcal{K}$.

In this section we describe and analyze constrained gradient descent. Algorithmically, the change from the previous section is small: after updating the current point in the direction of the gradient, one may need to project back to the decision set. However, the analysis is somewhat more involved, and instructive for the later parts of this text.

2.3.1 Basic gradient descent—linear convergence

Algorithmic box 4 describes a template for gradient descent over a constrained set. It is a template since the sequence of step sizes $\{\eta_t\}$ is left as an input parameter, and the several variants of the algorithm differ on its choice.

Algorithm 4 Basic gradient descent

1: Input: f, T, initial point $\mathbf{x}_1 \in \mathcal{K}$, sequence of step sizes $\{\eta_t\}$
2: **for** $t = 1$ to T **do**
3: Let $\mathbf{y}_{t+1} = \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t),\ \mathbf{x}_{t+1} = \Pi_{\mathcal{K}}(\mathbf{y}_{t+1})$
4: **end for**
5: **return** $\mathbf{x}_{T+1}$

As opposed to the unconstrained setting, here we require a precise setting of the learning rate to obtain the optimal convergence rate.

Theorem 2.6. *For constrained minimization of γ-well-conditioned functions and $\eta_t = \frac{1}{\beta}$, Algorithm 4 converges as*

$$h_{t+1} \leq h_1 \cdot e^{-\frac{\gamma t}{4}}$$

Proof. By strong convexity we have for every $\mathbf{x}, \mathbf{x}_t \in \mathcal{K}$ (where we denote $\nabla_t = \nabla f(\mathbf{x}_t)$ as before):

$$\nabla_t^\top (\mathbf{x} - \mathbf{x}_t) \leq f(\mathbf{x}) - f(\mathbf{x}_t) - \frac{\alpha}{2}\|\mathbf{x} - \mathbf{x}_t\|^2. \tag{2.3}$$

Next, appealing to the algorithm's definition and the choice $\eta_t = \frac{1}{\beta}$, we have

$$\mathbf{x}_{t+1} \quad = \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \nabla_t^\top (\mathbf{x} - \mathbf{x}_t) + \tfrac{\beta}{2}\|\mathbf{x} - \mathbf{x}_t\|^2 \right\}. \tag{2.4}$$

To see this, notice that

$$\begin{aligned}
&\prod_{\mathcal{K}}(\mathbf{x}_t - \eta_t \nabla_t) \\
&= \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \|\mathbf{x} - (\mathbf{x}_t - \eta_t\nabla_t)\|^2 \right\} && \text{definition of projection} \\
&= \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \nabla_t^\top (\mathbf{x} - \mathbf{x}_t) + \frac{1}{2\eta_t}\|\mathbf{x} - \mathbf{x}_t\|^2 \right\}. && \text{see exercises}
\end{aligned}$$

Thus, we have

$$\begin{aligned}
h_{t+1} - h_t &= f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \\
&\le \nabla_t^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{\beta}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 && \text{smoothness} \\
&\le \min_{\mathbf{x}\in\mathcal{K}} \left\{ \nabla_t^\top (\mathbf{x} - \mathbf{x}_t) + \frac{\beta}{2}\|\mathbf{x} - \mathbf{x}_t\|^2 \right\} && (2.4) \\
&\le \min_{\mathbf{x}\in\mathcal{K}} \left\{ f(\mathbf{x}) - f(\mathbf{x}_t) + \frac{\beta - \alpha}{2}\|\mathbf{x} - \mathbf{x}_t\|^2 \right\}. && (2.3)
\end{aligned}$$

The minimum can only grow if we take it over a subset of $\mathcal{K}$. Thus we can restrict our attention to all points that are convex combination of $\mathbf{x}_t$ and $\mathbf{x}^\star$, which we denote by the interval $[\mathbf{x}_t, \mathbf{x}^\star] = \{(1-\mu)\mathbf{x}_t + \mu\mathbf{x}^\star, \mu \in [0,1]\}$, and write

$$\begin{aligned}
h_{t+1} - h_t &\le \min_{\mathbf{x}\in[\mathbf{x}_t,\mathbf{x}^\star]} \left\{ f(\mathbf{x}) - f(\mathbf{x}_t) + \frac{\beta - \alpha}{2}\|\mathbf{x} - \mathbf{x}_t\|^2 \right\} \\
&= f((1-\mu)\mathbf{x}_t + \mu\mathbf{x}^\star) - f(\mathbf{x}_t) + \frac{\beta - \alpha}{2}\mu^2\|\mathbf{x}^\star - \mathbf{x}_t\|^2 \\
&\le (1-\mu)f(\mathbf{x}_t) + \mu f(\mathbf{x}^\star) - f(\mathbf{x}_t) + \frac{\beta - \alpha}{2}\mu^2\|\mathbf{x}^\star - \mathbf{x}_t\|^2 \quad \text{convexity} \\
&= -\mu h_t + \frac{\beta - \alpha}{2}\mu^2\|\mathbf{x}^\star - \mathbf{x}_t\|^2.
\end{aligned}$$

Where the equality is by writing $\mathbf{x}$ as $\mathbf{x} = (1-\mu)\mathbf{x}_t + \mu\mathbf{x}^\star$. By strong

convexity, we have for any $\mathbf{x}_t$ and the minimizer $\mathbf{x}^\star$:

$$\begin{aligned} h_t &= f(\mathbf{x}_t) - f(\mathbf{x}^\star) \\ &\geq \nabla f(\mathbf{x}^\star)^\top (\mathbf{x}_t - \mathbf{x}^\star) + \frac{\alpha}{2}\|\mathbf{x}^\star - \mathbf{x}_t\|^2 && \alpha\text{-strong convexity} \\ &\geq \frac{\alpha}{2}\|\mathbf{x}^\star - \mathbf{x}_t\|^2. && \text{optimality Thm 2.2} \end{aligned}$$

Thus, plugging into the above, we get

$$\begin{aligned} h_{t+1} - h_t &\leq (-\mu + \frac{\beta - \alpha}{\alpha}\mu^2) h_t \\ &\leq -\frac{\alpha}{4(\beta - \alpha)} h_t. && \text{optimal choice of } \mu \end{aligned}$$

Thus,

$$h_{t+1} \leq h_t(1 - \frac{\alpha}{4(\beta - \alpha)}) \leq h_t(1 - \frac{\alpha}{4\beta}) \leq h_t e^{-\gamma/4}.$$

This gives the theorem statement by induction. □

2.4 Reductions to Non-smooth and Non-strongly Convex Functions

The previous section dealt with γ-well-conditioned functions, which may seem like a significant restriction over vanilla convexity. Indeed, many interesting convex functions are not strongly convex nor smooth, and as we have seen, the convergence rate of gradient descent greatly differs for these functions. We have completed the picture for unconstrained optimization, and in this section we complete it for a bounded set.

The literature on first order methods is abundant with specialized analyses that explore the convergence rate of gradient descent for more general functions. In this manuscript we take a different approach: instead of analyzing variants of GD from scratch, we use reductions to derive near-optimal convergence rates for smooth functions that are not strongly convex, or strongly convex functions that are not smooth, or general convex functions without any further restrictions.

While attaining sub-optimal convergence bounds (by logarithmic factors), the advantage of this approach is two-fold: first, the reduction method is very simple to state and analyze, and its analysis is significantly shorter than analyzing GD from scratch. Second, the reduction method is generic, and thus extends to the analysis of accelerated gradient descent (or any other first order method) along the same lines. We turn to these reductions next.

2.4.1 Reduction to smooth, non strongly convex functions

Our first reduction applies the GD algorithm to functions that are β-smooth but not strongly convex.

The idea is to add a controlled amount of strong convexity to the function f, and then apply the previous algorithm to optimize the new function. The solution is distorted by the added strong convexity, but a tradeoff guarantees a meaningful convergence rate.

Algorithm 5 Gradient descent, reduction to β-smooth functions

1: Input: f, T, $\mathbf{x}_1 \in \mathcal{K}$, parameter $\tilde{\alpha}$.
2: Let $g(\mathbf{x}) = f(\mathbf{x}) + \frac{\tilde{\alpha}}{2}\|\mathbf{x} - \mathbf{x}_1\|^2$
3: Apply Algorithm 4 with parameters $g, T, \{\eta_t = \frac{1}{\beta}\}, \mathbf{x}_1$, return $\mathbf{x}_T$.

Lemma 2.7. *For β-smooth convex functions, Algorithm 5 with parameter $\tilde{\alpha} = \frac{\beta \log t}{D^2 t}$ converges as*

$$h_{t+1} = O\left(\frac{\beta \log t}{t}\right)$$

Proof. The function g is $\tilde{\alpha}$-strongly convex and $(\beta + \tilde{\alpha})$-smooth (see exercises). Thus, it is $\gamma = \frac{\tilde{\alpha}}{\tilde{\alpha}+\beta}$-well-conditioned. Notice that

$$\begin{aligned} h_t &= f(\mathbf{x}_t) - f(\mathbf{x}^\star) \\ &= g(\mathbf{x}_t) - g(\mathbf{x}^\star) + \frac{\tilde{\alpha}}{2}(\|\mathbf{x}^\star - \mathbf{x}_1\|^2 - \|\mathbf{x}_t - \mathbf{x}_1\|^2) \\ &\le h_t^g + \tilde{\alpha} D^2. && \text{def of } D, \S 2.1 \end{aligned}$$

Here, we denote $h_t^g = g(\mathbf{x}_t) - g(\mathbf{x}^\star)$. Since $g(\mathbf{x})$ is $\frac{\tilde{\alpha}}{\tilde{\alpha}+\beta}$-well-conditioned,

$$\begin{aligned} h_{t+1} &\le h_{t+1}^g + \tilde{\alpha} D^2 \\ &\le h_1^g e^{-\frac{\tilde{\alpha} t}{4(\tilde{\alpha}+\beta)}} + \tilde{\alpha} D^2 && \text{Theorem 2.6} \\ &= O(\frac{\beta \log t}{t}), && \text{choosing } \tilde{\alpha} = \frac{\beta \log t}{D^2 t} \end{aligned}$$

where we ignore constants and terms depending on D and h_1^g. □

Stronger convergence rates of $O(\frac{\beta}{t})$ can be obtained by analyzing GD from scratch, and these are known to be tight. Thus, our reduction is suboptimal by a factor of $O(\log T)$, which we tolerate for the reasons stated at the beginning of this section.

2.4.2 Reduction to strongly convex, non-smooth functions

Our reduction from non-smooth functions to γ-well-conditioned functions is similar in spirit to the one of the previous subsection. However, whereas for strong convexity the obtained rates were off by a factor of $\log T$, in this section we will also be off by factor of d, the dimension of the decision variable $\mathbf{x}$, as compared to the standard analyses in convex optimization. For tight bounds, the reader is referred to the excellent reference books and surveys listed in the bibliography section.

Algorithm 6 Gradient descent, reduction to non-smooth functions

1: Input: $f, \mathbf{x}_1, T, \delta$
2: Let $\hat{f}_\delta(\mathbf{x}) = \mathbf{E}_{\mathbf{v}\sim\mathbb{B}}\left[f(\mathbf{x}+\delta\mathbf{v})\right]$
3: Apply Algorithm 4 on $\hat{f}_\delta, \mathbf{x}_1, T, \{\eta_t = \delta\}$, return $\mathbf{x}_T$

We apply the GD algorithm to a smoothed variant of the objective function. In contrast to the previous reduction, smoothing cannot be obtained by simple addition of a smooth (or any other) function. Instead, we need a smoothing operation. The one we describe is particularly simple and amounts to taking a local integral of the function. More sophisticated, but less general, smoothing operators exist that are based on the Moreau-Yoshida regularization, see bibliographic section for more details.

Let f be G-Lipschitz continuous and α-strongly convex. Define for any $\delta > 0$,

$$S_\delta[f] : \mathbb{R}^d \mapsto \mathbb{R} \quad , \quad S_\delta[f](\mathbf{x}) = \mathop{\mathbf{E}}_{\mathbf{v}\sim\mathbb{B}}\left[f(\mathbf{x}+\delta\mathbf{v})\right],$$

where $\mathbb{B} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq 1\}$ is the Euclidean ball and $\mathbf{v} \sim \mathbb{B}$ denotes a random variable drawn from the uniform distribution over $\mathbb{B}$. When the function f is clear from the context, we henceforth use the simpler notation $\hat{f}_\delta = S_\delta[f]$.

We will prove that the function $\hat{f}_\delta = S_\delta[f]$ is a smooth approximation to $f : \mathbb{R}^d \mapsto \mathbb{R}$, i.e., it is both smooth and close in value to f, as given in the following lemma.

Lemma 2.8. *$\hat{f}_\delta$ has the following properties:*

1. *If f is α-strongly convex, then so is $\hat{f}_\delta$*
2. *$\hat{f}_\delta$ is $\frac{dG}{\delta}$-smooth*
3. *$|\hat{f}_\delta(\mathbf{x}) - f(\mathbf{x})| \leq \delta G$ for all $\mathbf{x} \in \mathcal{K}$.*

Before proving this lemma, let us first complete the reduction. Using Lemma 2.8 and the convergence for γ-well-conditioned functions the following approximation bound is obtained.

Lemma 2.9. *For $\delta = \frac{dG}{\alpha}\frac{\log t}{t}$ Algorithm 6 converges as*

$$h_t = O\left(\frac{G^2 d \log t}{\alpha t}\right).$$

Before proving this lemma, notice that the gradient descent method is applied with gradients of the smoothed function $\hat{f}_\delta$ rather than gradients of the original objective f. In this section we ignore the computational cost of computing such gradients given only access to gradients of f, which may be significant. Techniques for estimating these gradients are further explored in chapter 6.

Proof. Note that by Lemma 2.8 the function $\hat{f}_\delta$ is γ-well-conditioned for $\gamma = \frac{\alpha\delta}{dG}$.

$$\begin{aligned}
h_{t+1} &= f(\mathbf{x}_{t+1}) - f(\mathbf{x}^\star) \\
&\le \hat{f}_\delta(\mathbf{x}_{t+1}) - \hat{f}_\delta(\mathbf{x}^\star) + 2\delta G && \text{Lemma 2.8} \\
&\le h_1 e^{-\frac{\gamma t}{4}} + 2\delta G && \text{Theorem 2.6} \\
&= h_1 e^{-\frac{\alpha t \delta}{4dG}} + 2\delta G && \gamma = \tfrac{\alpha\delta}{dG} \text{ by Lemma 2.8} \\
&= O\left(\frac{dG^2 \log t}{\alpha t}\right). && \delta = \tfrac{dG}{\alpha}\tfrac{\log t}{t}
\end{aligned}$$

□

We proceed to prove that $\hat{f}_\delta$ is indeed a good approximation to the original function.

Proof of Lemma 2.8. First, since $\hat{f}_\delta$ is an average of α-strongly convex functions, it is also α-strongly convex. In order to prove smoothness, we will use Stokes' theorem from calculus: For all $\mathbf{x} \in \mathbb{R}^d$ and for a vector random variable $\mathbf{v}$ which is uniformly distributed over the Euclidean sphere $\mathbb{S} = \{\mathbf{y} \in \mathbb{R}^d : \|\mathbf{y}\| = 1\}$,

$$\mathop{\mathbf{E}}_{\mathbf{v}\sim\mathbb{S}}[f(\mathbf{x} + \delta\mathbf{v})\mathbf{v}] = \frac{\delta}{d}\nabla\hat{f}_\delta(\mathbf{x}). \tag{2.5}$$

Recall that a function f is β-smooth if and only if for all $\mathbf{x}, \mathbf{y} \in \mathcal{K}$, it holds that $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq \beta \|\mathbf{x} - \mathbf{y}\|$. Now,

$$\begin{aligned}
&\|\nabla \hat{f}_\delta(\mathbf{x}) - \nabla \hat{f}_\delta(\mathbf{y})\| = \\
&= \frac{d}{\delta} \| \mathop{\mathbf{E}}_{\mathbf{v} \sim \mathbb{S}} [f(\mathbf{x} + \delta \mathbf{v})\mathbf{v}] - \mathop{\mathbf{E}}_{\mathbf{v} \sim \mathbb{S}} [f(\mathbf{y} + \delta \mathbf{v})\mathbf{v}] \| && \text{by (2.5)} \\
&= \frac{d}{\delta} \| \mathop{\mathbf{E}}_{\mathbf{v} \sim \mathbb{S}} [f(\mathbf{x} + \delta \mathbf{v})\mathbf{v} - f(\mathbf{y} + \delta \mathbf{v})\mathbf{v}] \| && \text{linearity of expectation} \\
&\leq \frac{d}{\delta} \mathop{\mathbf{E}}_{\mathbf{v} \sim \mathbb{S}} \| f(\mathbf{x} + \delta \mathbf{v})\mathbf{v} - f(\mathbf{y} + \delta \mathbf{v})\mathbf{v} \| && \text{Jensen's inequality} \\
&\leq \frac{dG}{\delta} \|\mathbf{x} - \mathbf{y}\| \mathop{\mathbf{E}}_{\mathbf{v} \sim \mathbb{S}} [\|\mathbf{v}\|] && \text{Lipschitz continuity} \\
&= \frac{dG}{\delta} \|\mathbf{x} - \mathbf{y}\|. && \mathbf{v} \in \mathbb{S}
\end{aligned}$$

This proves the second property of Lemma 2.8. We proceed to show the third property, namely that $\hat{f}_\delta$ is a good approximation to f.

$$\begin{aligned}
|\hat{f}_\delta(\mathbf{x}) - f(\mathbf{x})| &= \left| \mathop{\mathbf{E}}_{\mathbf{v} \sim \mathbb{B}} [f(\mathbf{x} + \delta \mathbf{v})] - f(\mathbf{x}) \right| && \text{definition of } \hat{f}_\delta \\
&\leq \mathop{\mathbf{E}}_{\mathbf{v} \sim \mathbb{B}} [|f(\mathbf{x} + \delta \mathbf{v}) - f(\mathbf{x})|] && \text{Jensen's inequality} \\
&\leq \mathop{\mathbf{E}}_{\mathbf{v} \sim \mathbb{B}} [G \|\delta \mathbf{v}\|] && f \text{ is } G\text{-Lipschitz} \\
&\leq G\delta. && \mathbf{v} \in \mathbb{B}
\end{aligned}$$

□

We note that GD variants for α-strongly convex functions, even without the smoothing approach used in our reduction, are known to converge quickly and without dependence on the dimension. We state the known algorithm and result here without proof (see bibliography for references).

Theorem 2.10. *Let f be α-strongly convex, and let $\mathbf{x}_1, ..., \mathbf{x}_t$ be the iterates of Algorithm 4 applied to f with $\eta_t = \frac{2}{\alpha(t+1)}$. Then*

$$f\left(\frac{1}{t} \sum_{s=1}^{t} \frac{2s}{t+1} \mathbf{x}_s\right) - f(\mathbf{x}^\star) \leq \frac{2G^2}{\alpha(t+1)}.$$

2.4.3 Reduction to general convex functions

One can apply both reductions simultaneously to obtain a rate of $\tilde{O}(\frac{d}{\sqrt{t}})$. While near-optimal in terms of the number of iterations, the weakness of this bound lies in its dependence on the dimension. In the next chapter we shall show a rate of $O(\frac{1}{\sqrt{t}})$ as a direct consequence of a more general online convex optimization algorithm.

2.5 Example: Support Vector Machine Training

To illustrate the usfulness of the simple gradient descent algorithm of the previous sections, let us describe an optimization problem that has gained much attention in machine learning and can be solved efficiently using the methods we have just analyzed.

A very basic and successful learning paradigm is the linear classification model. In this model, the learner is presented with positive and negative examples of a concept. Each example, denoted by $\mathbf{a}_i$, is represented in Euclidean space by a d dimensional feature vector. For example, a common representation for emails in the spam-classification problem are binary vectors in Euclidean space, where the dimension of the space is the number of words in the language. The i'th email is a vector $\mathbf{a}_i$ whose entries are given as ones for coordinates corresponding to words that appear in the email, and zero otherwise[7]. In addition, each example has a label $b_i \in \{-1, +1\}$, corresponding to whether the email has been labeled spam/not spam. The goal is to find a hyperplane separating the two classes of vectors: those with positive labels and those with negative labels. If such a hyperplane, which completely separates the training set according to the labels, does not exist, then the goal is to find a hyperplane that achieves a separation of the training set with the smallest number of mistakes.

Mathematically speaking, given a set of n examples to train on, we seek $\mathbf{x} \in \mathbb{R}^d$ that minimizes the number of incorrectly classified examples, i.e.

$$\min_{\mathbf{x}\in\mathbb{R}^d} \sum_{i\in[n]} \delta(\text{sign}(\mathbf{x}^\top \mathbf{a}_i) \neq b_i) \tag{2.6}$$

where $\text{sign}(x) \in \{-1, +1\}$ is the sign function, and $\delta(z) \in \{0, 1\}$ is the indicator function that takes the value 1 if the condition z is satisfied and zero otherwise.

This optimization problem, which is at the heart of the linear classification formulation, is NP-hard, and in fact NP-hard to even approximate

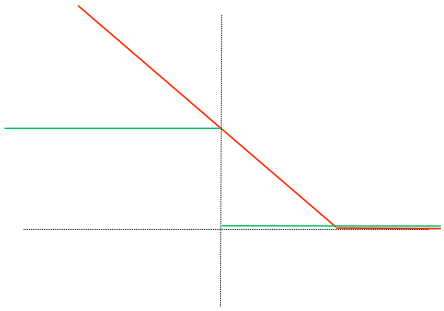

Figure 2.4: The hinge loss function versus the 0/1 loss function

non-trivially [8]. However, in the special case that a linear classifier (a hyperplane $\mathbf{x}$) that classifies all of the examples correctly exists, the problem is solvable in polynomial time via linear programming.

Various relaxations have been proposed to solve the more general case, when no perfect linear classifier exists. One of the most successful in practice is the Support Vector Machine (SVM) formulation.

The soft margin SVM relaxation replaces the 0/1 loss in (2.6) with a convex loss function, called the hinge-loss, given by

$$\ell_{\mathbf{a},b}(\mathbf{x}) = \text{hinge}(b \cdot \mathbf{x}^\top \mathbf{a}) = \max\{0, 1 - b \cdot \mathbf{x}^\top \mathbf{a}\}.$$

In figure 2.4 we depict how the hinge loss is a convex relaxation for the non-convex 0/1 loss. Further, the SVM formulation adds to the loss minimization objective a term that regularizes the size of the elements in $\mathbf{x}$. The reason and meaning of this additional term shall be addressed in later sections. For now, let us consider the SVM convex program:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ \lambda \frac{1}{n} \sum_{i\in[n]} \ell_{\mathbf{a}_i,b_i}(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}\|^2 \right\} \tag{2.7}$$

This is an unconstrained non-smooth and strongly convex program. It follows from Theorems 2.3 and 2.10 that $O(\frac{1}{\varepsilon})$ iterations suffice to attain an ε-approximate solution. We spell out the details of applying the subgradient descent algorithm to this formulation in Algorithm 7.

Notice that the learning rates are left unspecified, even though they can be explicitly set as in Theorem 2.10, or using the Polyak rate. The Polyak

Algorithm 7 SVM training via subgradient descent

1: Input: training set of n examples $\{(\mathbf{a}_i, b_i)\}$, T, learning rates $\{\eta_t\}$, initial $\mathbf{x}_1 = 0$.
2: **for** $t = 1$ to T **do**
3: Let $\nabla_t = \lambda \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_{\mathbf{a}_i, b_i}(\mathbf{x}_t) + \mathbf{x}_t$ where

$$\nabla \ell_{\mathbf{a}_i, b_i}(\mathbf{x}) = \begin{cases} 0, & b_i \mathbf{x}^\top \mathbf{a}_i > 1 \\ -b_i \mathbf{a}_i, & \text{otherwise} \end{cases}$$

4: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_t$ for $\eta_t = \frac{2}{t+1}$
5: **end for**
6: **return** $\bar{\mathbf{x}}_T = \frac{1}{T} \sum_{t=1}^{T} \frac{2t}{T+1} \mathbf{x}_t$

rate requires knowing the function value at optimality, although this can be relaxed (see bibliography).

A caveat of using gradient descent for SVM is the requirement to compute the full gradient, which may require a full pass over the data for each iteration. We will see a significantly more efficient algorithm in the next chapter!

2.6 Bibliographic Remarks

The reader is referred to dedicated books on convex optimization for much more in-depth treatment of the topics surveyed in this background chapter. For background in convex analysis see the texts [Borwein and Lewis, 2006, Rockafellar, 1997]. The classic textbook of Boyd and Vandenberghe [2004] gives a broad introduction to convex optimization with numerous applications, see also [Boyd, 2014]. For detailed rigorous convergence proofs and in depth analysis of first order methods, see lecture notes by Nesterov [2004] and books by Nemirovski and Yudin [1983], Nemirovskii [2004], as well as more recent lecture notes and texts [Bubeck, 2015, Hazan, 2019]. Theorem 2.10 is taken from [Bubeck, 2015] Theorem 3.9.

The logarithmic overhead in the reductions of section 2.4 can be removed with a more careful reduction and analysis, for details see [Allen-Zhu and Hazan, 2016]. A more sophisticated smoothing operator is the Moreau-Yoshida regularization: it avoids the dimension factor loss. However, it is sometimes less computationally efficient to work with [Parikh and Boyd, 2014].

The Polyak learning rate is detailed in [Polyak, 1987]. A recent exposition allows obtaining the same optimal rate without knowledge of the optimal function value [Hazan and Kakade, 2019].

Using linear separators and halfspaces to learn and separate data was considered in the very early days of AI [Rosenblatt, 1958, Minsky and Papert, 1969]. Notable the Perceptron algorithm was one of the first learning algorithms, and closely related to gradient descent. Support vector machines were introduced in [Cortes and Vapnik, 1995, Boser et al., 1992], see also the book of Schölkopf and Smola [2002].

Learning halfspaces with the zero-one loss is computationally hard, and hard to even approximate non-trivially [Daniely, 2016]. Proving that a problem is hard to approximate is at the forefront of computational complexity, and based on novel characterizations of the complexity class NP [Arora and Barak, 2009].

2.7 Exercises

1. Prove that a differentiable function $f(x) : \mathbb{R} \to \mathbb{R}$ is convex if and only if for any $x, y \in \mathbb{R}$ it holds that $f(x) - f(y) \leq (x - y)f'(x)$.

2. Recall that we say that a function $f : \mathbb{R}^n \to \mathbb{R}$ has a condition number $\gamma = \alpha/\beta$ over $K \subseteq \mathbb{R}^d$ if the following two inequalities hold for all $\mathbf{x}, \mathbf{y} \in \mathcal{K}$:

(a) $f(\mathbf{y}) \geq f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^\top \nabla f(\mathbf{x}) + \frac{\alpha}{2}\|\mathbf{x} - \mathbf{y}\|^2$

(b) $f(\mathbf{y}) \leq f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^\top \nabla f(\mathbf{x}) + \frac{\beta}{2}\|\mathbf{x} - \mathbf{y}\|^2$ Prove that if f is twice differentiable and it holds that $\beta \mathbf{I} \succcurlyeq \nabla^2 f(\mathbf{x}) \succcurlyeq \alpha \mathbf{I}$ for any $\mathbf{x} \in \mathcal{K}$, then the condition number of f over $\mathcal{K}$ is α/β.

3. Prove:

(a) The sum of convex functions is convex.

(b) Let f be α_1-strongly convex and g be α_2-strongly convex. Then $f + g$ is $(\alpha_1 + \alpha_2)$-strongly convex.

(c) Let f be β_1-smooth and g be β_2-smooth. Then $f + g$ is $(\beta_1 + \beta_2)$-smooth.

4. Let $\mathcal{K} \subseteq \mathbb{R}^d$ be bounded and closed. Prove that convexity of $\mathcal{K}$ is a necessary and sufficient condition for all $\mathbf{x} \in \mathbb{R}^d$ for $\Pi_K(\mathbf{x})$ to be a singleton, that is for $|\Pi_K(\mathbf{x})| = 1$. To prove that this is a necessary condition, it is enough to provide a counterexample.

5. Consider the n-dimensional simplex

$$\Delta_n = \{\mathbf{x} \in \mathbb{R}^n \mid \sum_{i=1}^{n} \mathbf{x}_i = 1,\, \mathbf{x}_i \geq 0 \ ,\ \forall i \in [n]\}.$$

Give an algorithm for computing the projection of a point $\mathbf{x} \in \mathbb{R}^n$ onto the set Δ_n (a near-linear time algorithm exists).

6. Prove the following identity:

$$\begin{aligned} &\arg\min_{\mathbf{x} \in \mathcal{K}} \left\{ \nabla_t^\top (\mathbf{x} - \mathbf{x}_t) + \frac{1}{2\eta_t} \|\mathbf{x} - \mathbf{x}_t\|^2 \right\} \\ =&\arg\min_{\mathbf{x} \in \mathcal{K}} \left\{ \|\mathbf{x} - (\mathbf{x}_t - \eta_t \nabla_t)\|^2 \right\}. \end{aligned}$$

7. Let $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ be a convex differentiable function and $\mathcal{K} \subseteq \mathbb{R}^n$ be a convex set. Prove that $\mathbf{x}^\star \in \mathcal{K}$ is a minimizer of f over $\mathcal{K}$ if and only if for any $\mathbf{y} \in \mathcal{K}$ it holds that $(\mathbf{y} - \mathbf{x}^\star)^\top \nabla f(\mathbf{x}^\star) \geq 0$.

8. * Extending Nesterov's accelerated GD algorithm:
Assume a black-box access to Nesterov's algorithm that attains the rate of $e^{-\sqrt{\gamma}\ T}$ for γ-well-conditioned functions, as in Table 2.1. Apply a reduction to obtain the $\frac{\beta}{T^2}$ rate for β-smooth functions, as in Table 2.1, up to logarithmic factors.

Chapter 3

First-Order Algorithms for Online Convex Optimization

In this chapter we describe and analyze the most simple and basic algorithms for online convex optimization (recall the definition of the model as introduced in chapter 1), which are also surprisingly useful and applicable in practice. We use the same notation introduced in §2.1. However, in contrast to the previous chapter, the goal of the algorithms introduced in this chapter is to minimize *regret*, rather than the optimization error (which is ill-defined in an online setting).

Recall the definition of regret in an OCO setting, as given in equation (1.1), with subscripts, superscripts and the supremum over the function class omitted when they are clear from the context:

$$\mathrm{Regret}_T = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}).$$

Table 3.1 details known upper and lower bounds on the regret for different types of convex functions as it depends on the number of prediction iterations.

In order to compare regret to optimization error it is useful to consider the average regret, or Regret/T. Let $\bar{\mathbf{x}}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}_t$ be the average decision. If the functions f_t are all equal to a single function $f : \mathcal{K} \mapsto \mathbb{R}$, then Jensen's inequality implies that $f(\bar{\mathbf{x}}_T)$ converges to $f(\mathbf{x}^\star)$ at a rate at most the average regret, since

$$f(\bar{\mathbf{x}}_T) - f(\mathbf{x}^\star) \leq \frac{1}{T}\sum_{t=1}^{T}[f(\mathbf{x}_t) - f(\mathbf{x}^\star)] = \frac{\mathrm{Regret}_T}{T}.$$

	α-strongly convex	β-smooth	δ-exp-concave
Upper bound	$\frac{1}{\alpha}\log T$	$\sqrt{T}$	$\frac{n}{\delta}\log T$
Lower bound	$\frac{1}{\alpha}\log T$	$\sqrt{T}$	$\frac{n}{\delta}\log T$
Average regret	$\frac{\log T}{\alpha T}$	$\frac{1}{\sqrt{T}}$	$\frac{n\log T}{\delta T}$

Table 3.1: Attainable asymptotic regret bounds for loss function classes.

The reader may compare to Table 2.1 of offline convergence of first order methods: as opposed to offline optimization, smoothness does not improve asymptotic regret rates. However, exp-concavity, a weaker property than strong convexity, comes into play and gives improved regret rates.

This chapter will present algorithms and lower bounds that realize the above known results for OCO. The property of exp-concavity and its applications, as well as logarithmic regret algorithms for exp-concave functions are deferred to the next chapter.

3.1 Online Gradient Descent

Perhaps the simplest algorithm that applies to the most general setting of online convex optimization is online gradient descent. This algorithm, which is based on standard gradient descent from offline optimization, was introduced in its online form by Zinkevich (see bibliography at the end of this section).

Algorithm 8 online gradient descent

1: Input: convex set $\mathcal{K}$, T, $\mathbf{x}_1 \in \mathcal{K}$, step sizes $\{\eta_t\}$
2: **for** $t = 1$ to T **do**
3: Play $\mathbf{x}_t$ and observe cost $f_t(\mathbf{x}_t)$.
4: Update and project:

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \eta_t \nabla f_t(\mathbf{x}_t)$$
$$\mathbf{x}_{t+1} = \prod_{\mathcal{K}}(\mathbf{y}_{t+1})$$

5: **end for**

Pseudo-code for the algorithm is given in Algorithm 8, and a conceptual illustration is given in figure 3.1.

In each iteration, the algorithm takes a step from the previous point in

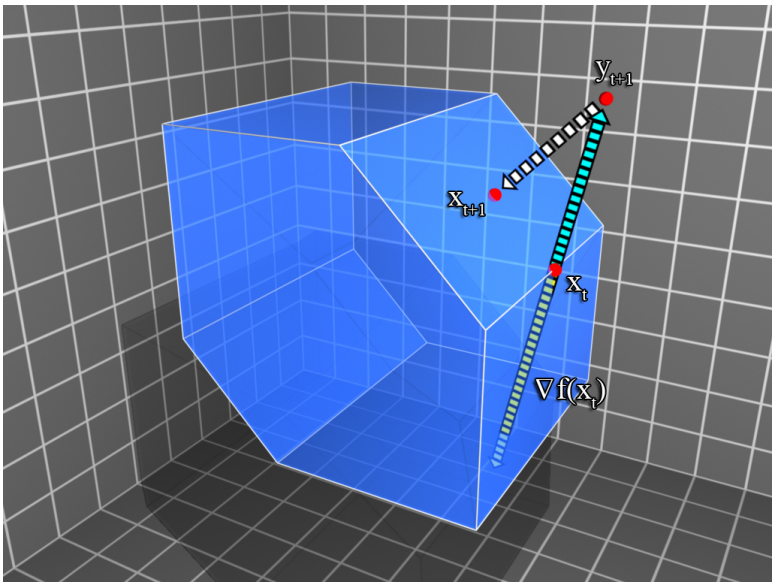

Figure 3.1: OGD: the iterate $\mathbf{x}_{t+1}$ is derived by advancing $\mathbf{x}_t$ in the direction of the current gradient ∇_t, and projecting back into $\mathcal{K}$

the direction of the gradient of the previous cost. This step may result in a point outside of the underlying convex set. In such cases, the algorithm projects the point back to the convex set, i.e. finds its closest point in the convex set. Despite the fact that the next cost function may be completely different than the costs observed thus far, the regret attained by the algorithm is sublinear. This is formalized in the following theorem (recall the definition of G and D from the previous chapter).

Theorem 3.1. *Online gradient descent with step sizes* $\{\eta_t = \frac{D}{G\sqrt{t}},\ t \in [T]\}$ *guarantees the following for all* $T \geq 1$*:*

$$\text{Regret}_T = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}^\star \in \mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \ \leq \frac{3}{2} GD\sqrt{T}.$$

Proof. Let $\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x})$. Define $\nabla_t \stackrel{\text{def}}{=} \nabla f_t(\mathbf{x}_t)$. By convexity

$$f_t(\mathbf{x}_t) - f_t(\mathbf{x}^\star) \leq \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star) \tag{3.1}$$

We first upper-bound $\nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star)$ using the update rule for $\mathbf{x}_{t+1}$ and The-

orem 2.1 (the Pythagorean theorem):

$$\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \;=\; \left\|\prod_{\mathcal{K}}(\mathbf{x}_t - \eta_t \nabla_t) - \mathbf{x}^\star\right\|^2 \leq \|\mathbf{x}_t - \eta_t \nabla_t - \mathbf{x}^\star\|^2 . \tag{3.2}$$

Hence,

$$\begin{aligned} \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 &\leq \|\mathbf{x}_t - \mathbf{x}^\star\|^2 + \eta_t^2 \|\nabla_t\|^2 - 2\eta_t \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star) \\ 2\nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star) &\leq \frac{\|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2}{\eta_t} + \eta_t G^2. \end{aligned} \tag{3.3}$$

Summing (3.1) and (3.3) from $t = 1$ to T, and setting $\eta_t = \frac{D}{G\sqrt{t}}$ (with $\frac{1}{\eta_0} \stackrel{\text{def}}{=} 0$):

$$\begin{aligned} &2\left(\sum_{t=1}^T f_t(\mathbf{x}_t) - f_t(\mathbf{x}^\star)\right) \leq 2\sum_{t=1}^T \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star) \\ &\leq \sum_{t=1}^T \frac{\|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2}{\eta_t} + G^2 \sum_{t=1}^T \eta_t \\ &\leq \sum_{t=1}^T \|\mathbf{x}_t - \mathbf{x}^\star\|^2 \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) + G^2 \sum_{t=1}^T \eta_t && \frac{1}{\eta_0} \stackrel{\text{def}}{=} 0, \\ & && \|\mathbf{x}_{T+1} - \mathbf{x}^*\|^2 \geq 0 \\ &\leq D^2 \sum_{t=1}^T \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) + G^2 \sum_{t=1}^T \eta_t \\ &\leq D^2 \frac{1}{\eta_T} + G^2 \sum_{t=1}^T \eta_t && \text{telescoping series} \\ &\leq 3DG\sqrt{T}. \end{aligned}$$

The last inequality follows since $\eta_t = \frac{D}{G\sqrt{t}}$ and $\sum_{t=1}^T \frac{1}{\sqrt{t}} \leq 2\sqrt{T}$. □

The online gradient descent algorithm is straightforward to implement, and updates take linear time given the gradient. However, there is a projection step which may take significantly longer, as discussed in §2.1.1 and chapter 7.

3.2 Lower Bounds

The previous section introduces and analyzes a very simple and natural approach to online convex optimization. Before continuing our venture, it is worthwhile to consider whether the previous bound can be improved? We measure performance of OCO algorithms both by regret and by computational efficiency. Therefore, we ask ourselves whether even simpler algorithms that attain tighter regret bounds exist.

The computational efficiency of online gradient descent seemingly leaves little room for improvement, putting aside the projection step it runs in linear time per iteration. What about obtaining better regret?

Perhaps surprisingly, the answer is negative: online gradient descent attains, in the worst case, tight regret bounds up to small constant factors! This is formally given in the following theorem.

Theorem 3.2. *Any algorithm for online convex optimization incurs* $\Omega(DG\sqrt{T})$ *regret in the worst case. This is true even if the cost functions are generated from a fixed stationary distribution.*

We give a sketch of the proof; filling in all details is left as an exercise at the end of this chapter.

Consider an instance of OCO where the convex set $\mathcal{K}$ is the n-dimensional hypercube, i.e.

$$\mathcal{K} = \{\mathbf{x} \in \mathbb{R}^n \ , \ \|\mathbf{x}\|_\infty \leq 1\}.$$

There are 2^n linear cost functions, one for each vertex $\mathbf{v} \in \{\pm 1\}^n$, defined as

$$\forall \mathbf{v} \in \{\pm 1\}^n \ , \ f_{\mathbf{v}}(\mathbf{x}) = \mathbf{v}^\top \mathbf{x}.$$

Notice that both the diameter of $\mathcal{K}$ and the bound on the norm of the cost function gradients, denoted G, are bounded by

$$D \leq \sqrt{\sum_{i=1}^{n} 2^2} = 2\sqrt{n}, \ G \leq \sqrt{\sum_{i=1}^{n} (\pm 1)^2} = \sqrt{n}$$

The cost functions in each iteration are chosen at random, with uniform probability, from the set $\{f_{\mathbf{v}}, \mathbf{v} \in \{\pm 1\}^n\}$. Denote by $\mathbf{v}_t \in \{\pm 1\}^n$ the vertex chosen in iteration t, and denote $f_t = f_{\mathbf{v}_t}$. By uniformity and independence,

for any t and $\mathbf{x}_t$ chosen online, $\mathbf{E}_{\mathbf{v}_t}[f_t(\mathbf{x}_t)] = \mathbf{E}_{\mathbf{v}_t}[\mathbf{v}_t^\top \mathbf{x}_t] = 0$. However,

$$\begin{aligned}
\mathop{\mathbf{E}}_{\mathbf{v}_1,\ldots,\mathbf{v}_T}\left[\min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x})\right] &= \mathbf{E}\left[\min_{\mathbf{x}\in\mathcal{K}} \sum_{i\in[n]} \sum_{t=1}^{T} \mathbf{v}_t(i)\cdot \mathbf{x}_i\right] \\
&= n\,\mathbf{E}\left[-\left|\sum_{t=1}^{T} \mathbf{v}_t(1)\right|\right] \qquad \text{i.i.d. coordinates} \\
&= -\Omega(n\sqrt{T}).
\end{aligned}$$

The last equality is left as an exercise.

The facts above nearly complete the proof of Theorem 3.2; see the exercises at the end of this chapter.

3.3 Logarithmic Regret

At this point, the reader may wonder: we have introduced a seemingly sophisticated and obviously general framework for learning and prediction, as well as a linear-time algorithm for the most general case, complete with tight regret bounds, and done so with elementary proofs! Is this all OCO has to offer?

The answer to this question is two-fold:

1. Simple is good: the philosophy behind OCO treats simplicity as a merit. The main reason OCO has taken the stage in online learning in recent years is the simplicity of its algorithms and their analysis, which allow for numerous variations and tweaks in their host applications.

2. A very wide class of settings, which will be the subject of the next sections, admit more efficient algorithms, in terms of both regret and computational complexity.

In §2 we surveyed optimization algorithms with convergence rates that vary greatly according to the convexity properties of the function to be optimized. Do the regret bounds in online convex optimization vary as much as the convergence bounds in offline convex optimization over different classes of convex cost functions?

Indeed, next we show that for important classes of loss functions significantly better regret bounds are possible.

3.3.1 Online gradient descent for strongly convex functions

The first algorithm that achieves regret logarithmic in the number of iterations is a twist on the online gradient descent algorithm, changing only the step size. The following theorem establishes logarithmic bounds on the regret if the cost functions are strongly convex.

Theorem 3.3. *For α-strongly convex loss functions, online gradient descent with step sizes $\eta_t = \frac{1}{\alpha t}$ achieves the following guarantee for all $T \geq 1$*

$$\text{Regret}_T \;\leq\; \frac{G^2}{2\alpha}(1 + \log T).$$

Proof. Let $\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x})$. Recall the definition of regret

$$\text{Regret}_T \;= \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^\star).$$

Define $\nabla_t \stackrel{\text{def}}{=} \nabla f_t(\mathbf{x}_t)$. Applying the definition of α-strong convexity to the pair of points $\{\mathbf{x}_t,\mathbf{x}^*\}$, we have

$$2(f_t(\mathbf{x}_t) - f_t(\mathbf{x}^\star)) \;\leq\; 2\nabla_t^\top(\mathbf{x}_t - \mathbf{x}^\star) - \alpha\|\mathbf{x}^\star - \mathbf{x}_t\|^2. \tag{3.4}$$

We proceed to upper-bound $\nabla_t^\top(\mathbf{x}_t - \mathbf{x}^\star)$. Using the update rule for $\mathbf{x}_{t+1}$ and the Pythagorean theorem 2.1, we get

$$\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \;=\; \|\prod_{\mathcal{K}}(\mathbf{x}_t - \eta_t\nabla_t) - \mathbf{x}^\star\|^2 \leq \|\mathbf{x}_t - \eta_t\nabla_t - \mathbf{x}^\star\|^2.$$

Hence,

$$\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \quad\leq\quad \|\mathbf{x}_t - \mathbf{x}^\star\|^2 + \eta_t^2\|\nabla_t\|^2 - 2\eta_t\nabla_t^\top(\mathbf{x}_t - \mathbf{x}^\star)$$

and

$$2\nabla_t^\top(\mathbf{x}_t - \mathbf{x}^\star) \quad\leq\quad \frac{\|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2}{\eta_t} + \eta_t G^2. \tag{3.5}$$

Summing (3.5) from $t = 1$ to T, setting $\eta_t = \frac{1}{\alpha t}$ (define $\frac{1}{\eta_0} \stackrel{\text{def}}{=} 0$), and

combining with (3.4), we have:

$$\begin{aligned}
& 2\sum_{t=1}^{T}(f_t(\mathbf{x}_t) - f_t(\mathbf{x}^\star)) \\
\leq \quad & \sum_{t=1}^{T}\|\mathbf{x}_t - \mathbf{x}^\star\|^2\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} - \alpha\right) + G^2\sum_{t=1}^{T}\eta_t \\
& \text{since } \frac{1}{\eta_0} \stackrel{\text{def}}{=} 0, \|\mathbf{x}_{T+1} - \mathbf{x}^*\|^2 \geq 0 \\
= \quad & 0 + G^2\sum_{t=1}^{T}\frac{1}{\alpha t} \\
\leq \quad & \frac{G^2}{\alpha}(1 + \log T)
\end{aligned}$$

□

3.4 Application: Stochastic Gradient Descent

A special case of Online Convex Optimization is the well-studied setting of stochastic optimization. In stochastic optimization, the optimizer attempts to minimize a convex function over a convex domain as given by the mathematical program:

$$\min_{\mathbf{x}\in\mathcal{K}} f(\mathbf{x}).$$

However, unlike standard offline optimization, the optimizer is given access to a noisy gradient oracle, defined by

$$\mathcal{O}(\mathbf{x}) \stackrel{\text{def}}{=} \tilde{\nabla}_{\mathbf{x}} \text{ s.t. } \mathbf{E}[\tilde{\nabla}_{\mathbf{x}}] = \nabla f(\mathbf{x}) \ , \ \mathbf{E}[\|\tilde{\nabla}_{\mathbf{x}}\|^2] \leq G^2$$

That is, given a point in the decision set, a noisy gradient oracle returns a random vector whose expectation is the gradient at the point and whose variance is bounded by G^2.

We will show that regret bounds for OCO translate to convergence rates for stochastic optimization. As a special case, consider the online gradient descent algorithm whose regret is bounded by

$$\text{Regret}_T = O(DG\sqrt{T})$$

Applying the OGD algorithm over a sequence of linear functions that are defined by the noisy gradient oracle at consecutive points, and finally returning the average of all points along the way, we obtain the stochastic gradient descent algorithm, presented in Algorithm 9.

Algorithm 9 stochastic gradient descent

1: Input: $\mathcal{O}$,$\mathcal{K}$, T, $\mathbf{x}_1 \in \mathcal{K}$, step sizes $\{\eta_t\}$
2: **for** $t = 1$ to T **do**
3: Let $\tilde{\nabla}_t = \mathcal{O}(\mathbf{x}_t)$
4: Update and project:

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \eta_t \tilde{\nabla}_t$$

$$\mathbf{x}_{t+1} = \prod_{\mathcal{K}}(\mathbf{y}_{t+1})$$

5: **end for**
6: **return** $\bar{\mathbf{x}}_T \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}_t$

Theorem 3.4. *Algorithm 9 with step sizes $\eta_t = \frac{D}{G\sqrt{t}}$ guarantees*

$$\mathbf{E}[f(\bar{\mathbf{x}}_T)] \leq \min_{\mathbf{x}^\star \in \mathcal{K}} f(\mathbf{x}^\star) + \frac{3GD}{2\sqrt{T}}.$$

Proof. For the analysis, we define the linear functions $f_t(\mathbf{x}) \stackrel{\text{def}}{=} \tilde{\nabla}_t^\top \mathbf{x}$. Using the regret guarantee of OGD, we have

$$\begin{aligned}
&\mathbf{E}[f(\bar{\mathbf{x}}_T)] - f(\mathbf{x}^\star) \\
&\leq \mathbf{E}[\frac{1}{T}\sum_t f(\mathbf{x}_t)] - f(\mathbf{x}^\star) && \text{convexity of } f \text{ (Jensen)} \\
&\leq \frac{1}{T}\,\mathbf{E}[\sum_t \nabla f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^\star)] && \text{convexity again} \\
&= \frac{1}{T}\,\mathbf{E}[\sum_t \tilde{\nabla}_t^\top (\mathbf{x}_t - \mathbf{x}^\star)] && \text{noisy gradient estimator} \\
&= \frac{1}{T}\,\mathbf{E}[\sum_t f_t(\mathbf{x}_t) - f_t(\mathbf{x}^\star)] && \text{Algorithm 9, line (3)} \\
&\leq \frac{\text{Regret}_T}{T} && \text{definition} \\
&\leq \frac{3GD}{2\sqrt{T}} && \text{theorem 3.1}
\end{aligned}$$

□

It is important to note that in the proof above, we have used the fact that the regret bounds of online gradient descent hold against an adaptive adversary. This need arises since the cost functions f_t defined in Algorithm 9 depend on the choice of decision $\mathbf{x}_t \in \mathcal{K}$.

In addition, the careful reader may notice that by plugging in different step sizes (also called learning rates) and applying SGD to strongly convex functions, one can attain $\tilde{O}(1/T)$ convergence rates, where the $\tilde{O}$ notation hides logarithmic factors in T. Details of this derivation are left as an exercise.

3.4.1 Example: stochastic gradient descent for SVM training

Recall our example of Support Vector Machine training from §2.5. The task of training an SVM over a given data set amounts to solving the following convex program (equation (2.7)):

$$f(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^d} \left\{ \lambda \frac{1}{n} \sum_{i\in[n]} \ell_{\mathbf{a}_i,b_i}(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}\|^2 \right\}$$
$$\ell_{\mathbf{a},b}(\mathbf{x}) = \max\{0, 1 - b \cdot \mathbf{x}^\top \mathbf{a}\}.$$

Algorithm 10 SGD for SVM training

1: Input: training set of n examples $\{(\mathbf{a}_i, b_i)\}$, T. Set $\mathbf{x}_1 = 0$
2: **for** $t = 1$ to T **do**
3: Pick an example uniformly at random $t \in [n]$.
4: Let $\tilde{\nabla}_t = \lambda \nabla \ell_{\mathbf{a}_t,b_t}(\mathbf{x}_t) + \mathbf{x}_t$ where

$$\nabla \ell_{\mathbf{a}_t,b_t}(\mathbf{x}_t) = \begin{cases} 0, & b_t \mathbf{x}_t^\top \mathbf{a}_t > 1 \\ -b_t \mathbf{a}_t, & \text{otherwise} \end{cases}$$

5: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \tilde{\nabla}_t$
6: **end for**
7: **return** $\bar{\mathbf{x}}_T \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=1}^T \mathbf{x}_t$

Using the technique described in this chapter, namely the OGD and SGD algorithms, we can devise a much faster algorithm than the one presented in the previous chapter. The idea is to generate an unbiased estimator for

the gradient of the objective using a single example in the dataset, and use it in lieu of the entire gradient. This is given formally in the SGD algorithm for SVM training presented in Algorithm 10.

It follows from Theorem 3.4 that this algorithm, with appropriate parameters η_t, returns an ε-approximate solution after $T = O(\frac{1}{\varepsilon^2})$ iterations. Furthermore, with a little more care and using Theorem 3.3, a rate of $\tilde{O}(\frac{1}{\varepsilon})$ is obtained with parameters $\eta_t = O(\frac{1}{t})$.

This matches the convergence rate of standard offline gradient descent. However, observe that each iteration is significantly cheaper—only one example in the data set need be considered! That is the magic of SGD; we have matched the nearly optimal convergence rate of first order methods using extremely cheap iterations. This makes it the method of choice in numerous applications.

3.5 Bibliographic Remarks

The OCO framework was introduced by Zinkevich [2003], where the OGD algorithm was introduced and analyzed. Precursors to this algorithm, albeit for less general settings, were introduced and analyzed in [Kivinen and Warmuth, 1997]. Logarithmic regret algorithms for Online Convex Optimization were introduced and analyzed in [Hazan et al., 2007].

The stochastic gradient descent (SGD) algorithm dates back to Robbins and Monro [1951], where it was called "stochastic approximation". The importance of SGD for machine learning was advocated for in [Bottou, 1998, Bottou and Bousquet, 2008]. The literature on SGD is vast and the reader is referred to the text of Bubeck [2015] and paper by Lan [2012].

Application of SGD to soft-margin SVM training was explored in [Shalev-Shwartz et al., 2011a]. Tight convergence rates of SGD for strongly convex and non-smooth functions were only recently obtained in [Hazan and Kale, 2011, Rakhlin et al., 2012, Shamir and Zhang, 2013].

3.6 Exercises

1. Prove that SGD for a strongly convex function can, with appropriate parameters η_t, converge as $\tilde{O}(\frac{1}{T})$. Recall that the $\tilde{O}$ notation hides logarithmic factors in the parameters, including T. You may assume that the gradient estimators have Euclidean norms bounded by the constant G.

2. * In this exercise we show how to remove some a-priory knowledge from the design of online convex optimization algorithms.

(a) Design an OCO algorithm that attains the same asymptotic regret bound as OGD, up to factors logarithmic in G without knowing the parameter G ahead of time.

(b) Do the same for the parameter D: design an OCO algorithm that attains the same asymptotic regret bound as OGD, up to factors logarithmic in D without knowing the parameter D ahead of time. This time you may assume G is known. You may assume that it is possible to compute projections onto $\mathcal{K}$ without knowing its diameter.

3. In this exercise we prove a tight lower bound on the regret of any algorithm for online convex optimization.

(a) For any sequence of T fair coin tosses, let N_h be the number of head outcomes and N_t be the number of tails. Give an asymptotically tight upper and lower bound on $\mathbf{E}[|N_h - N_t|]$. That is, give an order of growth of this random variable as a function of T, up to multiplicative and additive constants.

(b) Consider a 2-expert problem, in which the losses are inversely correlated: either expert one incurs a loss of one and the second expert negative one, or vice versa. Use the fact above to design a setting in which any experts algorithm incurs regret asymptotically matching the upper bound.

(c) Consider the general OCO setting over a convex set $\mathcal{K}$. Design a setting in which the cost functions have gradients whose norm is bounded by G, and obtain a lower bound on the regret as a function of G, the diameter of $\mathcal{K}$, and the number of game iterations.

4. Implement the SGD algorithm for SVM training. Apply it on the MNIST dataset. Compare your results to the offline GD algorithm from the previous chapter.

Chapter 4

Second-Order Methods

The motivation for this chapter is the application of online portfolio selection, considered in the first chapter of this book. We begin with a detailed description of this application. We proceed to describe a new class of convex functions that model this problem. This new class of functions is more general than the class of strongly convex functions discussed in the previous chapter. It allows for logarithmic regret algorithms, which are based on second order methods from convex optimization. In contrast to first order methods, which have been our focus thus far and relied on (sub)gradients, second order methods exploit information about the second derivative of the objective function.

4.1 Motivation: Universal Portfolio Selection

In this subsection we give the formal definition of the universal portfolio selection problem that was informally described in §1.2.

4.1.1 Mainstream portfolio theory

Mainstream financial theory models stock prices as a stochastic process known as Geometric Brownian Motion (GBM). This model assumes that the fluctuations in the prices of the stocks behave essentially as a random walk. It is perhaps easier to think about a price of an asset (stock) on time segments, obtained from a discretization of time into equal segments. Thus, the logarithm of the price at segment $t+1$, denoted l_{t+1}, is given by the sum of the logarithm of the price at segment t and a Gaussian random variable

with a particular mean and variance,

$$l_{t+1} \sim l_t + \mathcal{N}(\mu, \sigma).$$

This is only an informal way of thinking about GBM. The formal model is continuous in time, roughly equivalent to the above as the time intervals, means and variances approach zero.

The GBM model gives rise to particular algorithms for portfolio selection (as well as more sophisticated applications such as options pricing). Given the means and variances of the stock prices over time of a set of assets, as well as their cross-correlations, a portfolio with maximal expected gain (mean) for a specific risk (variance) threshold can be formulated.

The fundamental question is, of course, how does one obtain the mean and variance parameters, not to mention the cross-correlations, of a given set of stocks? One accepted solution is to estimate these from historical data, e.g., by taking the recent history of stock prices.

4.1.2 Universal portfolio theory

The theory of universal portfolio selection is very different from the above. The main difference being the lack of statistical assumptions about the stock market. The idea is to model investing as a repeated decision making scenario, which fits nicely into our OCO framework, and to measure regret as a performance metric.

Consider the following scenario: at each iteration $t \in [T]$, the decision maker chooses $\mathbf{x}_t$, a distribution of her wealth over n assets, such that $\mathbf{x}_t \in \Delta_n$. Here $\Delta_n = \{\mathbf{x} \in \mathbb{R}^n_+, \sum_i \mathbf{x}_i = 1\}$ is the n-dimensional simplex, i.e., the set of all distributions over n elements. An adversary independently chooses market returns for the assets, i.e., a vector $\mathbf{r}_t \in \mathbb{R}^n_+$ such that each coordinate $\mathbf{r}_t(i)$ is the price ratio for the i'th asset between the iterations t and $t+1$. For example, if the i'th coordinate is the Google ticker symbol GOOG traded on the NASDAQ, then

$$\mathbf{r}_t(i) = \frac{\text{price of GOOG at time } t+1}{\text{price of GOOG at time } t}$$

How does the decision maker's wealth change? Let W_t be her total wealth at iteration t. Then, ignoring transaction costs, we have

$$W_{t+1} = W_t \cdot \mathbf{r}_t^\top \mathbf{x}_t$$

Over T iterations, the total wealth of the investor is given by

$$W_T = W_1 \cdot \prod_{t=1}^{T} \mathbf{r}_t^\top \mathbf{x}_t$$

The goal of the decision maker, to maximize the overall wealth gain W_T/W_0, can be attained by maximizing the following more convenient logarithm of this quantity, given by

$$\log \frac{W_T}{W_1} = \sum_{t=1}^{T} \log \mathbf{r}_t^\top \mathbf{x}_t$$

The above formulation is already very similar to our OCO setting, albeit phrased as a gain maximization rather than a loss minimization setting. Let

$$f_t(\mathbf{x}) = \log(\mathbf{r}_t^\top \mathbf{x})$$

The convex set is the n-dimensional simplex $\mathcal{K} = \Delta_n$, and define the regret to be

$$\text{Regret}_T = \max_{\mathbf{x}^\star \in \mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}^\star) - \sum_{t=1}^{T} f_t(\mathbf{x}_t)$$

The functions f_t are concave rather than convex, which is perfectly fine as we are framing the problem as a maximization rather than a minimization. Note also that the regret is the negation of the usual regret notion (1.1) we have considered for minimization problems.

Since this is an online convex optimization instance, we can use the online gradient descent algorithm from the previous chapter to invest, which ensures $O(\sqrt{T})$ regret (see exercises). What guarantee do we attain in terms of investing? To answer this, in the next section we reason about what $\mathbf{x}^\star$ in the above expression may be.

4.1.3 Constant rebalancing portfolios

As $\mathbf{x}^\star \in \mathcal{K} = \Delta_n$ is a point in the n-dimensional simplex, consider the special case of $\mathbf{x}^\star = \mathbf{e}_1$, i.e., the first standard basis vector (the vector that has zero in all coordinates except the first, which is set to one). The term $\sum_{t=1}^{T} f_t(\mathbf{e}_1)$ becomes $\sum_{t=1}^{T} \log \mathbf{r}_t(1)$, or

$$\log \prod_{t=1}^{T} \mathbf{r}_t(1) = \log \left(\frac{\text{price of stock at time } T+1}{\text{initial price of stock}} \right)$$

As T becomes large, any sublinear regret guarantee (e.g., the $O(\sqrt{T})$ regret guarantee achieved using online gradient descent) achieves an average regret that approaches zero. In this context, this implies that the log-wealth gain achieved (in average over T rounds) is as good as that of the first stock. Since $\mathbf{x}^\star$ can be taken to be any vector, sublinear regret guarantees average log-wealth growth as good as any stock!

However, $\mathbf{x}^\star$ can be significantly better, as shown in the following example. Consider a market of two stocks that fluctuate wildly. The first stock increases by 100% every even day and returns to its original price the following (odd) day. The second stock does exactly the opposite: decreases by 50% on even days and rises back on odd days. Formally, we have

$$\mathbf{r}_t(1) = (2\ ,\ \frac{1}{2}\ ,\ 2\ ,\ \frac{1}{2}, ...)$$

$$\mathbf{r}_t(2) = (\frac{1}{2}\ ,\ 2\ ,\ \frac{1}{2}\ ,\ 2\ , ...)$$

Clearly, any investment in either of the stocks will not gain in the long run. However, the portfolio $\mathbf{x}^\star = (0.5, 0.5)$ increases wealth by a factor of $\mathbf{r}_t^\top \mathbf{x}^\star = (\frac{1}{2})^2 + 1 = 1.25$ daily! Such a mixed distribution is called a fixed rebalanced portfolio, as it needs to rebalance the proportion of total capital invested in each stock at each iteration to maintain this fixed distribution strategy.

Thus, vanishing average regret guarantees long-run growth as the best constant rebalanced portfolio in hindsight. Such a portfolio strategy is called *universal*. We have seen that the online gradient descent algorithm gives essentially a universal algorithm with regret $O(\sqrt{T})$. Can we get better regret guarantees?

4.2 Exp-Concave Functions

For convenience, we return to considering losses of convex functions, rather than gains of concave functions as in the application for portfolio selection. The two problems are equivalent: we simply replace the maximization of the concave $f(\mathbf{x}) = \log(\mathbf{r}_t^\top \mathbf{x})$ with the minimization of the convex $f(\mathbf{x}) = -\log(\mathbf{r}_t^\top \mathbf{x})$.

In the previous chapter we have seen that the OGD algorithm with carefully chosen step sizes can deliver logarithmic regret for strongly convex functions. However, the loss function for the OCO setting of portfolio selection, $f_t(\mathbf{x}) = -\log(\mathbf{r}_t^\top \mathbf{x})$, is not strongly convex. Instead, the Hessian of

this function is given by

$$\nabla^2 f_t(\mathbf{x}) = \frac{\mathbf{r}_t \mathbf{r}_t^\top}{(\mathbf{r}_t^\top \mathbf{x})^2}$$

which is a rank one matrix. Recall that the Hessian of a twice-differentiable strongly convex function is larger than a multiple of identity matrix and is positive definite and in particular has full rank. Thus, the loss function above is quite far from being strongly convex.

However, an important observation is that this Hessian is large in the direction of the gradient. This property is called exp-concavity. We proceed to define this property rigorously and show that it suffices to attain logarithmic regret.

Definition 4.1. *A convex function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is defined to be α-exp-concave over $\mathcal{K} \subseteq \mathbb{R}^n$ if the function g is concave, where $g : \mathcal{K} \mapsto \mathbb{R}$ is defined as*

$$g(\mathbf{x}) = e^{-\alpha f(\mathbf{x})}$$

For the following discussion, recall the notation of §2.1, and in particular our convention over matrices that $A \succcurlyeq B$ if and only if $A - B$ is positive semidefinite. Exp-concavity implies strong-convexity in the direction of the gradient. This reduces to the following property:

Lemma 4.2. *A twice-differentiable function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is α-exp-concave at $\mathbf{x}$ if and only if*

$$\nabla^2 f(\mathbf{x}) \succcurlyeq \alpha \nabla f(\mathbf{x}) \nabla f(\mathbf{x})^\top.$$

The proof of this lemma is given as a guided exercise at the end of this chapter. We prove a slightly stronger lemma below.

Lemma 4.3. *Let $f : \mathcal{K} \to \mathbb{R}$ be an α-exp-concave function, and D, G denote the diameter of $\mathcal{K}$ and a bound on the (sub)gradients of f respectively. The following holds for all $\gamma \leq \frac{1}{2}\min\{\frac{1}{GD}, \alpha\}$ and all $\mathbf{x}, \mathbf{y} \in \mathcal{K}$:*

$$f(\mathbf{x}) \geq f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{\gamma}{2} (\mathbf{x} - \mathbf{y})^\top \nabla f(\mathbf{y}) \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}).$$

Proof. The composition of a concave and non-decreasing function with another concave function is concave [9]. Therefore, since $2\gamma \leq \alpha$, the composition of $g(x) = x^{2\gamma/\alpha}$ with $f(\mathbf{x}) = \exp(-\alpha f(\mathbf{x}))$ is concave. It follows that the function $h(\mathbf{x}) \stackrel{\text{def}}{=} \exp(-2\gamma f(\mathbf{x}))$ is concave. Then by the concavity of $h(\mathbf{x})$,

$$h(\mathbf{x}) \leq h(\mathbf{y}) + \nabla h(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})$$

Plugging in $\nabla h(\mathbf{y}) = -2\gamma \exp(-2\gamma f(\mathbf{y}))\nabla f(\mathbf{y})$ gives

$$\exp(-2\gamma f(\mathbf{x})) \leq \exp(-2\gamma f(\mathbf{y}))[1 - 2\gamma \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})].$$

Simplifying gives

$$f(\mathbf{x}) \geq f(\mathbf{y}) - \frac{1}{2\gamma} \log \left(1 - 2\gamma \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})\right).$$

Next, note that $|2\gamma \nabla f(\mathbf{y})^\top (\mathbf{x}-\mathbf{y})| \leq 2\gamma GD \leq 1$ and that using the Taylor approximation, for $z \geq -1$, it holds that $-\log(1-z) \geq z + \frac{1}{4}z^2$. Applying the inequality for $z = 2\gamma \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})$ implies the lemma. □

4.3 Exponentially Weighted Online Convex Optimization

Before diving into efficient second order methods, we first describe a simple algorithm based on the multiplicative updates method which gives logarithmic regret for exp-concave losses. Algorithm (11) below, called EWOO, is a close relative to the Hedge Algorithm (1). Its regret guarantee is robust: it does not include a Lipschitz constant or a diameter bound. In addition, it is particularly simple to describe and analyze.

The downside of EWOO is its running time. A naive implementation would run in exponential time of the dimension. It is possible to given a randomized polynomial time implementation based on random sampling techniques, where the polynomial depends both on the dimension as well as the number of iterations, see bibliographic section for more details.

Algorithm 11 Exponentially Weighted Online Optimizer

1: Input: convex set $\mathcal{K}$, T, parameter $\alpha > 0$.
2: **for** $t = 1$ to T **do**
3: Let $w_t(\mathbf{x}) = e^{-\alpha \sum_{\tau=1}^{t-1} f_\tau(\mathbf{x})}$.
4: Play $\mathbf{x}_t$ given by

$$\mathbf{x}_t = \frac{\int_{\mathcal{K}} \mathbf{x}\, w_t(\mathbf{x}) d\mathbf{x}}{\int_{\mathcal{K}} w_t(\mathbf{x}) d\mathbf{x}}.$$

5: **end for**

In the analysis below, it can be observed that choosing $\mathbf{x}_t$ at random with density proportional to $w_t(\mathbf{x})$, instead of computing the entire integral, also guarantees our regret bounds on the expectation. This is the basis for

the polynomial time implementation. We proceed to give the logarithmic regret bounds.

Theorem 4.4.

$$\text{Regret}_T(EWOO) \;\leq\; \frac{d}{\alpha}\log T + \frac{2}{\alpha}.$$

Proof. Let $h_t(\mathbf{x}) = e^{-\alpha f_t(\mathbf{x})}$. Since f_t is α-exp-concave, we have that h_t is concave and thus

$$h_t(\mathbf{x}_t) \geq \frac{\int_{\mathcal{K}} h_t(\mathbf{x}) \prod_{\tau=1}^{t-1} h_\tau(\mathbf{x})\; d\mathbf{x}}{\int_{\mathcal{K}} \prod_{\tau=1}^{t-1} h_\tau(\mathbf{x})\; d\mathbf{x}}.$$

Hence, we have by telescoping product,

$$\prod_{\tau=1}^{t} h_\tau(\mathbf{x}_\tau) \geq \frac{\int_{\mathcal{K}} \prod_{\tau=1}^{t} h_\tau(\mathbf{x})\; d\mathbf{x}}{\int_{\mathcal{K}} 1\; d\mathbf{x}} = \frac{\int_{\mathcal{K}} \prod_{\tau=1}^{t} h_\tau(\mathbf{x})\; d\mathbf{x}}{\text{vol}(\mathcal{K})} \tag{4.1}$$

By definition of $\mathbf{x}^\star$ we have $\mathbf{x}^\star \in \arg\max_{\mathbf{x}\in\mathcal{K}} \prod_{t=1}^{T} h_t(\mathbf{x})$. Denote by $S_\delta \subset \mathcal{K}$ the translated Minkowski set given by

$$S_\delta = (1-\delta)\mathbf{x}^\star + \mathcal{K}_{1-\delta} = \left\{\mathbf{x} = (1-\delta)\mathbf{x}^\star + \delta\mathbf{y}\;,\; \mathbf{y}\in\mathcal{K}\right\}.$$

By concavity of h_t and the fact that h_t is non-negative, we have that,

$$\forall \mathbf{x} \in S_\delta\;. \quad h_t(\mathbf{x}) \geq (1-\delta)h_t(\mathbf{x}^\star).$$

Hence,

$$\forall \mathbf{x} \in S_\delta \quad \prod_{\tau=1}^{T} h_\tau(\mathbf{x}) \geq (1-\delta)^T \prod_{\tau=1}^{T} h_\tau(\mathbf{x}^\star)$$

Finally, since $S_\delta = (1-\delta)\mathbf{x}^\star + \delta\mathcal{K}$ is simply a rescaling of $\mathcal{K}$ followed by a translation, and we are in d dimensions, $\text{vol}(S_\delta) = \text{vol}(\mathcal{K}) \times \delta^d$. Putting this together with equation (4.1), we have

$$\prod_{\tau=1}^{T} h_\tau(\mathbf{x}_\tau) \geq \frac{\text{vol}(S_\delta)}{\text{vol}(\mathcal{K})}(1-\delta)^T \prod_{\tau=1}^{T} h_\tau(\mathbf{x}^\star) \geq \delta^d(1-\delta)^T \prod_{\tau=1}^{T} h_\tau(\mathbf{x}^\star).$$

We can now simplify by taking logarithms and changing sides,

$$\begin{aligned}
\text{Regret}_T(EWOO) &= \textstyle\sum_t f_t(\mathbf{x}_t) - f_t(\mathbf{x}^\star) \\
&= \tfrac{1}{\alpha}\log\frac{\prod_{\tau=1}^{T} h_\tau(\mathbf{x}^\star)}{\prod_{\tau=1}^{T} h_\tau(\mathbf{x}_\tau)} \\
&\leq \tfrac{1}{\alpha}\left(d\log\tfrac{1}{\delta} + T\log\tfrac{1}{1-\delta}\right) \leq \tfrac{d}{\alpha}\log T + \tfrac{2}{\alpha},
\end{aligned}$$

where the last step is by choosing $\delta = \frac{1}{T}$. □

4.4 The Online Newton Step Algorithm

Thus far we have only considered first order methods for regret minimization. In this section we consider a quasi-Newton approach, i.e., an online convex optimization algorithm that approximates the second derivative, or Hessian in more than one dimension. However, strictly speaking, the algorithm we analyze is also first order, in the sense that it only uses gradient information.

The algorithm we introduce and analyze, called online Newton step, is detailed in Algorithm 12. At each iteration, this algorithm chooses a vector that is the projection of the sum of the vector chosen at the previous iteration and an additional vector. Whereas for the online gradient descent algorithm this added vector was the gradient of the previous cost function, for online Newton step this vector is different: it is reminiscent to the direction in which the Newton-Raphson method would proceed if it were an offline optimization problem for the previous cost function. The Newton-Raphson algorithm would move in the direction of the vector which is the inverse Hessian times the gradient. In online Newton step, this direction is $A_t^{-1}\nabla_t$, where the matrix A_t is related to the Hessian as will be shown in the analysis.

Since adding a multiple of the Newton vector $A_t^{-1}\nabla_t$ to the current vector may result in a point outside the convex set, an additional projection step is required to obtain $\mathbf{x}_t$, the decision at time t. This projection is different than the standard Euclidean projection used by online gradient descent in Section 3.1. It is the projection according to the norm defined by the matrix A_t, rather than the Euclidean norm.

Algorithm 12 online Newton step

Input: convex set $\mathcal{K}$, T, $\mathbf{x}_1 \in \mathcal{K} \subseteq \mathbb{R}^n$, parameters $\gamma, \varepsilon > 0$, $A_0 = \varepsilon \mathbf{I}_n$
for $t = 1$ to T **do**
    Play $\mathbf{x}_t$ and observe cost $f_t(\mathbf{x}_t)$.
    Rank-1 update: $A_t = A_{t-1} + \nabla_t \nabla_t^\top$
    Newton step and generalized projection:

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \frac{1}{\gamma} A_t^{-1} \nabla_t$$

$$\mathbf{x}_{t+1} = \prod_{\mathcal{K}}^{A_t}(\mathbf{y}_{t+1}) = \arg\min_{\mathbf{x} \in \mathcal{K}} \left\{ \|\mathbf{y}_{t+1} - \mathbf{x}\|_{A_t}^2 \right\}$$

end for

The advantage of the online Newton step algorithm is its logarithmic re-

gret guarantee for exp-concave functions, as defined in the previous section. The following theorem bounds the regret of online Newton step.

Theorem 4.5. *Algorithm 12 with parameters* $\gamma = \frac{1}{2}\min\{\frac{1}{GD}, \alpha\}$, $\varepsilon = \frac{1}{\gamma^2 D^2}$ *and* $T \geq 4$ *guarantees*

$$\mathrm{Regret}_T \;\leq\; 2\left(\frac{1}{\alpha} + GD\right) n \log T.$$

As a first step, we prove the following lemma.

Lemma 4.6. *The regret of online Newton step is bounded by*

$$\mathrm{Regret}_T(\mathit{ONS}) \;\leq\; \left(\frac{1}{\alpha} + GD\right)\left(\sum_{t=1}^{T} \nabla_t^\top A_t^{-1} \nabla_t + 1\right)$$

Proof. Let $\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x})$ be the best decision in hindsight. By Lemma 4.3, we have for $\gamma = \frac{1}{2}\min\{\frac{1}{GD}, \alpha\}$,

$$f_t(\mathbf{x}_t) - f_t(\mathbf{x}^\star) \;\leq\; R_t,$$

where we define

$$R_t \stackrel{\text{def}}{=} \nabla_t^\top(\mathbf{x}_t - \mathbf{x}^\star) - \frac{\gamma}{2}(\mathbf{x}^\star - \mathbf{x}_t)^\top \nabla_t \nabla_t^\top (\mathbf{x}^\star - \mathbf{x}_t).$$

According to the update rule of the algorithm $\mathbf{x}_{t+1} = \Pi_{\mathcal{K}}^{A_t}(\mathbf{y}_{t+1})$. Now, by the definition of $\mathbf{y}_{t+1}$:

$$\mathbf{y}_{t+1} - \mathbf{x}^\star = \mathbf{x}_t - \mathbf{x}^\star - \frac{1}{\gamma} A_t^{-1} \nabla_t, \text{ and} \tag{4.2}$$

$$A_t(\mathbf{y}_{t+1} - \mathbf{x}^\star) = A_t(\mathbf{x}_t - \mathbf{x}^\star) - \frac{1}{\gamma}\nabla_t. \tag{4.3}$$

Multiplying the transpose of (4.2) by (4.3) we get

$$(\mathbf{y}_{t+1} - \mathbf{x}^\star)^\top A_t (\mathbf{y}_{t+1} - \mathbf{x}^\star) =$$
$$(\mathbf{x}_t - \mathbf{x}^\star)^\top A_t (\mathbf{x}_t - \mathbf{x}^\star) - \frac{2}{\gamma}\nabla_t^\top(\mathbf{x}_t - \mathbf{x}^\star) + \frac{1}{\gamma^2}\nabla_t^\top A_t^{-1}\nabla_t. \tag{4.4}$$

Since $\mathbf{x}_{t+1}$ is the projection of $\mathbf{y}_{t+1}$ in the norm induced by A_t, we have by the Pythagorean theorem (see §2.1.1)

$$\begin{aligned}(\mathbf{y}_{t+1} - \mathbf{x}^\star)^\top A_t (\mathbf{y}_{t+1} - \mathbf{x}^\star) &= \|\mathbf{y}_{t+1} - \mathbf{x}^\star\|_{A_t}^2 \\ &\geq \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|_{A_t}^2 \\ &= (\mathbf{x}_{t+1} - \mathbf{x}^\star)^\top A_t (\mathbf{x}_{t+1} - \mathbf{x}^\star).\end{aligned}$$

This inequality is the reason for using generalized projections as opposed to standard projections, which were used in the analysis of online gradient descent (see §3.1 Equation (3.2)). This fact together with (4.4) gives

$$\begin{aligned}\nabla_t^\top(\mathbf{x}_t-\mathbf{x}^\star) \leq\ & \frac{1}{2\gamma}\nabla_t^\top A_t^{-1}\nabla_t + \frac{\gamma}{2}(\mathbf{x}_t-\mathbf{x}^\star)^\top A_t(\mathbf{x}_t-\mathbf{x}^\star) \\ & - \frac{\gamma}{2}(\mathbf{x}_{t+1}-\mathbf{x}^\star)^\top A_t(\mathbf{x}_{t+1}-\mathbf{x}^\star).\end{aligned}$$

Now, summing up over $t=1$ to T we get that

$$\begin{aligned}\sum_{t=1}^T \nabla_t^\top(\mathbf{x}_t-\mathbf{x}^\star) &\leq \frac{1}{2\gamma}\sum_{t=1}^T \nabla_t^\top A_t^{-1}\nabla_t + \frac{\gamma}{2}(\mathbf{x}_1-\mathbf{x}^\star)^\top A_1(\mathbf{x}_1-\mathbf{x}^\star) \\ &\quad + \frac{\gamma}{2}\sum_{t=2}^T(\mathbf{x}_t-\mathbf{x}^\star)^\top(A_t-A_{t-1})(\mathbf{x}_t-\mathbf{x}^\star) \\ &\quad - \frac{\gamma}{2}(\mathbf{x}_{T+1}-\mathbf{x}^\star)^\top A_T(\mathbf{x}_{T+1}-\mathbf{x}^\star) \\ &\leq \frac{1}{2\gamma}\sum_{t=1}^T \nabla_t^\top A_t^{-1}\nabla_t + \frac{\gamma}{2}\sum_{t=1}^T(\mathbf{x}_t-\mathbf{x}^\star)^\top\nabla_t\nabla_t^\top(\mathbf{x}_t-\mathbf{x}^\star) \\ &\quad + \frac{\gamma}{2}(\mathbf{x}_1-\mathbf{x}^\star)^\top(A_1-\nabla_1\nabla_1^\top)(\mathbf{x}_1-\mathbf{x}^\star).\end{aligned}$$

In the last inequality we use the fact that $A_t - A_{t-1} = \nabla_t\nabla_t^\top$, and the fact that the matrix A_T is PSD and hence the last term before the inequality is negative. Thus,

$$\sum_{t=1}^T R_t \ \leq\ \frac{1}{2\gamma}\sum_{t=1}^T \nabla_t^\top A_t^{-1}\nabla_t + \frac{\gamma}{2}(\mathbf{x}_1-\mathbf{x}^\star)^\top(A_1-\nabla_1\nabla_1^\top)(\mathbf{x}_1-\mathbf{x}^\star).$$

Using the algorithm parameters $A_1 - \nabla_1\nabla_1^\top = \varepsilon\mathbf{I}_n$, $\varepsilon = \frac{1}{\gamma^2 D^2}$ and our notation for the diameter $\|\mathbf{x}_1-\mathbf{x}^\star\|^2 \leq D^2$ we have

$$\begin{aligned}\text{Regret}_T(\mathit{ONS}) \quad &\leq \quad \sum_{t=1}^T R_t \ \leq\ \frac{1}{2\gamma}\sum_{t=1}^T \nabla_t^\top A_t^{-1}\nabla_t + \frac{\gamma}{2}D^2\varepsilon \\ &\leq \quad \frac{1}{2\gamma}\sum_{t=1}^T \nabla_t^\top A_t^{-1}\nabla_t + \frac{1}{2\gamma}.\end{aligned}$$

Since $\gamma = \frac{1}{2}\min\{\frac{1}{GD},\alpha\}$, we have $\frac{1}{\gamma} \leq 2(\frac{1}{\alpha}+GD)$. This gives the lemma. □

We can now prove Theorem 4.5.

Proof of Theorem 4.5. First we show that the term $\sum_{t=1}^{T} \nabla_t^\top A_t^{-1} \nabla_t$ is upper bounded by a telescoping sum. Notice that

$$\nabla_t^\top A_t^{-1} \nabla_t = A_t^{-1} \bullet \nabla_t \nabla_t^\top = A_t^{-1} \bullet (A_t - A_{t-1})$$

where for matrices $A, B \in \mathbb{R}^{n \times n}$ we denote by $A \bullet B = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} B_{ij} = \mathbf{Tr}(AB^\top)$, which is equivalent to the inner product of these matrices as vectors in $\mathbb{R}^{n^2}$.

For real numbers $a, b \in \mathbb{R}_+$, the first order Taylor expansion of the logarithm of b at a implies $a^{-1}(a-b) \leq \log \frac{a}{b}$. An analogous fact holds for positive semidefinite matrices, i.e., $A^{-1} \bullet (A - B) \leq \log \frac{|A|}{|B|}$, where $|A|$ denotes the determinant of the matrix A (this is proved in Lemma 4.7). Using this fact we have

$$\begin{aligned}
\sum_{t=1}^{T} \nabla_t^\top A_t^{-1} \nabla_t &= \sum_{t=1}^{T} A_t^{-1} \bullet \nabla_t \nabla_t^\top \\
&= \sum_{t=1}^{T} A_t^{-1} \bullet (A_t - A_{t-1}) \\
&\leq \sum_{t=1}^{T} \log \frac{|A_t|}{|A_{t-1}|} = \log \frac{|A_T|}{|A_0|}.
\end{aligned}$$

Since $A_T = \sum_{t=1}^{T} \nabla_t \nabla_t^\top + \varepsilon I_n$ and $\|\nabla_t\| \leq G$, the largest eigenvalue of A_T is at most $TG^2 + \varepsilon$. Hence the determinant of A_T can be bounded by $|A_T| \leq (TG^2 + \varepsilon)^n$. Hence recalling that $\varepsilon = \frac{1}{\gamma^2 D^2}$ and $\gamma = \frac{1}{2} \min\{\frac{1}{GD}, \alpha\}$, for $T > 4$,

$$\sum_{t=1}^{T} \nabla_t^\top A_t^{-1} \nabla_t \leq \log \left(\tfrac{TG^2+\varepsilon}{\varepsilon} \right)^n \leq n \log(TG^2\gamma^2 D^2 + 1) \leq n \log T.$$

Plugging into Lemma 4.6 we obtain

$$\mathrm{Regret}_T(\mathrm{ONS}) \leq \left(\frac{1}{\alpha} + GD \right) (n \log T + 1),$$

which implies the theorem for $n > 1,\ T \geq 4$.

□

It remains to prove the technical lemma for positive semidefinite (PSD) matrices used above.

Lemma 4.7. *Let $A \succcurlyeq B \succ 0$ be positive definite matrices. Then*

$$A^{-1} \bullet (A - B) \;\leq\; \log \frac{|A|}{|B|}$$

Proof. For any positive definite matrix C, denote by $\lambda_1(C), \ldots, \lambda_n(C)$ its eigenvalues (which are positive).

$$\begin{aligned}
A^{-1} \bullet (A - B) \;&=\; \mathbf{Tr}(A^{-1}(A - B)) \\
&= \mathbf{Tr}(A^{-1/2}(A - B)A^{-1/2}) && \mathbf{Tr}(XY) = \mathbf{Tr}(YX) \\
&= \mathbf{Tr}(I - A^{-1/2}BA^{-1/2}) \\
&= \sum_{i=1}^{n} \left[1 - \lambda_i(A^{-1/2}BA^{-1/2})\right] && \mathbf{Tr}(C) = \sum_{i=1}^{n} \lambda_i(C) \\
&\leq -\sum_{i=1}^{n} \log \left[\lambda_i(A^{-1/2}BA^{-1/2})\right] && 1 - x \leq -\log(x) \\
&= -\log \left[\prod_{i=1}^{n} \lambda_i(A^{-1/2}BA^{-1/2})\right] \\
&= -\log |A^{-1/2}BA^{-1/2}| = \log \frac{|A|}{|B|} && |C| = \prod_{i=1}^{n} \lambda_i(C)
\end{aligned}$$

In the last equality we use the facts $|AB| = |A||B|$ and $|A^{-1}| = \frac{1}{|A|}$ for positive definite matrices (see exercises). □

Implementation and running time. The online Newton step algorithm requires $O(n^2)$ space to store the matrix A_t. Every iteration requires the computation of the matrix A_t^{-1}, the current gradient, a matrix-vector product, and possibly a projection onto the underlying convex set $\mathcal{K}$.

A naïve implementation would require computing the inverse of the matrix A_t on every iteration. However, in the case that A_t is invertible, the matrix inversion lemma (see bibliography) states that for invertible matrix A and vector $\mathbf{x}$,

$$(A + \mathbf{x}\mathbf{x}^\top)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{x}\mathbf{x}^\top A^{-1}}{1 + \mathbf{x}^\top A^{-1}\mathbf{x}}.$$

Thus, given A_{t-1}^{-1} and ∇_t one can compute A_t^{-1} in time $O(n^2)$ using only matrix-vector and vector-vector products.

The online Newton step algorithm also needs to make projections onto $\mathcal{K}$, but of a slightly different nature than online gradient descent and other online convex optimization algorithms. The required projection, denoted by $\Pi_{\mathcal{K}}^{A_t}$, is in the vector norm induced by the matrix A_t, viz. $\|\mathbf{x}\|_{A_t} = \sqrt{\mathbf{x}^\top A_t \mathbf{x}}$. It is equivalent to finding the point $\mathbf{x} \in \mathcal{K}$ which minimizes $(\mathbf{x}-\mathbf{y})^\top A_t (\mathbf{x}-\mathbf{y})$ where $\mathbf{y}$ is the point we are projecting. This is a convex program which can be solved up to any degree of accuracy in polynomial time.

Modulo the computation of generalized projections, the online Newton step algorithm can be implemented in time and space $O(n^2)$. In addition, the only information required is the gradient at each step (and the exp-concavity constant α of the loss functions).

4.5 Bibliographic Remarks

The Geometric Brownian Motion model for stock prices was suggested and studied as early as 1900 in the PhD thesis of Louis Bachelier [Bachelier, 1900], see also [Osborne, 1959], and used in the Nobel Prize winning work of Black and Scholes on options pricing [Black and Scholes, 1973]. In a strong deviation from standard financial theory, Thomas Cover [Cover, 1991] put forth the universal portfolio model, whose algorithmic theory we have historically sketched in chapter 1. The EWOO algorithm was essentially given in Cover's paper for the application of portfolio selection and logarithmic loss functions, and extended to exp-concave loss functions in [Hazan et al., 2006]. The randomized extension of Cover's algorithm that runs in polynomial running time is due to Kalai and Vempala [2003], and it naturally extends to EWOO.

Some bridges between classical portfolio theory and the universal model appear in [Abernethy et al., 2012]. Options pricing and its relation to regret minimization was recently also explored in the work of [DeMarzo et al., 2006].

Exp-concave functions have been considered in the context of prediction in [Kivinen and Warmuth, 1999], see also [Cesa-Bianchi and Lugosi, 2006] (chapter 3.3 and bibliography). A more general condition than exp-concavity called mixability was used by Vovk [1990] to give a general multiplicative update algorithm, see also [Foster et al., 2018]. For a thorough discussion of various conditions that allow logarithmic regret in online learning see [van Erven et al., 2015].

For the square-loss, [Azoury and Warmuth, 2001] gave a specially tailored and near-optimal prediction algorithm. Logarithmic regret algorithms for online convex optimization and the Online Newton Step algorithm were presented in [Hazan et al., 2007].

Logarithmic regret algorithms were used to derive $\tilde{O}(\frac{1}{\varepsilon})$-convergent algorithms for non-smooth convex optimization in the context of training support vector machines in [Shalev-Shwartz et al., 2011a]. Building upon these results, tight convergence rates of SGD for strongly convex and non-smooth functions were obtained in [Hazan and Kale, 2011].

The Sherman-Morrison formula, a.k.a. the matrix inversion lemma, gives the form of the inverse of a matrix after a rank-1 update, see [Riedel, 1991].

4.6 Exercises

1. For this question, assume all functions are twice differentiable. Prove that exp-concave functions are a larger class than strongly convex and Lipschitz functions. That is, prove that a strongly convex function over a bounded domain that is also G-Lipschitz is also exp-concave. Show that the converse is not necessarily true.

2. Prove that a twice-differentiable function f is α-exp-concave over $\mathcal{K}$ if and only if for all $\mathbf{x} \in \mathcal{K}$,

$$\nabla^2 f(\mathbf{x}) \succcurlyeq \alpha \nabla f(\mathbf{x}) \nabla f(\mathbf{x})^\top.$$

Hint: consider the Hessian of the function $e^{-\alpha f(\mathbf{x})}$, and use the fact that the Hessian of a convex function is always positive semidefinite.

3. Write pseudo-code for a portfolio selection algorithm based on online gradient descent. That is, given a set of return vectors, spell out the exact constants and updates based upon the gradients of the reward functions. Derive the regret bound based on Theorem 3.1. You may assume that the multiplicative change in price for any single asset is bounded, and use this quantity in your regret bound.
Do the same (pseudo-code and regret bound) for the Online Newton Step algorithm applied to portfolio selection.

Note: you are not required to give pseudo-code for projections onto the simplex.

4. Download stock prices from your favorite online finance website over a period of at least three years. Create a dataset for testing portfolio selection algorithms by creating price-return vectors. Implement the OGD and ONS algorithms and benchmark them on your data.

5. Prove that for positive definite matrices, $A, B \succ 0$, the following hold: $|AB| = |A||B|$ and $|A^{-1}| = \frac{1}{|A|}$, where $|A|$ denotes the determinant of A.

6. Let $h(x) : \mathbb{R} \mapsto \mathbb{R}$ be concave and non-decreasing, and let $g(\mathbf{x}) : \mathcal{K} \mapsto \mathbb{R}$ be concave. Prove that the function $f(\mathbf{x}) = h(g(\mathbf{x}))$ is concave.

Chapter 5

Regularization

In the previous chapters we have explored algorithms for OCO that are motivated by convex optimization. However, unlike convex optimization, the OCO framework optimizes the Regret performance metric. This distinction motivates a family of algorithms, called "Regularized Follow The Leader" (RFTL), which we introduce in this chapter.

In an OCO setting of regret minimization, the most straightforward approach for the online player is to use at any time the optimal decision (i.e., point in the convex set) in hindsight. Formally, let

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{K}} \sum_{\tau=1}^{t} f_\tau(\mathbf{x}).$$

This flavor of strategy is known as "fictitious play" in economics, and has been named "Follow the Leader" (FTL) in machine learning. It is not hard to see that this simple strategy fails miserably in a worst-case sense. That is, this strategy's regret can be linear in the number of iterations, as the following example shows: Consider $\mathcal{K} = [-1, 1]$, let $f_1(x) = \frac{1}{2}x$, and let f_τ for $\tau = 2, \ldots, T$ alternate between $-x$ or x. Thus,

$$\sum_{\tau=1}^{t} f_\tau(x) = \begin{cases} \frac{1}{2}x, & t \text{ is odd} \\ -\frac{1}{2}x, & \text{otherwise} \end{cases}$$

The FTL strategy will keep shifting between $x_t = -1$ and $x_t = 1$, always making the wrong choice.

The intuitive FTL strategy fails in the example above because it is unstable. Can we modify the FTL strategy such that it won't change decisions often, thereby causing it to attain low regret?

This question motivates the need for a general means of stabilizing the FTL method. Such a means is referred to as "regularization".

5.1 Regularization Functions

In this chapter we consider regularization functions, denoted $R : \mathcal{K} \mapsto \mathbb{R}$, which are strongly convex and smooth (recall definitions in §2.1).

Although it is not strictly necessary, we assume that the regularization functions in this chapter are twice differentiable over $\mathcal{K}$ and, for all points $\mathbf{x} \in \text{int}(\mathcal{K})$ in the interior of the decision set, have a Hessian $\nabla^2 R(\mathbf{x})$ that is, by the strong convexity of R, positive definite.

We denote the diameter of the set $\mathcal{K}$ relative to the function R as

$$D_R = \sqrt{\max_{\mathbf{x},\mathbf{y} \in \mathcal{K}} \{R(\mathbf{x}) - R(\mathbf{y})\}}.$$

Henceforth we make use of general norms and their dual. The dual norm to a norm $\|\cdot\|$ is given by the following definition:

$$\|\mathbf{y}\|^* \stackrel{\text{def}}{=} \sup_{\|\mathbf{x}\| \leq 1} \left\{ \mathbf{x}^\top \mathbf{y} \right\}.$$

A positive definite matrix A gives rise to the matrix norm $\|\mathbf{x}\|_A = \sqrt{\mathbf{x}^\top A \mathbf{x}}$. The dual norm of a matrix norm is $\|\mathbf{x}\|_A^* = \|\mathbf{x}\|_{A^{-1}}$.

The generalized Cauchy-Schwarz theorem asserts $\mathbf{x}^\top \mathbf{y} \leq \|\mathbf{x}\| \|\mathbf{y}\|^*$ and in particular for matrix norms, $\mathbf{x}^\top \mathbf{y} \leq \|\mathbf{x}\|_A \|\mathbf{y}\|_A^*$ (see exercises).

In our derivations, we usually consider matrix norms with respect to $\nabla^2 R(\mathbf{x})$, the Hessian of the regularization function $R(\mathbf{x})$, as well as the inverse Hessian denoted $\nabla^{-2} R(\mathbf{x})$. In such cases, we use the notation

$$\|\mathbf{x}\|_{\mathbf{y}} \stackrel{\text{def}}{=} \|\mathbf{x}\|_{\nabla^2 R(\mathbf{y})},$$

and similarly

$$\|\mathbf{x}\|_{\mathbf{y}}^* \stackrel{\text{def}}{=} \|\mathbf{x}\|_{\nabla^{-2} R(\mathbf{y})}.$$

A crucial quantity in the analysis of OCO algorithms that use regularization is the remainder term of the Taylor approximation of the regularization function, and especially the remainder term of the first order Taylor approximation. The difference between the value of the regularization function at $\mathbf{x}$ and the value of the first order Taylor approximation is known as the Bregman divergence, given by

Definition 5.1. *Denote by $B_R(\mathbf{x}||\mathbf{y})$ the Bregman divergence with respect to the function R, defined as*

$$B_R(\mathbf{x}||\mathbf{y}) = R(\mathbf{x}) - R(\mathbf{y}) - \nabla R(\mathbf{y})^\top(\mathbf{x}-\mathbf{y}).$$

For twice differentiable functions, Taylor expansion and the mean-value theorem assert that the Bregman divergence is equal to the second derivative at an intermediate point, i.e., (see exercises)

$$B_R(\mathbf{x}||\mathbf{y}) = \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_{\mathbf{z}}^2,$$

for some point $\mathbf{z} \in [\mathbf{x},\mathbf{y}]$, meaning there exists some $\alpha \in [0,1]$ such that $\mathbf{z} = \alpha\mathbf{x} + (1-\alpha)\mathbf{y}$. Therefore, the Bregman divergence defines a local norm, which has a dual norm. We shall denote this dual norm by

$$\|\cdot\|_{\mathbf{x},\mathbf{y}}^* \stackrel{\text{def}}{=} \|\cdot\|_{\mathbf{z}}^*.$$

With this notation we have

$$B_R(\mathbf{x}||\mathbf{y}) = \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_{\mathbf{x},\mathbf{y}}^2.$$

In online convex optimization, we commonly refer to the Bregman divergence between two consecutive decision points $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$. In such cases, we shorthand notation for the norm defined by the Bregman divergence with respect to R on the intermediate point in $[\mathbf{x}_t,\mathbf{x}_{t+1}]$ as $\|\cdot\|_t \stackrel{\text{def}}{=} \|\cdot\|_{\mathbf{x}_t,\mathbf{x}_{t+1}}$. The latter norm is called the local norm at iteration t. With this notation, we have $B_R(\mathbf{x}_t||\mathbf{x}_{t+1}) = \frac{1}{2}\|\mathbf{x}_t - \mathbf{x}_{t+1}\|_t^2$.

Finally, we consider below generalized projections that use the Bregman divergence as a distance instead of a norm. Formally, the projection of a point $\mathbf{y}$ according to the Bregman divergence with respect to function R is given by

$$\arg\min_{\mathbf{x}\in\mathcal{K}} B_R(\mathbf{x}||\mathbf{y}).$$

5.2 The RFTL Algorithm and its Analysis

Recall the caveat with straightforward use of follow-the-leader: as in the bad example we have considered, the predictions of FTL may vary wildly from one iteration to the next. This motivates the modification of the basic FTL strategy in order to stabilize the prediction. By adding a regularization term, we obtain the RFTL (Regularized Follow the Leader) algorithm.

We proceed to formally describe the RFTL algorithmic template and analyze it. The analysis gives asymptotically optimal regret bounds. However, we do not optimize the constants in the regret bounds in order to improve clarity of presentation.

Throughout this chapter, recall the notation of ∇_t to denote the gradient of the current cost function at the current point, i.e.,

$$\nabla_t \stackrel{\text{def}}{=} \nabla f_t(\mathbf{x}_t).$$

In the OCO setting, the regret of convex cost functions can be bounded by a linear function via the inequality $f_t(\mathbf{x}_t) - f_t(\mathbf{x}^\star) \leq \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star)$. Thus, the overall regret (recall definition (1.1)) of an OCO algorithm can be bounded by

$$\sum_t f_t(\mathbf{x}_t) - f_t(\mathbf{x}^\star) \leq \sum_t \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star). \tag{5.1}$$

5.2.1 Meta-algorithm definition

The generic RFTL meta-algorithm is defined in Algorithm 13. The regularization function R is assumed to be strongly convex, smooth, and twice differentiable.

Algorithm 13 Regularized Follow The Leader

1: Input: $\eta > 0$, regularization function R, and a bounded, convex and closed set $\mathcal{K}$.
2: Let $\mathbf{x}_1 = \arg\min_{\mathbf{x}\in\mathcal{K}} \{R(\mathbf{x})\}$.
3: **for** $t = 1$ to T **do**
4: Play $\mathbf{x}_t$ and observe cost $f_t(\mathbf{x}_t)$.
5: Update

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \eta \sum_{s=1}^{t} \nabla_s^\top \mathbf{x} + R(\mathbf{x}) \right\}$$

6: **end for**

5.2.2 The regret bound

Theorem 5.2. *The RFTL Algorithm 13 attains for every $\mathbf{u} \in \mathcal{K}$ the following bound on the regret:*

$$\text{Regret}_T \leq 2\eta \sum_{t=1}^{T} \|\nabla_t\|_t^{*2} + \frac{R(\mathbf{u}) - R(\mathbf{x}_1)}{\eta}.$$

If an upper bound on the local norms is known, i.e., $\|\nabla_t\|_t^* \leq G_R$ for all times t, then we can further optimize over the choice of η to obtain

$$\text{Regret}_T \leq 2D_R G_R \sqrt{2T}.$$

To prove Theorem 5.2, we first relate the regret to the "stability" in prediction. This is formally captured by the following lemma[10].

Lemma 5.3. *Algorithm 13 guarantees the following regret bound*

$$\text{Regret}_T \leq \sum_{t=1}^{T} \nabla_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{1}{\eta} D_R^2$$

Proof. For convenience of the derivations, define the functions

$$g_0(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{\eta} R(\mathbf{x}) \ , \ g_t(\mathbf{x}) \stackrel{\text{def}}{=} \nabla_t^\top \mathbf{x}.$$

By equation (5.1), it suffices to bound $\sum_{t=1}^{T} [g_t(\mathbf{x}_t) - g_t(\mathbf{u})]$. As a first step, we prove the following inequality:

Lemma 5.4. *For every* $\mathbf{u} \in \mathcal{K}$*,*

$$\sum_{t=0}^{T} g_t(\mathbf{u}) \geq \sum_{t=0}^{T} g_t(\mathbf{x}_{t+1}).$$

Proof. by induction on T:
Induction base:
By definition, we have that $\mathbf{x}_1 = \arg\min_{\mathbf{x} \in \mathcal{K}} \{R(\mathbf{x})\}$, and thus $g_0(\mathbf{u}) \geq g_0(\mathbf{x}_1)$ for all $\mathbf{u}$.
Induction step:
Assume that for T, we have

$$\sum_{t=0}^{T} g_t(\mathbf{u}) \geq \sum_{t=0}^{T} g_t(\mathbf{x}_{t+1})$$

and let us prove the statement for $T+1$. Since $\mathbf{x}_{T+2} = \arg\min_{\mathbf{x} \in \mathcal{K}} \{\sum_{t=0}^{T+1} g_t(\mathbf{x})\}$

we have:

$$\begin{aligned}\sum_{t=0}^{T+1} g_t(\mathbf{u}) &\geq \sum_{t=0}^{T+1} g_t(\mathbf{x}_{T+2}) \\ &= \sum_{t=0}^{T} g_t(\mathbf{x}_{T+2}) + g_{T+1}(\mathbf{x}_{T+2}) \\ &\geq \sum_{t=0}^{T} g_t(\mathbf{x}_{t+1}) + g_{T+1}(\mathbf{x}_{T+2}) \\ &= \sum_{t=0}^{T+1} g_t(\mathbf{x}_{t+1}).\end{aligned}$$

Where in the third line we used the induction hypothesis for $\mathbf{u} = \mathbf{x}_{T+2}$. □

We conclude that

$$\begin{aligned}\sum_{t=1}^{T}[g_t(\mathbf{x}_t) - g_t(\mathbf{u})] &\leq \sum_{t=1}^{T}[g_t(\mathbf{x}_t) - g_t(\mathbf{x}_{t+1})] + [g_0(\mathbf{u}) - g_0(\mathbf{x}_1)] \\ &= \sum_{t=1}^{T} g_t(\mathbf{x}_t) - g_t(\mathbf{x}_{t+1}) + \frac{1}{\eta}\left[R(\mathbf{u}) - R(\mathbf{x}_1)\right] \\ &\leq \sum_{t=1}^{T} g_t(\mathbf{x}_t) - g_t(\mathbf{x}_{t+1}) + \frac{1}{\eta} D_R^2.\end{aligned}$$

□

Proof of Theorem 5.2. Recall that $R(\mathbf{x})$ is a convex function and $\mathcal{K}$ is a convex set. Denote:

$$\Phi_t(\mathbf{x}) \stackrel{\text{def}}{=} \eta \sum_{s=1}^{t} \nabla_s^\top \mathbf{x} + R(\mathbf{x}).$$

By the Taylor expansion (with its explicit remainder term via the mean-value theorem) at $\mathbf{x}_{t+1}$, and by the definition of the Bregman divergence,

$$\begin{aligned}\Phi_t(\mathbf{x}_t) &= \Phi_t(\mathbf{x}_{t+1}) + (\mathbf{x}_t - \mathbf{x}_{t+1})^\top \nabla\Phi_t(\mathbf{x}_{t+1}) + B_{\Phi_t}(\mathbf{x}_t||\mathbf{x}_{t+1}) \\ &\geq \Phi_t(\mathbf{x}_{t+1}) + B_{\Phi_t}(\mathbf{x}_t||\mathbf{x}_{t+1}) \\ &= \Phi_t(\mathbf{x}_{t+1}) + B_R(\mathbf{x}_t||\mathbf{x}_{t+1}).\end{aligned}$$

The inequality holds since $\mathbf{x}_{t+1}$ is a minimum of Φ_t over $\mathcal{K}$, as in Theorem 2.2. The last equality holds since the component $\nabla_s^\top \mathbf{x}$ is linear and thus does not affect the Bregman divergence. Thus,

$$\begin{aligned} B_R(\mathbf{x}_t||\mathbf{x}_{t+1}) &\leq \Phi_t(\mathbf{x}_t) - \Phi_t(\mathbf{x}_{t+1}) \\ &= (\Phi_{t-1}(\mathbf{x}_t) - \Phi_{t-1}(\mathbf{x}_{t+1})) + \eta \nabla_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \\ &\leq \eta \, \nabla_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \quad (\mathbf{x}_t \text{ is the minimizer}) \end{aligned} \tag{5.2}$$

To proceed, recall the shorthand for the norm induced by the Bregman divergence with respect to R on point $\mathbf{x}_t, \mathbf{x}_{t+1}$ as $\|\cdot\|_t = \|\cdot\|_{\mathbf{x}_t, \mathbf{x}_{t+1}}$. Similarly for the dual local norm $\|\cdot\|_t^* = \|\cdot\|_{\mathbf{x}_t, \mathbf{x}_{t+1}}^*$. With this notation, we have $B_R(\mathbf{x}_t||\mathbf{x}_{t+1}) = \frac{1}{2}\|\mathbf{x}_t - \mathbf{x}_{t+1}\|_t^2$. By the generalized Cauchy-Schwarz inequality,

$$\begin{aligned} \nabla_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) &\leq \|\nabla_t\|_t^* \cdot \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_t && \text{Cauchy-Schwarz} \\ &= \|\nabla_t\|_t^* \cdot \sqrt{2 B_R(\mathbf{x}_t||\mathbf{x}_{t+1})} \\ &\leq \|\nabla_t\|_t^* \cdot \sqrt{2\,\eta\, \nabla_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1})}. && (5.2) \end{aligned}$$

After rearranging we get

$$\nabla_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \leq 2\,\eta\, \|\nabla_t\|_t^{*2}.$$

Combining this inequality with Lemma 5.3 we obtain the theorem statement. □

5.3 Online Mirror Descent

In the convex optimization literature, "Mirror Descent" refers to a general class of first order methods generalizing gradient descent. Online Mirror descent (OMD) is the online counterpart of this class of methods. This relationship is analogous to the relationship of online gradient descent to traditional (offline) gradient descent.

OMD is an iterative algorithm that computes the current decision using a simple gradient update rule and the previous decision, much like OGD. The generality of the method stems from the update being carried out in a "dual" space, where the duality notion is defined by the choice of regularization: the gradient of the regularization function defines a mapping from $\mathbb{R}^n$ onto itself, which is a vector field. The gradient updates are then carried out in this vector field.

For the RFTL algorithm the intuition was straightforward—the regularization was used to ensure stability of the decision. For OMD, regularization has an additional purpose: regularization transforms the space in which gradient updates are performed. This transformation enables better bounds in terms of the geometry of the space.

The OMD algorithm comes in two flavors: an agile and a lazy version. The lazy version keeps track of a point in Euclidean space and projects onto the convex decision set $\mathcal{K}$ only at decision time. In contrast, the agile version maintains a feasible point at all times, much like OGD.

Algorithm 14 Online Mirror Descent

Input: parameter $\eta > 0$, regularization function $R(\mathbf{x})$.
Let $\mathbf{y}_1$ be such that $\nabla R(\mathbf{y}_1) = \mathbf{0}$ and $\mathbf{x}_1 = \arg\min_{\mathbf{x}\in\mathcal{K}} B_R(\mathbf{x}||\mathbf{y}_1)$.
for $t = 1$ to T **do**
    Play $\mathbf{x}_t$.
    Observe the loss function f_t and let $\nabla_t = \nabla f_t(\mathbf{x}_t)$.
    Update $\mathbf{y}_t$ according to the rule:

$$\begin{aligned} &\text{[Lazy version]} && \nabla R(\mathbf{y}_{t+1}) = \nabla R(\mathbf{y}_t) - \eta\,\nabla_t \\ &\text{[Agile version]} && \nabla R(\mathbf{y}_{t+1}) = \nabla R(\mathbf{x}_t) - \eta\,\nabla_t \end{aligned}$$

    Project according to B_R:

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{K}} B_R(\mathbf{x}||\mathbf{y}_{t+1})$$

end for

Both versions can be analyzed to give roughly the same regret bounds as the RFTL algorithm. In light of what we will see next, this is not surprising: for linear cost functions, the RFTL and lazy-OMD algorithms are equivalent!

Thus, we get regret bounds for free for the lazy version. The agile version can be shown to attain similar regret bounds, and is in fact superior in certain settings that require adaptivity. This issue is further explored in chapter 10. The analysis of the agile version is of independent interest and we give it below.

5.3.1 Equivalence of lazy OMD and RFTL

The OMD (lazy version) and RFTL are identical for linear cost functions, as we show next.

Lemma 5.5. *Let $f_1, ..., f_T$ be linear cost functions. The lazy OMD and RFTL algorithms produce identical predictions, i.e.,*

$$\arg\min_{\mathbf{x}\in\mathcal{K}} \{B_R(\mathbf{x}||\mathbf{y}_t)\} = \arg\min_{\mathbf{x}\in\mathcal{K}} \left(\eta \sum_{s=1}^{t-1} \nabla_s^\top \mathbf{x} + R(\mathbf{x}) \right).$$

Proof. First, observe that the unconstrained minimum

$$\mathbf{x}_t^\star \stackrel{\text{def}}{=} \arg\min_{\mathbf{x}\in\mathbb{R}^n} \left\{ \sum_{s=1}^{t-1} \nabla_s^\top \mathbf{x} + \frac{1}{\eta} R(\mathbf{x}) \right\}$$

satisfies

$$\nabla R(\mathbf{x}_t^\star) = -\eta \sum_{s=1}^{t-1} \nabla_s.$$

By definition, $\mathbf{y}_t$ also satisfies the above equation, but since $R(\mathbf{x})$ is strictly convex, there is only one solution for the above equation and thus $\mathbf{y}_t = \mathbf{x}_t^\star$. Hence,

$$\begin{aligned} B_R(\mathbf{x}||\mathbf{y}_t) &= R(\mathbf{x}) - R(\mathbf{y}_t) - (\nabla R(\mathbf{y}_t))^\top (\mathbf{x} - \mathbf{y}_t) \\ &= R(\mathbf{x}) - R(\mathbf{y}_t) + \eta \sum_{s=1}^{t-1} \nabla_s^\top (\mathbf{x} - \mathbf{y}_t) \ . \end{aligned}$$

Since $R(\mathbf{y}_t)$ and $\sum_{s=1}^{t-1} \nabla_s^\top \mathbf{y}_t$ are independent of $\mathbf{x}$, it follows that $B_R(\mathbf{x}||\mathbf{y}_t)$ is minimized at the point $\mathbf{x}$ that minimizes $R(\mathbf{x}) + \eta \sum_{s=1}^{t-1} \nabla_s^\top \mathbf{x}$ over $\mathcal{K}$ which, in turn, implies that

$$\arg\min_{\mathbf{x}\in\mathcal{K}} B_R(\mathbf{x}||\mathbf{y}_t) = \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \sum_{s=1}^{t-1} \nabla_s^\top \mathbf{x} + \frac{1}{\eta} R(\mathbf{x}) \right\} .$$

□

5.3.2 Regret bounds for Mirror Descent

In this subsection we prove regret bounds for the agile version of the RFTL algorithm. The analysis is quite different than the one for the lazy version, and of independent interest.

Theorem 5.6. *The OMD Algorithm 14 attains for every $\mathbf{u} \in \mathcal{K}$ the following bound on the regret:*

$$\text{Regret}_T \leq \frac{\eta}{4} \sum_{t=1}^{T} \|\nabla_t\|_t^{*2} + \frac{R(\mathbf{u}) - R(\mathbf{x}_1)}{2\eta}.$$

If an upper bound on the local norms is known, i.e., $\|\nabla_t\|_t^* \leq G_R$ for all times t, then we can further optimize over the choice of η to obtain

$$\text{Regret}_T \leq D_R G_R \sqrt{T}.$$

Proof. Since the functions $\mathbf{f}_t$ are convex, for any $\mathbf{x}^* \in K$,

$$\mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{x}^*) \leq \nabla \mathbf{f}_t(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^*).$$

The following property of Bregman divergences follows from the definition: for any vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$,

$$(\mathbf{x} - \mathbf{y})^\top (\nabla \mathcal{R}(\mathbf{z}) - \nabla \mathcal{R}(\mathbf{y})) = B_\mathcal{R}(\mathbf{x}, \mathbf{y}) - B_\mathcal{R}(\mathbf{x}, \mathbf{z}) + B_\mathcal{R}(\mathbf{y}, \mathbf{z}).$$

Combining both observations,

$$\begin{aligned}
\mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{x}^*) &\leq \nabla \mathbf{f}_t(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^*) \\
&= \frac{1}{\eta} (\nabla \mathcal{R}(\mathbf{y}_{t+1}) - \nabla \mathcal{R}(\mathbf{x}_t))^\top (\mathbf{x}^* - \mathbf{x}_t) \\
&= \frac{1}{\eta} [B_\mathcal{R}(\mathbf{x}^*, \mathbf{x}_t) - B_\mathcal{R}(\mathbf{x}^*, \mathbf{y}_{t+1}) + B_\mathcal{R}(\mathbf{x}_t, \mathbf{y}_{t+1})] \\
&\leq \frac{1}{\eta} [B_\mathcal{R}(\mathbf{x}^*, \mathbf{x}_t) - B_\mathcal{R}(\mathbf{x}^*, \mathbf{x}_{t+1}) + B_\mathcal{R}(\mathbf{x}_t, \mathbf{y}_{t+1})]
\end{aligned}$$

where the last inequality follows from the generalized Pythagorean theorem, as $\mathbf{x}_{t+1}$ is the projection w.r.t the Bregman divergence of $\mathbf{y}_{t+1}$ and $\mathbf{x}^* \in K$ is in the convex set. Summing over all iterations,

$$\begin{aligned}
\text{Regret} &\leq \frac{1}{\eta} [B_\mathcal{R}(\mathbf{x}^*, \mathbf{x}_1) - B_\mathcal{R}(\mathbf{x}^*, \mathbf{x}_T)] + \sum_{t=1}^{T} \frac{1}{\eta} B_\mathcal{R}(\mathbf{x}_t, \mathbf{y}_{t+1}) \\
&\leq \frac{1}{\eta} D_R^2 + \sum_{t=1}^{T} \frac{1}{\eta} B_\mathcal{R}(\mathbf{x}_t, \mathbf{y}_{t+1}) \qquad (5.3)
\end{aligned}$$

We proceed to bound $B_\mathcal{R}(\mathbf{x}_t, \mathbf{y}_{t+1})$. By definition of Bregman divergence, and the generalized Cauchy-Schwartz inequality,

$$\begin{aligned}
B_\mathcal{R}(\mathbf{x}_t, \mathbf{y}_{t+1}) + B_\mathcal{R}(\mathbf{y}_{t+1}, \mathbf{x}_t) &= (\nabla \mathcal{R}(\mathbf{x}_t) - \nabla \mathcal{R}(\mathbf{y}_{t+1}))^\top (\mathbf{x}_t - \mathbf{y}_{t+1}) \\
&= \eta \nabla \mathbf{f}_t(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{y}_{t+1}) \\
&\leq \eta \|\nabla \mathbf{f}_t(\mathbf{x}_t)\|_t^* \|\mathbf{x}_t - \mathbf{y}_{t+1}\|_t \\
&\leq \frac{1}{2} \eta^2 G_R^2 + \frac{1}{2} \|\mathbf{x}_t - \mathbf{y}_{t+1}\|_t^2.
\end{aligned}$$

where in the last inequality follows from $(a-b)^2 \geq 0$. Thus, we have

$$B_{\mathcal{R}}(\mathbf{x}_t, \mathbf{y}_{t+1}) \leq \frac{1}{2}\eta^2 G_R^2 + \frac{1}{2}\|\mathbf{x}_t - \mathbf{y}_{t+1}\|_t^2 - B_{\mathcal{R}}(\mathbf{y}_{t+1}, \mathbf{x}_t) = \frac{1}{2}\eta^2 G_R^2.$$

Plugging back into Equation (5.3), and by non-negativity of the Bregman divergence, we get

$$\text{Regret} \leq \frac{1}{2}[\frac{1}{\eta}D_R^2 + \frac{1}{2}\eta T G_R^2] \leq D_R G_R \sqrt{T} \ ,$$

by taking $\eta = \frac{D_R}{\sqrt{T}G_R}$

□

5.4 Application and Special Cases

In this section we illustrate the generality of the regularization technique: we show how to derive the two most important and famous online algorithms—the online gradient descent algorithm and the online exponentiated gradient (based on the multiplicative update method)—from the RFTL meta-algorithm.

Other important special cases of the RFTL meta-algorithm are derived with matrix-norm regularization—namely, the von Neumann entropy function, and the log-determinant function, as well as self-concordant barrier regularization—which we shall explore in detail in the next chapter.

5.4.1 Deriving online gradient descent

To derive the online gradient descent algorithm, we take $R(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|_2^2$ for an arbitrary $\mathbf{x}_0 \in \mathcal{K}$. Projection with respect to this divergence is the standard Euclidean projection (see exercises), and in addition, $\nabla R(\mathbf{x}) = \mathbf{x} - \mathbf{x}_0$. Hence, the update rule for the OMD Algorithm 14 becomes:

$$\mathbf{x}_t = \prod_{\mathcal{K}}(\mathbf{y}_t), \ \mathbf{y}_t = \mathbf{y}_{t-1} - \eta\nabla_{t-1} \qquad \text{lazy version}$$

$$\mathbf{x}_t = \prod_{\mathcal{K}}(\mathbf{y}_t), \ \mathbf{y}_t = \mathbf{x}_{t-1} - \eta\nabla_{t-1} \qquad \text{agile version}$$

The latter algorithm is exactly online gradient descent, as described in Algorithm 8 in chapter 3. However, both variants behave very differently, as explored in chapter 10 (see also exercises).

Theorem 5.2 gives us the following bound on the regret (where $D_R, \|\cdot\|_t$ are the diameter and local norm defined with respect to the regularizer R

as defined in the beginning of this chapter, and D is the Euclidean diameter as defined in chapter 2)

$$\text{Regret}_T \leq \frac{1}{\eta} D_R^2 + 2\eta \sum_t \|\nabla_t\|_t^{*2} \leq \frac{1}{2\eta} D^2 + 2\eta \sum_t \|\nabla_t\|^2 \leq 2GD\sqrt{T},$$

where the second inequality follows since for $R(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|^2$, the local norm $\|\cdot\|_t$ reduces to the Euclidean norm.

5.4.2 Deriving multiplicative updates

Let $R(\mathbf{x}) = \mathbf{x} \log \mathbf{x} = \sum_i \mathbf{x}_i \log \mathbf{x}_i$ be the negative entropy function, where $\log \mathbf{x}$ is to be interpreted element-wise. Then $\nabla R(\mathbf{x}) = \mathbf{1} + \log \mathbf{x}$, and hence the update rules for the OMD algorithm become:

$$\mathbf{x}_t = \arg\min_{\mathbf{x} \in \mathcal{K}} B_R(\mathbf{x} || \mathbf{y}_t), \ \log \mathbf{y}_t = \log \mathbf{y}_{t-1} - \eta \nabla_{t-1} \qquad \text{lazy version}$$

$$\mathbf{x}_t = \arg\min_{\mathbf{x} \in \mathcal{K}} B_R(\mathbf{x} || \mathbf{y}_t), \ \log \mathbf{y}_t = \log \mathbf{x}_{t-1} - \eta \nabla_{t-1} \qquad \text{agile version}$$

With this choice of regularizer, a notable special case is the experts problem we encountered in §1.3, for which the decision set $\mathcal{K}$ is the n-dimensional simplex $\Delta_n = \{\mathbf{x} \in \mathbb{R}_+^n \mid \sum_i \mathbf{x}_i = 1\}$. In this special case, the projection according to the negative entropy becomes scaling by the ℓ_1 norm (see exercises), which implies that both update rules amount to the same algorithm:

$$\mathbf{x}_{t+1}(i) = \frac{\mathbf{x}_t(i) \cdot e^{-\eta \nabla_t(i)}}{\sum_{j=1}^n \mathbf{x}_t(j) \cdot e^{-\eta \nabla_t(j)}},$$

which is exactly the Hedge algorithm from the first chapter!

Theorem 5.6 gives us the following bound on the regret:

$$\text{Regret}_T \leq 2\sqrt{2D_R^2 \sum_t \|\nabla_t\|_t^{*2}}.$$

If the costs per individual expert are in the range $[0, 1]$, it can be shown that

$$\|\nabla_t\|_t^* \leq \|\nabla_t\|_\infty \leq 1 = G_R.$$

In addition, when R is the negative entropy function, the diameter over the simplex can be shown to be bounded by $D_R^2 \leq \log n$ (see exercises), giving rise to the bound

$$\text{Regret}_T \leq 2D_R G_R \sqrt{2T} \leq 2\sqrt{2T \log n}.$$

For an arbitrary range of costs, we obtain the exponentiated gradient algorithm described in Algorithm 15.

Algorithm 15 Exponentiated Gradient

Input: parameter $\eta > 0$.
Let $\mathbf{y}_1 = \mathbf{1}$, $\mathbf{x}_1 = \frac{\mathbf{y}_1}{\|\mathbf{y}_1\|_1}$.
for $t = 1$ to T **do**
    Predict $\mathbf{x}_t$.
    Observe f_t, update $\mathbf{y}_{t+1}(i) = \mathbf{y}_t(i)e^{-\eta \nabla_t(i)}$ for all $i \in [n]$.
    Project: $\mathbf{x}_{t+1} = \frac{\mathbf{y}_{t+1}}{\|\mathbf{y}_{t+1}\|_1}$
end for

The regret achieved by the exponentiated gradient algorithm can be bounded using the following corollary of Theorem 5.2:

Corollary 5.7. *The exponentiated gradient algorithm with gradients bounded by* $\|\nabla_t\|_\infty \leq G_\infty$ *and parameter* $\eta = \sqrt{\frac{\log n}{2TG_\infty^2}}$ *has regret bounded by*

$$\text{Regret}_T \leq 2G_\infty \sqrt{2T \log n}.$$

5.5 Randomized Regularization

The connection between stability in decision making and low regret has motivated our discussion of regularization thus far. However, this stability need not be achieved only using strongly convex regularization functions. An alternative method to achieve stability in decisions is by introducing randomization into the algorithm. In fact, historically, this method preceded methods based on strongly convex regularization (see bibliography).

In this section we first describe a deterministic algorithm for online convex optimization that is easily amenable to speedup via randomization. We then give an efficient randomized algorithm for the special case of OCO with linear losses.

Oblivious vs. adaptive adversaries. For simplicity, we consider ourselves in this section with a slightly restricted version of OCO. So far, we have not restricted the cost functions in any way, and they could depend on the choice of decision by the online learner. However, when dealing with randomized algorithms, this issue becomes a bit more subtle: can the cost functions depend on the randomness of the decision making algorithm

itself? Furthermore, when analyzing the regret, which is now a random variable, dependencies across different iterations require probabilistic machinery which adds little to the fundamental understanding of randomized OCO algorithms. To avoid these complications, we make the following assumption throughout this section: the cost functions $\{\mathbf{f}_t\}$ are adversarially chosen ahead of time, and do not depend on the actual decisions of the online learner. This version of OCO is called the *oblivious* setting, to distinguish it from the *adaptive* setting.

5.5.1 Perturbation for convex losses

The prediction in Algorithm 16 is according to a version of the follow-the-leader algorithm, augmented with an additional component of randomization. It is a deterministic algorithm that computes the expected decision according to a random variable. The random variable is the minimizer over the decision set according to the sum of gradients of the cost functions and an additional random vector.

In practice, the expectation need not be computed exactly. Estimation (via random sampling) up to a precision that depends linearly on the number of iterations would suffice.

The algorithm accepts as input a distribution, with the probability density function (PDF) denoted $\mathcal{D}$, over vectors in n-dimensional Euclidean space $\mathbf{n} \in \mathbb{R}^n$. For $\sigma, L \in \mathbb{R}$, we say that a distribution $\mathcal{D}$ is $(\sigma, L) = (\sigma_a, L_a)$ stable with respect to the norm $\|\cdot\|_a$ if

$$\mathop{\mathbf{E}}_{\mathbf{n}\sim\mathcal{D}}[\|\mathbf{n}\|_a^*] = \sigma_a,$$

and

$$\forall \mathbf{u},\ \int_{\mathbf{n}} |\mathcal{D}(\mathbf{n}) - \mathcal{D}(\mathbf{n}-\mathbf{u})|\, d\mathbf{n} \le L_a \|\mathbf{u}\|_a^*.$$

Here $\mathbf{n} \sim \mathcal{D}$ denotes a vector $\mathbf{n} \in \mathbb{R}^n$ sampled according to distribution $\mathcal{D}$, and $\mathcal{D}(\mathbf{n})$ is the value of the probability density function $\mathcal{D}$ over $\mathbf{n}$. The subscript a is omitted if clear from the context.

The first parameter, σ, is related to the variance of $\mathcal{D}$, while the second, L, is a measure of the sensitivity of the distribution[11]. For example, if $\mathcal{D}$ is the uniform distribution over the hypercube $[0,1]^n$, then it holds that (see exercises) for the Euclidean norm

$$\sigma_2 \le \sqrt{n}\ ,\ L_2 \le 1.$$

Reusing notation from previous chapters, denote by $D = D_a$ the diameter of the set $\mathcal{K}$ according to the norm $\|\cdot\|_a$, and by $D^* = D_a^*$ the diameter

according to its dual norm. Similarly, denote by $G = G_a$ and $G^* = G_a^*$ an upper bound on the norm (and dual norm) of the gradients.

Algorithm 16 Follow-the-perturbed-leader for convex losses

1: Input: $\eta > 0$, distribution $\mathcal{D}$ over $\mathbb{R}^n$, decision set $\mathcal{K} \subseteq \mathbb{R}^n$.
2: Let $\mathbf{x}_1 = \mathbf{E}_{\mathbf{n}\sim\mathcal{D}}\left[\arg\min_{\mathbf{x}\in\mathcal{K}}\left\{\mathbf{n}^\top\mathbf{x}\right\}\right]$.
3: **for** $t = 1$ to T **do**
4: Predict $\mathbf{x}_t$.
5: Observe the loss function f_t, suffer loss $f_t(\mathbf{x}_t)$ and let $\nabla_t = \nabla f_t(\mathbf{x}_t)$.
6: Update

$$\mathbf{x}_{t+1} = \mathop{\mathbf{E}}_{\mathbf{n}\sim\mathcal{D}}\left[\arg\min_{\mathbf{x}\in\mathcal{K}}\left\{\eta\sum_{s=1}^{t}\nabla_s^\top\mathbf{x} + \mathbf{n}^\top\mathbf{x}\right\}\right] \tag{5.4}$$

7: **end for**

Theorem 5.8. *Let the distribution $\mathcal{D}$ be (σ, L)-stable with respect to norm $\|\cdot\|_a$. The FPL algorithm attains the following bound on the regret:*

$$\mathrm{Regret}_T \le \eta D G^{*2} L T + \frac{1}{\eta}\sigma D.$$

We can further optimize over the choice of η to obtain

$$\mathrm{Regret}_T \le 2LDG^*\sqrt{\sigma T}.$$

Proof. Define the random function g_0 as

$$g_0(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{\eta}\mathbf{n}^\top\mathbf{x}.$$

It follows from Lemma 5.4 applied to the functions $\{g_t(\mathbf{x}) = \nabla_t^\top\mathbf{x}\}$ that

$$\mathbf{E}\left[\sum_{t=0}^{T} g_t(\mathbf{u})\right] \ge \mathbf{E}\left[\sum_{t=0}^{T} g_t(\mathbf{x}_{t+1})\right],$$

and thus,

$$\begin{aligned}
&\sum_{t=1}^{T} \nabla_t(\mathbf{x}_t - \mathbf{x}^\star) \\
&= \sum_{t=1}^{T} g_t(\mathbf{x}_t) - \sum_{t=1}^{T} g_t(\mathbf{x}^\star) \\
&\leq \sum_{t=1}^{T} g_t(\mathbf{x}_t) - \sum_{t=1}^{T} g_t(\mathbf{x}_{t+1}) + \mathbf{E}[g_0(\mathbf{x}^\star) - g_0(\mathbf{x}_1)] \\
&\leq \sum_{t=1}^{T} \nabla_t(\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{1}{\eta}\,\mathbf{E}[\|\mathbf{n}\|^*\|\mathbf{x}^\star - \mathbf{x}_1\|] \qquad \text{Cauchy-Schwarz} \\
&\leq \sum_{t=1}^{T} \nabla_t(\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{1}{\eta}\sigma D.
\end{aligned}$$

Hence,

$$\begin{aligned}
&\textstyle\sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \\
&\quad \leq \textstyle\sum_{t=1}^{T} \nabla_t^\top(\mathbf{x}_t - \mathbf{x}^*) \\
&\leq \textstyle\sum_{t=1}^{T} \nabla_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{1}{\eta}\sigma D \quad \text{above} \\
&\leq G^* \textstyle\sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{x}_{t+1}\| + \frac{1}{\eta}\sigma D. \quad \text{Cauchy-Schwarz}
\end{aligned} \tag{5.5}$$

We now argue that $\|\mathbf{x}_t - \mathbf{x}_{t+1}\| = O(\eta)$. Let

$$h_t(\mathbf{n}) = \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \eta \sum_{s=1}^{t-1} \nabla_s^\top \mathbf{x} + \mathbf{n}^\top \mathbf{x} \right\},$$

and hence $\mathbf{x}_t = \mathbf{E}_{\mathbf{n}\sim\mathcal{D}}[h_t(\mathbf{n})]$. Recalling that $\mathcal{D}(\mathbf{n})$ denotes the value of the probability density function $\mathcal{D}$ over $\mathbf{n} \in \mathbb{R}^n$, we can write:

$$\mathbf{x}_t = \int_{\mathbf{n}\in\mathbb{R}^n} h_t(\mathbf{n})\mathcal{D}(\mathbf{n})d\mathbf{n},$$

and:

$$\mathbf{x}_{t+1} = \int_{\mathbf{n}\in\mathbb{R}^n} h_t(\mathbf{n} + \eta\nabla_t)\mathcal{D}(\mathbf{n})d\mathbf{n} = \int_{\mathbf{n}\in\mathbb{R}^n} h_t(\mathbf{n})\mathcal{D}(\mathbf{n} - \eta\nabla_t)d\mathbf{n}.$$

Notice that $\mathbf{x}_t, \mathbf{x}_{t+1}$ may depend on each other. However, by linearity of expectation, we have that

$$\begin{aligned}
&\|\mathbf{x}_t - \mathbf{x}_{t+1}\| \\
&= \left\| \int_{\mathbf{n}\in\mathbb{R}^n} (h_t(\mathbf{n}) - h_t(\mathbf{n} + \eta\nabla_t))\mathcal{D}(\mathbf{n})d\mathbf{n} \right\| \\
&= \left\| \int_{\mathbf{n}\in\mathbb{R}^n} h_t(\mathbf{n})(\mathcal{D}(\mathbf{n}) - \mathcal{D}(\mathbf{n} - \eta\nabla_t))d\mathbf{n} \right\| \\
&= \left\| \int_{\mathbf{n}\in\mathbb{R}^n} (h_t(\mathbf{n}) - h_t(\mathbf{0}))(\mathcal{D}(\mathbf{n}) - \mathcal{D}(\mathbf{n} - \eta\nabla_t))d\mathbf{n} \right\| \\
&\le \int_{\mathbf{n}\in\mathbb{R}^n} \|h_t(\mathbf{n}) - h_t(\mathbf{0})\||\mathcal{D}(\mathbf{n}) - \mathcal{D}(\mathbf{n} - \eta\nabla_t)|d\mathbf{n} \\
&\le D \int_{\mathbf{n}\in\mathbb{R}^n} |\mathcal{D}(\mathbf{n}) - \mathcal{D}(\mathbf{n} - \eta\nabla_t)|\, d\mathbf{n} \quad \text{since } \|\mathbf{x}_t - h_t(\mathbf{0})\| \le D \\
&\le DL \cdot \eta\|\nabla_t\|^* \le \eta DLG^*. \text{ since } \mathcal{D} \text{ is } (\sigma, L)\text{-stable.}
\end{aligned}$$

Substituting this bound back into (5.5) we have

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \le \eta LDG^{*2}T + \tfrac{1}{\eta}\sigma D.$$

□

For the choice of $\mathcal{D}$ as the uniform distribution over the unit hypercube $[0,1]^n$, which has parameters $\sigma_2 \le \sqrt{n}$ and $L_2 \le 1$ for the Euclidean norm, the optimal choice of η gives a regret bound of $DGn^{1/4}\sqrt{T}$. This is a factor $n^{1/4}$ worse than the online gradient descent regret bound of Theorem 3.1. For certain decision sets $\mathcal{K}$ a better choice of distribution $\mathcal{D}$ results in near-optimal regret bounds.

5.5.2 Perturbation for linear cost functions

The case of linear cost functions $f_t(\mathbf{x}) = \mathbf{g}_t^\top \mathbf{x}$ is of particular interest in the context of randomized regularization. Denote

$$w_t(\mathbf{n}) = \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \eta \sum_{s=1}^{t} \mathbf{g}_s^\top \mathbf{x} + \mathbf{n}^\top \mathbf{x} \right\}.$$

By linearity of expectation, we have that

$$f_t(\mathbf{x}_t) = f_t(\mathop{\mathbf{E}}_{\mathbf{n}\sim\mathcal{D}}[w_t(\mathbf{n})]) = \mathop{\mathbf{E}}_{\mathbf{n}\sim\mathcal{D}}[f_t(w_t(\mathbf{n}))].$$

Thus, instead of computing $\mathbf{x}_t$ precisely, we can sample a single vector $\mathbf{n}_0 \sim \mathcal{D}$, and use it to compute $\hat{\mathbf{x}}_t = w_t(\mathbf{n}_0)$, as illustrated in Algorithm 17.

Algorithm 17 FPL for linear losses

Input: $\eta > 0$, distribution $\mathcal{D}$ over $\mathbb{R}^n$, decision set $\mathcal{K} \subseteq \mathbb{R}^n$.
Sample $\mathbf{n}_0 \sim \mathcal{D}$. Let $\hat{\mathbf{x}}_1 \in \arg\min_{\mathbf{x}\in\mathcal{K}}\{-\mathbf{n}_0^\top \mathbf{x}\}$.
for $t = 1$ to T **do**
    Predict $\hat{\mathbf{x}}_t$.
    Observe the linear loss function, suffer loss $\mathbf{g}_t^\top \mathbf{x}_t$.
    Update

$$\hat{\mathbf{x}}_t = \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \eta \sum_{s=1}^{t-1} \mathbf{g}_s^\top \mathbf{x} + \mathbf{n}_0^\top \mathbf{x} \right\}$$

end for

By the above arguments, we have that the expected regret for the random variables $\hat{\mathbf{x}}_t$ is the same as that for $\mathbf{x}_t$. We obtain the following Corollary:

Corollary 5.9.

$$\mathop{\mathbf{E}}_{\mathbf{n}_0\sim\mathcal{D}} \left[\sum_{t=1}^{T} f_t(\hat{\mathbf{x}}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \right] \leq \eta L D G^{*2} T + \frac{1}{\eta}\sigma D.$$

The main advantage of this algorithm is computational: with a single linear optimization step over the decision set $\mathcal{K}$ (which does not even have to be convex!), we attain near optimal expected regret bounds.

5.5.3 Follow-the-perturbed-leader for expert advice

An interesting special case (and in fact the first use of perturbation in decision making) is that of non-negative linear cost functions over the unit n-dimensional simplex with costs bounded by one, or the problem of prediction of expert advice we have considered in chapter 1.

Algorithm 17 applied to the probability simplex and with exponentially distributed noise is known as the follow-the-perturbed-leader for prediction from expert advice method. We spell it out in Algorithm 18.

Algorithm 18 FPL for prediction from expert advice

Input: $\eta > 0$
Draw n exponentially distributed variables $\mathbf{n}(i) \sim e^{-\eta x}$.
Let $\mathbf{x}_1 = \arg\min_{\mathbf{e}_i \in \Delta_n} \{-\mathbf{e}_i^\top \mathbf{n}\}$.
for $t = 1$ to T **do**
    Predict using expert i_t such that $\hat{\mathbf{x}}_t = \mathbf{e}_{i_t}$
    Observe the loss vector and suffer loss $\mathbf{g}_t^\top \hat{\mathbf{x}}_t = \mathbf{g}_t(i_t)$
    Update (w.l.o.g. choose $\hat{\mathbf{x}}_{t+1}$ to be a vertex)

$$\hat{\mathbf{x}}_{t+1} = \arg\min_{\mathbf{x} \in \Delta_n} \left\{ \sum_{s=1}^{t} \mathbf{g}_s^\top \mathbf{x} - \mathbf{n}^\top \mathbf{x} \right\}$$

end for

Notice that we take the perturbation to be distributed according to the one-sided negative exponential distribution, i.e., $\mathbf{n}(i) \sim e^{-\eta x}$, or more precisely

$$\Pr[\mathbf{n}(i) \leq x] = 1 - e^{-\eta x} \quad \forall x \geq 0.$$

Corollary 5.9 gives regret bounds that are suboptimal for this special case, thus we give here an alternative analysis that gives tight bounds up to constants amounting to the following theorem.

Theorem 5.10. *Algorithm 18 outputs a sequence of predictions* $\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_T \in \Delta_n$ *such that:*

$$(1 - \eta)\, \mathbf{E}\left[\sum_t \mathbf{g}_t^\top \hat{\mathbf{x}}_t \right] \leq \min_{\mathbf{x}^\star \in \Delta_n} \sum_t \mathbf{g}_t^\top \mathbf{x}^\star + \frac{4 \log n}{\eta}.$$

Notice that as a special case of the above theorem, choosing $\eta = \sqrt{\frac{\log n}{T}}$ yields a regret bound of

$$\text{Regret}_T = O(\sqrt{T \log n}),$$

which is equivalent up to constant factors to the guarantee given for the Hedge algorithm in Theorem 1.5.

Proof. We start with the same analysis technique throughout this chapter: let $\mathbf{g}_0 = -\mathbf{n}$. It follows from Lemma 5.4 applied to the functions $\{f_t(\mathbf{x}) = \mathbf{g}_t^\top \mathbf{x}\}$ that

$$\mathbf{E}\left[\sum_{t=0}^{T} \mathbf{g}_t^\top \mathbf{u} \right] \geq \mathbf{E}\left[\sum_{t=0}^{T} \mathbf{g}_t^\top \hat{\mathbf{x}}_{t+1} \right],$$

and thus,

$$\begin{aligned}
\mathbf{E}\left[\sum_{t=1}^{T} \mathbf{g}_t^\top(\hat{\mathbf{x}}_t - \mathbf{x}^\star)\right] &\leq \mathbf{E}\left[\sum_{t=1}^{T} \mathbf{g}_t^\top(\hat{\mathbf{x}}_t - \hat{\mathbf{x}}_{t+1})\right] + \mathbf{E}[\mathbf{g}_0^\top(\mathbf{x}^\star - \mathbf{x}_1)] \\
&\leq \mathbf{E}\left[\sum_{t=1}^{T} \mathbf{g}_t^\top(\hat{\mathbf{x}}_t - \hat{\mathbf{x}}_{t+1})\right] + \mathbf{E}[\|\mathbf{n}\|_\infty \|\mathbf{x}^\star - \mathbf{x}_1\|_1] \\
&\leq \sum_{t=1}^{T} \mathbf{E}\left[\mathbf{g}_t^\top(\hat{\mathbf{x}}_t - \hat{\mathbf{x}}_{t+1}) \mid \hat{\mathbf{x}}_t\right] + \frac{4}{\eta}\log n, \qquad (5.6)
\end{aligned}$$

where the second inequality follows by the generalized Cauchy-Schwarz inequality, and the last inequality follows since (see exercises)

$$\mathop{\mathbf{E}}_{\mathbf{n}\sim\mathcal{D}}[\|\mathbf{n}\|_\infty] \leq \frac{2\log n}{\eta}.$$

We proceed to bound $\mathbf{E}[\mathbf{g}_t^\top(\hat{\mathbf{x}}_t - \hat{\mathbf{x}}_{t+1})|\hat{\mathbf{x}}_t]$, which is naturally bounded by the probability that $\hat{\mathbf{x}}_t$ is not equal to $\hat{\mathbf{x}}_{t+1}$ multiplied by the maximum value that $\mathbf{g}_t$ can attain (i.e., its ℓ_∞ norm):

$$\mathbf{E}[\mathbf{g}_t^\top(\hat{\mathbf{x}}_t - \hat{\mathbf{x}}_{t+1}) \mid \hat{\mathbf{x}}_t] \leq \|\mathbf{g}_t\|_\infty \cdot \Pr[\hat{\mathbf{x}}_t \neq \hat{\mathbf{x}}_{t+1} \mid \hat{\mathbf{x}}_t] \leq \Pr[\hat{\mathbf{x}}_t \neq \hat{\mathbf{x}}_{t+1} \mid \hat{\mathbf{x}}_t].$$

Above we have that $\|\mathbf{g}_t\|_\infty \leq 1$ by assumption that the losses are bounded by one.

To bound the latter, notice that the probability $\hat{\mathbf{x}}_t = \mathbf{e}_{i_t}$ is the leader at time t is the probability that $-\mathbf{n}(i_t) > v$ for some value v that depends on the entire loss sequence till now. On the other hand, given $\hat{\mathbf{x}}_t$, we have that $\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{x}}_t$ remains the leader if $-\mathbf{n}(i_t) > v + \mathbf{g}_t(i_t)$, since it was a leader by a margin of more than the cost it will suffer. Thus,

$$\begin{aligned}
\Pr[\hat{\mathbf{x}}_t \neq \hat{\mathbf{x}}_{t+1} \mid \hat{\mathbf{x}}_t] &= 1 - \Pr[-\mathbf{n}(i_t) > v + \mathbf{g}_t(i_t) \mid -\mathbf{n}(i_t) > v] \\
&= 1 - \frac{\int_{v+\mathbf{g}_t(i_t)}^{\infty} \eta e^{-\eta x}}{\int_v^\infty \eta e^{-\eta x}} \\
&= 1 - e^{-\eta \mathbf{g}_t(i_t)} \\
&\leq \eta \mathbf{g}_t(i_t) = \eta \mathbf{g}_t^\top \hat{\mathbf{x}}_t.
\end{aligned}$$

Substituting this bound back into (5.6) we have

$$\mathbf{E}[\textstyle\sum_{t=1}^{T} \mathbf{g}_t^\top(\hat{\mathbf{x}}_t - \mathbf{x}^\star)] \leq \eta \sum_t \mathbf{E}_t[\mathbf{g}_t^\top \hat{\mathbf{x}}_t] + \frac{4\log n}{\eta},$$

which simplifies to the Theorem. □

5.6 * Adaptive Gradient Descent

Thus far we have introduced regularization as a general methodology for deriving online convex optimization algorithms. The main theorem of this chapter, Theorem 5.2, bounds the regret of the RFTL algorithm for any strongly convex regularizer as

$$\text{Regret}_T \leq \max_{\mathbf{u} \in \mathcal{K}} \sqrt{2 \sum_t \|\nabla_t\|_t^{*2} B_R(\mathbf{u}||\mathbf{x}_1)}. \tag{5.7}$$

In addition, we have seen how to derive the online gradient descent and the multiplicative weights algorithms as special cases of the RFTL methodology. But are there other special cases of interest, besides these two basic algorithms, that warrant such general and abstract treatment?

There are surprisingly few cases of interest besides the Euclidean and Entropic regularizations and their matrix analogues[12]. However, in this chapter we will give some justification of the abstract treatment of regularization.

Our treatment is motivated by the following question: thus far we have thought of R as a strongly convex function. But which strongly convex function should we choose to minimize regret? This is a deep and difficult question which has been considered in the optimization literature since its early developments. Naturally, the optimal regularization should depend on both the convex underlying decision set, as well as the actual cost functions (see exercises for a natural candidate of a regularization function that depends on the convex decision set).

We shall treat this question no differently than we treat other optimization problems throughout this manuscript itself: we'll learn the optimal regularization online! That is, a regularizer that adapts to the sequence of cost functions and is in a sense the "optimal" regularization to use in hindsight. This gives rise to the AdaGrad (Adaptive subGradient method) algorithm 19, which explicitely optimizes over the regularization choice in line (5) to minimize the gradient norms, which is the dominant expression in (5.7).

Algorithm 19 AdaGrad

1: Input: parameters $\eta, \mathbf{x}_1 \in \mathcal{K}$.
2: Initialize: $G_0 = \mathbf{0}$,
3: **for** $t = 1$ to T **do**
4: Predict $\mathbf{x}_t$, suffer loss $f_t(\mathbf{x}_t)$.
5: Update $G_t = G_{t-1} + \nabla_t \nabla_t^\top$ and define

[Diagonal version] $$H_t = \underset{H \succeq 0, H = \mathbf{diag}(H)}{\arg\min} \left\{ G_t \bullet H^{-1} + \mathbf{Tr}(H) \right\} = \mathbf{diag}({G_t}^{1/2})$$

[Full matrix version] $$H_t = \underset{H \succeq \mathbf{0}}{\arg\min} \left\{ G_t \bullet H^{-1} + \mathbf{Tr}(H) \right\} = {G_t}^{1/2}$$

6: Update

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \eta H_t^{-1} \nabla_t$$

$$\mathbf{x}_{t+1} = \underset{\mathbf{x} \in \mathcal{K}}{\arg\min} \|\mathbf{y}_{t+1} - \mathbf{x}\|_{H_t}^2$$

7: **end for**

AdaGrad comes in two versions: diagonal and full matrix, the first being particularly efficient to implement with negligible computational overhead over online gradient descent. In the algorithm definition and throughout this chapter, the notation A^{-1} refers to the Moore-Penrose pseudoinverse of the matrix A.

The computation in line (5) finds the regularization matrix H which minimizes the norm of the gradients from within the positive semi-definite cone, with or without a diagonal constraint. This is closely related, as we shall see, to optimization w.r.t. two natural sets of matrices:

1. $\mathcal{H}_1 = \{H = \mathbf{diag}(H), H \succeq 0 \ , \ \mathbf{Tr}(H) \leq 1\}$
2. $\mathcal{H}_2 = \{H \succeq 0 \ , \ \mathbf{Tr}(H) \leq 1\}$.

This results in a regularization matrix that is provably optimal in the following sense,

Lemma 5.11. *For $\mathcal{H}_i \in \{\mathcal{H}_1, \mathcal{H}_2\}$ with the corresponding H_T,*

$$\sqrt{\min_{H \in \mathcal{H}_i} \sum_{t=1}^{T} \|\nabla_t\|_H^{*2}} \quad = \mathbf{Tr}(H_T).$$

Using this lemma, we show the regret of AdaGrad is at most a constant factor larger than the minimum regret of all RFTL algorithm with

regularization functions whose Hessian is fixed and belongs to the class $\mathcal{H}_i$. Furthermore, the regret of the diagonal version can be a factor $\sqrt{d}$ smaller than that of online gradient descent for certain gradient geometries. The regret bound on AdaGrad is formally stated in the following theorem.

Theorem 5.12. *Let $\{\mathbf{x}_t\}$ be defined by Algorithm 19 with parameters $\eta = D$ (full matrix) or $\eta = D_\infty$ (diagonal). Then for any $\mathbf{x}^\star \in \mathcal{K}$,*

$$\text{Regret}_T(\textit{AdaGrad-diag}) \leq \sqrt{2} D_\infty \sqrt{\min_{H \in \mathcal{H}_1} \sum_t \|\nabla_t\|_H^{*2}},$$

$$\text{Regret}_T(\textit{AdaGrad-full}) \leq \sqrt{2} D \sqrt{\min_{H \in \mathcal{H}_2} \sum_t \|\nabla_t\|_H^{*2}}.$$

Before proceeding to the analysis, we consider when the regret bounds for AdaGrad improve upon those of Online Gradient Descent. One such case is when $\mathcal{K}$ is the unit cube in d-dimensional Euclidean space. This convex set has $D_\infty = 1$ and $D = \sqrt{d}$. Lemma 5.11 and Theorems 5.12,5.2 imply that the regret of diagonal AdaGrad and OGD are bounded by

$$\text{Regret}_T(\text{AdaGrad-diag}) \leq \sqrt{2}\mathbf{Tr}(\mathbf{diag}(G_T)^{1/2}),$$

$$\text{Regret}_T(\text{OGD}) \leq \sqrt{2d} \sqrt{\sum_t \|\nabla_t\|^2} = \sqrt{2d\mathbf{Tr}(\mathbf{diag}(G_T))}.$$

The relationship between the two terms depends on the matrix $\mathbf{diag}(G_T)$. If this matrix is sparse, then AdaGrad has a superior bound by at most $\sqrt{d}$ factor. For other convex bodies, such as the Euclidean ball, and when the matrix G_T is dense, the regret of OGD can be a factor $\sqrt{d}$ lower.

5.6.1 Analysis of adaptive regularization

We proceed with the proof of Theorem 5.12. The first component is the following Lemma, which generalizes the RFTL analysis to changing regularization.

Lemma 5.13. *Let $H_0 = \arg\min_{H \succeq 0} \{\mathbf{Tr}(H)\} = 0$,*

$$\text{Regret}_T(\textit{GenAdaReg}) \leq \frac{\eta}{2}(G_T \bullet H_T^{-1} + \mathbf{Tr}(H_T)) + \frac{1}{2\eta} \sum_{t=0}^{T} \|\mathbf{x}_t - \mathbf{x}^\star\|^2_{H_t - H_{t-1}}.$$

Proof. By the definition of $\mathbf{y}_{t+1}$:

$$\begin{aligned}\mathbf{y}_{t+1} - \mathbf{x}^\star &= \mathbf{x}_t - \mathbf{x}^\star - \eta {H_t}^{-1} \nabla_t \\ H_t(\mathbf{y}_{t+1} - \mathbf{x}^\star) &= H_t(\mathbf{x}_t - \mathbf{x}^\star) - \eta \nabla_t.\end{aligned}$$

Multiplying the transpose of the first equation by the second we get

$$\begin{aligned}(\mathbf{y}_{t+1} - \mathbf{x}^\star)^\top H_t (\mathbf{y}_{t+1} - \mathbf{x}^\star) = \\ (\mathbf{x}_t - \mathbf{x}^\star)^\top H_t (\mathbf{x}_t - \mathbf{x}^\star) - 2\eta \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star) + \eta^2 \nabla_t^\top H_t^{-1} \nabla_t. \end{aligned} \tag{5.8}$$

Since $\mathbf{x}_{t+1}$ is the projection of $\mathbf{y}_{t+1}$ in the norm induced by H_t, we have (see §2.1.1)

$$(\mathbf{y}_{t+1} - \mathbf{x}^\star)^\top H_t (\mathbf{y}_{t+1} - \mathbf{x}^\star) = \|\mathbf{y}_{t+1} - \mathbf{x}^\star\|_{H_t}^2 \geq \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|_{H_t}^2.$$

This inequality is the reason for using generalized projections as opposed to standard projections, which were used in the analysis of online gradient descent (see §3.1 Equation (3.2)). This fact together with (5.8) gives

$$\nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star) \leq \frac{\eta}{2} \nabla_t^\top H_t^{-1} \nabla_t + \frac{1}{2\eta} \left(\|\mathbf{x}_t - \mathbf{x}^\star\|_{H_t}^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|_{H_t}^2 \right).$$

Now, summing up over $t = 1$ to T we get that

$$\begin{aligned}\sum_{t=1}^T \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^\star) &\leq \frac{\eta}{2} \sum_{t=1}^T \nabla_t^\top H_t^{-1} \nabla_t + \frac{1}{2\eta} \|\mathbf{x}_1 - \mathbf{x}^\star\|_{H_0}^2 \\ &+ \frac{1}{2\eta} \sum_{t=1}^T \left(\|\mathbf{x}_t - \mathbf{x}^\star\|_{H_t}^2 - \|\mathbf{x}_t - \mathbf{x}^\star\|_{H_{t-1}}^2 \right) - \frac{1}{2\eta} \|\mathbf{x}_{T+1} - \mathbf{x}^\star\|_{H_T}^2 \\ &\leq \frac{\eta}{2} \sum_{t=1}^T \nabla_t^\top H_t^{-1} \nabla_t + \frac{1}{2\eta} \sum_{t=0}^T \|\mathbf{x}_t - \mathbf{x}^\star\|_{H_t - H_{t-1}}^2.\end{aligned} \tag{5.9}$$

In the last inequality we use the definition $H_0 = 0$. We proceed to bound the first term. To this end, define the functions

$$\Psi_t(H) = \nabla_t \nabla_t^\top \bullet H^{-1} \ , \ \Psi_0(H) = \mathbf{Tr}(H).$$

By definition, H_t is the minimizer of $\sum_{i=0}^t \Psi_i$ over $\mathcal{H}$. Therefore, using the BTL Lemma 5.4, we have that

$$\begin{aligned}\sum_{t=1}^T \nabla_t^\top H_t^{-1} \nabla_t &= \textstyle\sum_{t=1}^T \Psi_t(H_t) \\ &\leq \textstyle\sum_{t=1}^T \Psi_t(H_T) + \Psi_0(H_T) - \Psi_0(H_0) \\ &= G_T \bullet H_T^{-1} + \mathbf{Tr}(H_T).\end{aligned}$$

□

We can now continue with the proof of Theorem 5.12.

Proof of Theorem 5.12. We bound both parts of Lemma 5.13, with the following two lemmas,

Lemma 5.14. *For both the diagonal and full matrix versions of AdaGrad, the following holds*

$$G_T \bullet H_T^{-1} \leq \mathbf{Tr}(H_T).$$

Lemma 5.15. *Let D_∞ denote the ℓ_∞ diameter of $\mathcal{K}$, and D the Euclidean diameter. Then the following bounds hold,*

$$\textit{Diagonal AdaGrad:} \qquad \sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{x}^\star\|^2_{H_t - H_{t-1}} \leq D_\infty^2 \mathbf{Tr}(H_T).$$

$$\textit{Full matrix AdaGrad:} \qquad \sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{x}^\star\|^2_{H_t - H_{t-1}} \leq D^2 \mathbf{Tr}(H_T).$$

Now combining Lemma 5.13 with the above two lemmas, and using $\eta = \frac{D}{\sqrt{2}}$ or $\eta = \frac{D_\infty}{\sqrt{2}}$ appropriately, we obtain the theorem. □

We proceed to complete the proof of the two lemmas above.

Proof of Lemma 5.14. The optimization problem of choosing H_t in line (5) of Algorithm 19 has an explicit solution, given in the following proposition (whose proof is left as an exercise).

Proposition 5.16. *Consider the following optimization problems, for $A \succcurlyeq 0$:*

$$\min_{X \succeq 0, \mathbf{Tr}(X) \leq 1} \left\{ X^{-1} \bullet A \right\} \qquad \min_{X \succeq 0} \left\{ A \bullet X^{-1} + \mathbf{Tr}(X) \right\}.$$

Then the global optimizer to these problems is obtained at $X = \frac{A^{1/2}}{\mathbf{Tr}(A^{1/2})}$ and $X = A^{1/2}$ respectively. Over the set of diagonal matrices, the global optimizer is obtained at $X = \frac{\mathbf{diag}(A)^{1/2}}{\mathbf{Tr}(A^{1/2})}$ and $X = \mathbf{diag}(A)^{1/2}$ respectively.

A direct corollary of this proposition gives Lemma 5.11 as follows:

Corollary 5.17.

$$\sqrt{\min_{H \in \mathcal{H}} \sum_t \|\nabla_t\|_H^{*2}} = \sqrt{\min_{H \in \mathcal{H}} \mathbf{Tr}(H^{-1} \textstyle\sum_t \nabla_t \nabla_t^\top)}$$

$$= \mathbf{Tr} \sqrt{\textstyle\sum_t \nabla_t \nabla_t^\top} = \mathbf{Tr}(H_T)$$

□

The remaining term from Lemma 5.13 is the expression $\sum_{t=0}^{T} \|\mathbf{x}_t - \mathbf{x}^\star\|^2_{H_t - H_{t-1}}$, which we proceed to bound.

Proof of Lemma 5.15. By definition $G_t \succcurlyeq G_{t-1}$, and hence using proposition 5.16 and the definition of H_t in line (5), we have that $H_t = \mathbf{diag}(G_t^{1/2}) \succcurlyeq \mathbf{diag}(G_{t-1}^{1/2}) = H_{t-1}$. Since for a diagonal matrix H it holds that $\mathbf{x}^\top H \mathbf{x} \leq \|\mathbf{x}\|_\infty^2 \mathbf{Tr}(H)$, we have

$$
\begin{aligned}
&\sum_{t=1}^{T} (\mathbf{x}_t - \mathbf{x}^\star)^\top (H_t - H_{t-1})(\mathbf{x}_t - \mathbf{x}^\star) \\
&\leq \sum_{t=1}^{T} D_\infty^2 \mathbf{Tr}(H_t - H_{t-1}) && \text{diagonal structure, } H_t - H_{t-1} \succeq 0 \\
&= D_\infty^2 \sum_{t=1}^{T} (\mathbf{Tr}(H_t) - \mathbf{Tr}(H_{t-1})) && \text{linearity of the trace} \\
&\leq D_\infty^2 \mathbf{Tr}(H_T).
\end{aligned}
$$

Next, we consider the full matrix case. By definition $G_t \succcurlyeq G_{t-1}$, and hence $H_t \succcurlyeq H_{t-1}$. Thus,

$$
\begin{aligned}
&\sum_{t=1}^{T} (\mathbf{x}_t - \mathbf{x}^\star)^\top (H_t - H_{t-1})(\mathbf{x}_t - \mathbf{x}^\star) \\
&\leq \sum_{t=1}^{T} D^2 \lambda_{\max}(H_t - H_{t-1}) \\
&\leq D^2 \sum_{t=1}^{T} \mathbf{Tr}(H_t - H_{t-1}) && A \succcurlyeq 0 \;\Rightarrow\; \lambda_{\max}(A) \leq \mathbf{Tr}(A) \\
&= D^2 \sum_{t=1}^{T} (\mathbf{Tr}(H_t) - \mathbf{Tr}(H_{t-1})) && \text{linearity of the trace} \\
&\leq D^2 \mathbf{Tr}(H_T).
\end{aligned}
$$

□

5.7 Bibliographic Remarks

Regularization in the context of online learning was first studied in [Grove et al., 2001] and [Kivinen and Warmuth, 2001]. The influential paper of Kalai and Vempala [2005] coined the term "follow-the-leader" and introduced many of the techniques that followed in OCO. The latter paper studies random perturbation as a regularization and analyzes the follow-the-perturbed-leader algorithm, following an early development by Hannan [1957] that was overlooked in learning for many years.

In the context of OCO, the term follow-the-regularized-leader was coined in [Shalev-Shwartz and Singer, 2007, Shalev-Shwartz, 2007], and at roughly the same time an essentially identical algorithm was called "RFTL" in [Abernethy et al., 2008]. The equivalence of RFTL and Online Mirror Descent was observed by [Hazan and Kale, 2008]. The AdaGrad algorithm was introduced in [Duchi et al., 2010, 2011], its diagonal version was also discovered in parallel in [McMahan and Streeter, 2010]. The analysis of AdaGrad presented in this chapter is due to [Gupta et al., 2017].

Adaptive regularization has received significant attention due to its success in training deep neural networks, and notably the development of adaptive algorithms that incorporate momentum and other heuristics, most popular of which are AdaGrad, RMSprop [Tieleman and Hinton, 2012] and Adam [Kingma and Ba, 2014]. For a survey of optimization for deep learning, see the comprehensive text of Goodfellow et al. [2016].

There is a strong connection between randomized perturbation and deterministic regularization. For some special cases, adding randomization can be thought of as a special case of deterministic strongly convex regularization, see [Abernethy et al., 2014, 2016].

5.8 Exercises

1. This exercise concerns the notion of a dual norm.

(a) Show that the dual norm to a matrix norm given by $A \succ 0$ corresponds to the matrix norm of A^{-1}.

(b) Prove the generalized Cauchy-Schwarz inequality for any norm, i.e.,

$$\mathbf{x}^\top \mathbf{y} \leq \|\mathbf{x}\| \|\mathbf{y}\|^*.$$

2. Prove that the Bregman divergence is equal to the local norm at an intermediate point, that is:

$$B_R(\mathbf{x}||\mathbf{y}) = \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_{\mathbf{z}}^2,$$

where $\mathbf{z} \in [\mathbf{x}, \mathbf{y}]$, and the interval $[\mathbf{x}, \mathbf{y}]$ is defined as

$$[\mathbf{x}, \mathbf{y}] = \{\mathbf{v} = \alpha\mathbf{x} + (1-\alpha)\mathbf{y} \ , \ \alpha \in [0,1]\}.$$

3. Let $R(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}-\mathbf{x}_0\|^2$ be the (shifted) Euclidean regularization function. Prove that the corresponding Bregman divergence is the Euclidean metric. Conclude that projections with respect to this divergence are standard Euclidean projections.

4. Prove that the agile and lazy versions of the OMD meta-algorithm are different, in the sense that they can produce different predictions over the same setting and cost functions. Show this for the case that the regularization is Euclidean and the decision set is the Euclidean ball.

5. For this problem the decision set is the n-dimensional simplex. Let $R(\mathbf{x}) = \mathbf{x}\log\mathbf{x}$ be the negative entropy regularization function. Prove that the corresponding Bregman divergence is the relative entropy, and prove that the diameter D_R of the n-dimensional simplex with respect to this function is bounded by $\log n$. Show that projections with respect to this divergence over the simplex amounts to scaling by the ℓ_1 norm.

6. Prove that for the uniform distribution $\mathcal{D}$ over the unit hypercube $[0,1]^n$, the parameters σ, L defined in §5.5 with respect to the Euclidean norm can be bounded as $\sigma < \sqrt{n}$, $L \leq 1$.

7. Let $\mathcal{D}$ be a one-sided multi-dimensional exponential distribution, such that a vector $\mathbf{n} \sim \mathcal{D}$ is distributed over each coordinate exponentially:

$$\Pr[\mathbf{n}_i \leq x] = 1 - e^{-x} \quad \forall i \in [n], \ x \geq 0.$$

Prove that

$$\mathop{\mathbf{E}}_{\mathbf{n}\sim\mathcal{D}}[\|\mathbf{n}\|_\infty] \leq 2\log n.$$

(Hint: use the Chernoff bound)
Extra credit: prove that $\mathbf{E}_{\mathbf{n}\sim\mathcal{D}}[\|\mathbf{n}\|_\infty] = H_n$, where H_n is the n-th harmonic number.

8. * A set $\mathcal{K} \subseteq \mathbb{R}^d$ is symmetric if $\mathbf{x} \in \mathcal{K}$ implies $-\mathbf{x} \in \mathcal{K}$. Symmetric sets gives rise to a natural definition of a norm. Define the function $\|\cdot\|_\mathcal{K} : \mathbb{R}^d \mapsto \mathbb{R}$ as

$$\|\mathbf{x}\|_\mathcal{K} = \arg\min_{\alpha>0} \left\{ \frac{1}{\alpha}\mathbf{x} \in \mathcal{K} \right\}.$$

Prove that $\|\cdot\|_\mathcal{K}$ is a norm if and only if $\mathcal{K}$ is convex.

9. ** Prove a lower bound of $\Omega(T)$ on the regret of the RFTL algorithm with $\|\cdot\|_\mathcal{K}$ as a regularizer.

10. * Prove that for positive definite matrices $A \succcurlyeq B \succ 0$ it holds that

(a) $A^{1/2} \succcurlyeq B^{1/2}$

(b) $2\mathbf{Tr}((A-B)^{1/2}) + \mathbf{Tr}(A^{-1/2}B) \leq 2\mathbf{Tr}(A^{1/2})$.

11. * Consider the following minimization problem where $A \succ 0$:

$$\begin{aligned} \min_X \quad & \mathbf{Tr}(X^{-1}A) \\ \text{subject to} \quad & X \succ 0 \\ & \mathbf{Tr}(X) \leq 1. \end{aligned}$$

Prove that its minimizer is given by $X = A^{1/2}/\mathbf{Tr}(A^{1/2})$, and the minimum is obtained at $\mathbf{Tr}^2(A^{1/2})$.

Chapter 6

Bandit Convex Optimization

In many real-world scenarios the feedback available to the decision maker is noisy, partial or incomplete. Such is the case in online routing in data networks, in which an online decision maker iteratively chooses a path through a known network, and her loss is measured by the length (in time) of the path chosen. In data networks, the decision maker can measure the RTD (round trip delay) of a packet through the network, but rarely has access to the congestion pattern of the entire network.

Another useful example is that of online ad placement in web search. The decision maker iteratively chooses an ordered set of ads from an existing pool. Her reward is measured by the viewer's response—if the user clicks a certain ad, a reward is generated according to the weight assigned to the particular ad. In this scenario, the search engine can inspect which ads were clicked through, but cannot know whether different ads, had they been chosen to be displayed, would have been clicked through or not.

The examples above can readily be modeled in the OCO framework, with the underlying sets being the convex hull of decisions. The pitfall of the general OCO model is the feedback; it is unrealistic to expect that the decision maker has access to a gradient oracle at any point in the space for every iteration of the game.

6.1 The Bandit Convex Optimization Setting

The Bandit Convex Optimization (short: BCO) model is identical to the general OCO model we have explored in previous chapters with the only difference being the feedback available to the decision maker.

To be more precise, the BCO framework can be seen as a structured

repeated game. The protocol of this learning framework is as follows: At iteration t, the online player chooses $\mathbf{x}_t \in \mathcal{K}$. After committing to this choice, a convex cost function $f_t \in \mathcal{F} : \mathcal{K} \mapsto \mathbb{R}$ is revealed. Here $\mathcal{F}$ is the bounded family of cost functions available to the adversary. The cost incurred to the online player is the value of the cost function at the point she committed to $f_t(\mathbf{x}_t)$. As opposed to the OCO model, in which the decision maker has access to a gradient oracle for f_t over $\mathcal{K}$, in BCO **the loss $f_t(\mathbf{x}_t)$ is the only feedback available to the online player at iteration t.** In particular, the decision maker does not know the loss had she chosen a different point $\mathbf{x} \in \mathcal{K}$ at iteration t.

As before, let T denote the total number of game iterations (i.e., predictions and their incurred loss). Let $\mathcal{A}$ be an algorithm for BCO, which maps a certain game history to a decision in the decision set. We formally define the regret of $\mathcal{A}$ that predicted $x_1, ..., x_T$ to be

$$\text{Regret}_T(\mathcal{A}) = \sup_{\{f_1,...,f_T\} \subseteq \mathcal{F}} \left\{ \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x} \in \mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}) \right\}.$$

6.2 The Multiarmed Bandit (MAB) Problem

A classical model for decision making under uncertainty is the multiarmed bandit (MAB) model. The term MAB nowadays refers to a multitude of different variants and sub-scenarios that are too large to survey. This section addresses perhaps the simplest variant—the non-stochastic MAB problem—which is defined as follows:

Iteratively, a decision maker chooses between n different actions $i_t \in \{1, 2, ..., n\}$, while, at the same time, an adversary assigns each action a loss in the range $[0, 1]$. The decision maker receives the loss for i_t and observes this loss, and nothing else. The goal of the decision maker is to minimize her regret.

The reader undoubtedly observes this setting is identical to the setting of prediction from expert advice, the only difference being the feedback available to the decision maker: whereas in the expert setting the decision maker can observe the rewards or losses for all experts in retrospect, in the MAB setting, only the losses of the decisions actually chosen are known.

It is instructive to explicitly model this problem as a special case of BCO. Take the decision set to be the set of all distributions over n actions, i.e., $\mathcal{K} = \Delta_n$ is the n-dimensional simplex. The loss function is taken to be the

linearization of the costs of the individual actions, that is:

$$f_t(\mathbf{x}) = \ell_t^\top \mathbf{x} = \sum_{i=1}^{n} \ell_t(i)\mathbf{x}(i) \quad \forall \mathbf{x} \in \mathcal{K},$$

where $\ell_t(i)$ is the loss associated with the i'th action at the t'th iteration. Thus, the cost functions are linear functions in the BCO model.

The MAB problem exhibits an exploration-exploitation tradeoff: an efficient (low regret) algorithm has to explore the value of the different actions in order to make the best decision. On the other hand, having gained sufficient information about the environment, a reasonable algorithm needs to exploit this action by picking the best action.

The simplest way to attain a MAB algorithm would be to separate exploration and exploitation. Such a method would proceed by

1. With some probability, explore the action space (i.e., by choosing an action uniformly at random). Use the feedback to construct an estimate of the actions' losses.

2. Otherwise, use the estimates to apply a full-information experts algorithm as if the estimates are the true historical costs.

This simple scheme already gives a sublinear regret algorithm, presented in algorithm 20.

Lemma 6.1. *Algorithm 20, with $\mathcal{A}$ being the the online gradient descent algorithm, guarantees the following regret bound:*

$$\mathbf{E}\left[\sum_{t=1}^{T} \ell_t(i_t) - \min_i \sum_{t=1}^{T} \ell_t(i)\right] \leq O(T^{\frac{2}{3}} n^{\frac{2}{3}})$$

Proof. For the random functions $\{\hat{\ell}_t\}$ defined in algorithm 20, notice that

1. $\mathbf{E}[\hat{\ell}_t(i)] = \Pr[b_t = 1] \cdot \Pr[i_t = i | b_t = 1] \cdot \frac{n}{\delta} \ell_t(i) = \ell_t(i)$.

2. $\|\hat{\ell}_t\|_2 \leq \frac{n}{\delta} \cdot |\ell_t(i_t)| \leq \frac{n}{\delta}$.

Therefore the regret of the simple algorithm can be related to that of $\mathcal{A}$ on the estimated functions.

On the other hand, the simple MAB algorithm does not always play according to the distribution generated by $\mathcal{A}$: with probability δ it plays uniformly at random, which may lead to a regret of one on these exploration iterations. Let $S_t \subseteq [T]$ be those iterations in which $b_t = 1$. This is captured by the following lemma:

Algorithm 20 Simple MAB algorithm

Input: OCO algorithm $\mathcal{A}$, parameter δ.
for $t = 1$ to T **do**
  Let b_t be a Bernoulli random variable that equals 1 with probability δ.
  if $b_t = 1$ **then**
    Choose $i_t \in \{1, 2, ..., n\}$ uniformly at random and play i_t.
    Let

$$\hat{\ell}_t(i) = \begin{cases} \frac{n}{\delta} \cdot \ell_t(i_t), & i = i_t \\ 0, & \text{otherwise} \end{cases}.$$

    Let $\hat{f}_t(\mathbf{x}) = \hat{\ell}_t^\top \mathbf{x}$ and update $\mathbf{x}_{t+1} = \mathcal{A}(\hat{f}_1, ..., \hat{f}_t)$.
  else
    Choose $i_t \sim \mathbf{x}_t$ and play i_t.
    Update $\hat{f}_t = 0, \hat{\ell}_t = \mathbf{0}$, $\mathbf{x}_{t+1} = \mathbf{x}_t$.
  end if
end for

Lemma 6.2.

$$\mathbf{E}[\ell_t(i_t)] \leq \mathbf{E}[\hat{\ell}_t^\top x_t] + \delta$$

Proof.

$$\begin{aligned}
&\mathbf{E}[\ell_t(i_t)] \\
&= \Pr[b_t = 1] \cdot \mathbf{E}[\ell_t(i_t) | b_t = 1] \\
&+ \Pr[b_t = 0] \cdot \mathbf{E}[\ell_t(i_t) | b_t = 0] \\
&\leq \delta + \Pr[b_t = 0] \cdot \mathbf{E}[\ell_t(i_t) | b_t = 0] \\
&= \delta + (1 - \delta)\, \mathbf{E}[\ell_t^\top \mathbf{x}_t | b_t = 0] && b_t = 0 \to i_t \sim \mathbf{x}_t, \text{ independent of } l_t \\
&\leq \delta + \mathbf{E}[\ell_t^\top \mathbf{x}_t] && \text{non-negative random variables} \\
&= \delta + \mathbf{E}[\hat{\ell}_t^\top \mathbf{x}_t] && \hat{\ell}_t \text{ is independent of } \mathbf{x}_t
\end{aligned}$$

□

We thus have,

$$\begin{aligned}
&\mathbf{E}[\text{Regret}_T] \\
&= \mathbf{E}[\textstyle\sum_{t=1}^T \ell_t(i_t) - \sum_{t=1}^T \ell_t(i^\star)] \\
&= \mathbf{E}[\textstyle\sum_t \ell_t(i_t) - \sum_t \hat{\ell}_t(i^\star)] && i^\star \text{ is indep. of } \hat{\ell}_t \\
&\leq \mathbf{E}[\textstyle\sum_t \hat{\ell}_t(\mathbf{x}_t) - \min_i \sum_t \hat{\ell}_t(i)] + \delta T && \text{Lemma 6.2} \\
&= \mathbf{E}[\text{Regret}_{S_T}(\mathcal{A})] + \delta \cdot T \\
&\leq \tfrac{3}{2} GD\sqrt{\delta T} + \delta \cdot T && \text{Theorem 3.1}, \mathbf{E}[|S_T|] = \delta T \\
&\leq 3\tfrac{n}{\sqrt{\delta}}\sqrt{T} + \delta \cdot T && \text{For } \Delta_n,\ D \leq 2\ ,\ \|\hat{\ell}_t\| \leq \tfrac{n}{\delta} \\
&= O(T^{\frac{2}{3}} n^{\frac{2}{3}}). && \delta = n^{\frac{2}{3}} T^{-\frac{1}{3}}
\end{aligned}$$

□

6.2.1 EXP3: simultaneous exploration and exploitation

The simple algorithm of the previous section can be improved by combining the exploration and exploitation steps. This gives a near-optimal regret algorithm, called EXP3, presented below.

Algorithm 21 EXP3 - simple version

Input: parameter $\varepsilon > 0$. Set $\mathbf{x}_1 = (1/n)\mathbf{1}$.
for $t \in \{1, 2, ..., T\}$ **do**
  Choose $i_t \sim \mathbf{x}_t$ and play i_t.
  Let

$$\hat{\ell}_t(i) = \begin{cases} \frac{1}{\mathbf{x}_t(i_t)} \cdot \ell_t(i_t), & i = i_t \\ 0, & \text{otherwise} \end{cases}$$

  Update $\mathbf{y}_{t+1}(i) = \mathbf{x}_t(i) e^{-\varepsilon \hat{\ell}_t(i)}$, $\mathbf{x}_{t+1} = \frac{\mathbf{y}_{t+1}}{\|\mathbf{y}_{t+1}\|_1}$
end for

As opposed to the simple multiarmed bandit algorithm, the EXP3 algorithm explores every iteration by always creating an unbiased estimator of the entire loss vector. This results in a possibly large magnitude of the vectors $\hat{\ell}$ and a large gradient bound for use with online gradient descent. However, the large magnitude vectors are created with low probability (proportional to their magnitude), which allows for a finer analysis.

Ultimately, the EXP3 algorithm attains a worst case regret bound of $O(\sqrt{Tn \log n})$, which is nearly optimal (up to a logarithmic term in the number of actions).

Lemma 6.3. *Algorithm 21 with non-negative losses and* $\varepsilon = \sqrt{\frac{\log n}{Tn}}$ *guarantees the following regret bound:*

$$\mathbf{E}[\sum \ell_t(i_t) - \min_i \sum \ell_t(i)] \leq 2\sqrt{Tn \log n}.$$

Proof. For the random losses $\{\hat{\ell}_t\}$ defined in algorithm 21, notice that

$$\begin{aligned} \mathbf{E}[\hat{\ell}_t(i)] &= \Pr[i_t = i] \cdot \tfrac{\ell_t(i)}{\mathbf{x}_t(i)} = \mathbf{x}_t(i) \cdot \tfrac{\ell_t(i)}{\mathbf{x}_t(i)} = \ell_t(i). \\ \mathbf{E}[\mathbf{x}_t^\top \hat{\ell}_t^2] &= \textstyle\sum_i \Pr[i_t = i] \cdot \mathbf{x}_t(i) \hat{\ell}_t(i)^2 \\ &= \textstyle\sum_i \mathbf{x}_t(i)^2 \hat{\ell}_t(i)^2 = \sum_i \ell_t(i)^2 \leq n. \end{aligned} \tag{6.1}$$

Therefore we have $E[\hat{f}_t] = f_t$, and the expected regret with respect to the functions $\{\hat{f}_t\}$ is equal to that with respect to the functions $\{f_t\}$. Thus, the regret with respect to $\hat{\ell}_t$ can be related to that of ℓ_t.

The EXP3 algorithm applies Hedge to the losses given by $\hat{\ell}_t$, which are all non-negative and thus satisfy the conditions of Theorem 1.5. Thus, the expected regret with respect to $\hat{\ell}_t$, can be bounded by,

$$\begin{aligned} \mathbf{E}[\mathrm{Regret}_T] &= \mathbf{E}[\textstyle\sum_{t=1}^T \ell_t(i_t) - \min_i \sum_{t=1}^T \ell_t(i)] & \\ &= \mathbf{E}[\textstyle\sum_{t=1}^T \ell_t(i_t) - \sum_{t=1}^T \ell_t(i^\star)] & \\ &\leq \mathbf{E}[\textstyle\sum_{t=1}^T \hat{\ell}_t(\mathbf{x}_t) - \sum_{t=1}^T \hat{\ell}_t(i^\star)] & i^\star \text{ is indep. of } \hat{\ell}_t \\ &\leq \mathbf{E}[\varepsilon \textstyle\sum_{t=1}^T \sum_{i=1}^n \hat{\ell}_t(i)^2 \mathbf{x}_t(i) + \frac{\log n}{\varepsilon}] & \text{Theorem 1.5} \\ &\leq \varepsilon T n + \tfrac{\log n}{\varepsilon} & \text{equation (6.1)} \\ &\leq 2\sqrt{Tn \log n}. & \text{by choice of } \varepsilon \end{aligned}$$

□

We proceed to derive an algorithm for the more general setting of bandit convex optimization that attains near-optimal regret.

6.3 A Reduction from Limited Information to Full Information

In this section we derive a low regret algorithm for the general setting of bandit convex optimization. In fact, we shall describe a general technique for designing bandit algorithms, which is composed of two parts:

1. A general technique for taking an online convex optimization algorithm that uses only the gradients of the cost functions (formally defined below), and applying it to a family of vector random variables with carefully chosen properties.

2. Designing the random variables that allow the template reduction to produce meaningful regret guarantees.

We proceed to describe the two parts of this reduction, and in the remainder of this chapter we describe two examples of using this reduction to design bandit convex optimization algorithms.

6.3.1 Part 1: using unbiased estimators

The key idea behind many of the efficient algorithms for bandit convex optimization is the following: although we cannot calculate $\nabla f_t(\mathbf{x}_t)$ explicitly, it is possible to find an *observable* random variable $\mathbf{g}_t$ that satisfies $\mathbf{E}[\mathbf{g}_t] \approx \nabla f_t(\mathbf{x}_t) = \nabla_t$. Thus, $\mathbf{g}_t$ can be seen as an estimator of the gradient. By substituting $\mathbf{g}_t$ for ∇_t in an OCO algorithm, we will show that many times it retains its sublinear regret bound.

Formally, the family of regret minimization algorithms for which this reduction works is captured in the following definition.

Definition 6.4. *(**first order OCO Algorithm**) Let $\mathcal{A}$ be an OCO (deterministic) algorithm receiving an arbitrary sequence of differential loss functions $f_1, \ldots, f_T$, and producing decisions $\mathbf{x}_1 \leftarrow \mathcal{A}(\emptyset), \mathbf{x}_t \leftarrow \mathcal{A}(f_1, \ldots, f_{t-1})$. $\mathcal{A}$ is called a* first order online algorithm *if the following holds:*

- *The family of loss functions $\mathcal{F}$ is closed under addition of linear functions: if $f \in \mathcal{F}$ and $\mathbf{u} \in \mathbb{R}^n$ then $f + \mathbf{u}^\top \mathbf{x} \in \mathcal{F}$.*

- *Let $\hat{f}_t$ be the linear function $\hat{f}_t(\mathbf{x}) = \nabla f_t(\mathbf{x}_t)^\top \mathbf{x}$, then for every iteration $t \in [T]$:*

$$\mathcal{A}(f_1, \ldots, f_{t-1}) = \mathcal{A}(\hat{f}_1, ..., \hat{f}_{t-1})$$

We can now consider a formal reduction from any first order online algorithm to a bandit convex optimization algorithm as follows.

Perhaps surprisingly, under very mild conditions the reduction above guarantees the same regret bounds as the original first order algorithm up to the magnitude of the estimated gradients. This is captured in the following lemma.

Algorithm 22 Reduction to bandit feedback.

Input: convex set $\mathcal{K} \subset \mathbb{R}^n$, first order online algorithm $\mathcal{A}$.
Let $\mathbf{x}_1 = \mathcal{A}(\emptyset)$.
for $t = 1$ to T **do**
    Generate distribution $\mathcal{D}_t$, sample $\mathbf{y}_t \sim \mathcal{D}_t$ with $\mathbf{E}[\mathbf{y}_t] = \mathbf{x}_t$.
    Play $\mathbf{y}_t$.
    Observe $f_t(\mathbf{y}_t)$, generate $\mathbf{g}_t$ with $\mathbf{E}[\mathbf{g}_t] = \nabla f_t(\mathbf{x}_t)$.
    Let $\mathbf{x}_{t+1} = \mathcal{A}(\mathbf{g}_1, ..., \mathbf{g}_t)$.
end for

Lemma 6.5. *Let* $\mathbf{u}$ *be a* fixed *point in* $\mathcal{K}$*. Let* $f_1, \ldots, f_T : \mathcal{K} \to \mathbb{R}$ *be a sequence of differentiable functions. Let* $\mathcal{A}$ *be a first order online algorithm that ensures a regret bound of the form* $\text{Regret}_T(\mathcal{A}) \leq B_{\mathcal{A}}(\nabla f_1(\mathbf{x}_1), \ldots, \nabla f_T(\mathbf{x}_T))$ *in the full information setting. Define the points* $\{\mathbf{x}_t\}$ *as:* $\mathbf{x}_1 \leftarrow \mathcal{A}(\emptyset)$, $\mathbf{x}_t \leftarrow \mathcal{A}(\mathbf{g}_1, \ldots, \mathbf{g}_{t-1})$ *where each* $\mathbf{g}_t$ *is a vector valued random variable such that:*

$$\mathbf{E}[\mathbf{g}_t \big| \mathbf{x}_1, f_1, \ldots, \mathbf{x}_t, f_t] = \nabla f_t(\mathbf{x}_t).$$

Then the following holds for all $\mathbf{u} \in \mathcal{K}$*:*

$$\mathbf{E}[\sum_{t=1}^{T} f_t(\mathbf{x}_t)] - \sum_{t=1}^{T} f_t(\mathbf{u}) \leq \mathbf{E}[B_{\mathcal{A}}(\mathbf{g}_1, \ldots, \mathbf{g}_T)].$$

Proof. Define the functions $h_t : \mathcal{K} \to \mathbb{R}$ as follows:

$$h_t(\mathbf{x}) = f_t(\mathbf{x}) + \boldsymbol{\xi}_t^\top \mathbf{x}, \text{ where } \boldsymbol{\xi}_t = \mathbf{g}_t - \nabla f_t(\mathbf{x}_t).$$

Note that

$$\nabla h_t(\mathbf{x}_t) = \nabla f_t(\mathbf{x}_t) + \mathbf{g}_t - \nabla f_t(\mathbf{x}_t) = \mathbf{g}_t.$$

Therefore, deterministically applying a first order method $\mathcal{A}$ on the random functions h_t is equivalent to applying $\mathcal{A}$ on a stochastic first order approximation of the deterministic functions f_t. Thus by the full-information regret bound of $\mathcal{A}$ we have:

$$\sum_{t=1}^{T} h_t(\mathbf{x}_t) - \sum_{t=1}^{T} h_t(\mathbf{u}) \leq B_{\mathcal{A}}(\mathbf{g}_1, \ldots, \mathbf{g}_T). \tag{6.2}$$

Also note that:

$$\begin{aligned}\mathbf{E}[h_t(\mathbf{x}_t)] &= \mathbf{E}[f_t(\mathbf{x}_t)] + \mathbf{E}[\boldsymbol{\xi}_t^\top \mathbf{x}_t] \\ &= \mathbf{E}[f_t(\mathbf{x}_t)] + \mathbf{E}[\mathbf{E}[\boldsymbol{\xi}_t^\top \mathbf{x}_t | \mathbf{x}_1, f_1, \ldots, \mathbf{x}_t, f_t]] \\ &= \mathbf{E}[f_t(\mathbf{x}_t)] + \mathbf{E}[\mathbf{E}[\boldsymbol{\xi}_t | \mathbf{x}_1, f_1, \ldots, \mathbf{x}_t, f_t]^\top \mathbf{x}_t] \\ &= \mathbf{E}[f_t(\mathbf{x}_t)].\end{aligned}$$

where we used $\mathbf{E}[\boldsymbol{\xi}_t | \mathbf{x}_1, f_1, \ldots, \mathbf{x}_t, f_t] = 0$. Similarly, since $\mathbf{u} \in \mathcal{K}$ is fixed we have that $\mathbf{E}[h_t(\mathbf{u})] = f_t(\mathbf{u})$. The lemma follows from taking the expectation of Equation (6.2). □

6.3.2 Part 2: point-wise gradient estimators

In the preceding part we have described how to convert a first order algorithm for OCO to one that uses bandit information, using specially tailored random variables. We now describe how to create these vector random variables.

Although we cannot calculate $\nabla f_t(\mathbf{x}_t)$ explicitly, it is possible to find an *observable* random variable $\mathbf{g}_t$ that satisfies $\mathbf{E}[\mathbf{g}_t] \approx \nabla f_t$, and serves as an estimator of the gradient.

The question is how to find an appropriate $\mathbf{g}_t$, and in order to answer it we begin with an example in a 1-dimensional case.

Example 6.6. *A 1-dimensional gradient estimate*
Recall the definition of the derivative:

$$f'(x) = \lim_{\delta \to 0} \frac{f(x+\delta) - f(x-\delta)}{2\delta}.$$

The above shows that for a 1-dimensional derivative, two evaluations of f are required. Since in our problem we can perform only one evaluation, let us define $g(x)$ as follows:

$$g(x) = \begin{cases} \frac{f(x+\delta)}{\delta}, & \text{with probability } \frac{1}{2} \\ \\ -\frac{f(x-\delta)}{\delta}, & \text{with probability } \frac{1}{2} \end{cases}. \tag{6.3}$$

It is clear that

$$\mathbf{E}[g(x)] = \frac{f(x+\delta) - f(x-\delta)}{2\delta}.$$

Thus, **in expectation**, *for small δ, $g(x)$ approximates $f'(x)$.*

The sphere sampling estimator

We will now show how the gradient estimator (6.3) can be extended to the multidimensional case. Let $\mathbf{x} \in \mathbb{R}^n$, and let B_δ and S_δ denote the n-dimensional ball and sphere with radius δ :

$$B_\delta = \{\mathbf{x} | \, \|\mathbf{x}\| \leq \delta\},$$

$$S_\delta = \{\mathbf{x} | \, \|\mathbf{x}\| = \delta\}.$$

We define $\hat{f}(\mathbf{x}) = \hat{f}_\delta(\mathbf{x})$ to be a δ-smoothed version of $f(\mathbf{x})$:

$$\hat{f}_\delta(\mathbf{x}) = \mathop{\mathbf{E}}_{\mathbf{v}\in\mathbb{B}} [f(\mathbf{x} + \delta\mathbf{v})], \tag{6.4}$$

where $\mathbf{v}$ is drawn from a uniform distribution over the unit ball. This construction is very similar to the one used in Lemma 2.8 in context of convergence analysis for convex optimization. However, our goal here is very different.

Note that when f is linear, we have $\hat{f}_\delta(\mathbf{x}) = f(\mathbf{x})$. We shall address the case in which f is indeed linear as a special case, and show how to estimate the gradient of $\hat{f}(\mathbf{x})$, which, under the assumption, is also the gradient of $f(\mathbf{x})$. The following lemma shows a simple relation between the gradient $\nabla\hat{f}_\delta$ and a uniformly drawn unit vector.

Lemma 6.7. *Fix $\delta > 0$. Let $\hat{f}_\delta(\mathbf{x})$ be as defined in* (6.4)*, and let $\mathbf{u}$ be a uniformly drawn unit vector $\mathbf{u} \sim \mathbb{S}$. Then*

$$\mathop{\mathbf{E}}_{\mathbf{u}\in\mathbb{S}} [f(\mathbf{x} + \delta\mathbf{u})\,\mathbf{u}] = \frac{\delta}{n}\nabla\hat{f}_\delta(\mathbf{x}).$$

Proof. Using Stokes' theorem from calculus, we have

$$\nabla \int_{B_\delta} f(\mathbf{x} + \mathbf{v})\, d\mathbf{v} = \int_{S_\delta} f(\mathbf{x} + \mathbf{u}) \frac{\mathbf{u}}{\|\mathbf{u}\|} d\mathbf{u}. \tag{6.5}$$

From (6.4), and by definition of expectation, we have

$$\hat{f}_\delta(\mathbf{x}) = \frac{\int_{B_\delta} f(\mathbf{x} + \mathbf{v})\, d\mathbf{v}}{\mathrm{vol}(B_\delta)}. \tag{6.6}$$

where $\mathrm{vol}(B_\delta)$ is the volume of an n-dimensional ball of radius δ. Similarly,

$$\mathop{\mathbf{E}}_{\mathbf{u}\in S} [f(\mathbf{x} + \delta\mathbf{u})\,\mathbf{u}] = \frac{\int_{S_\delta} f(\mathbf{x} + \mathbf{u}) \frac{\mathbf{u}}{\|\mathbf{u}\|} du}{\mathrm{vol}(S_\delta)}. \tag{6.7}$$

Combining (6.4), (6.5), (6.6), and (6.7), and the fact that the ratio of the volume of a ball in n dimensions and the sphere of dimension $n-1$ is $\mathrm{vol}_n B_\delta / \mathrm{vol}_{n-1} S_\delta = \delta/n$ gives the desired result. □

Under the assumption that f is linear, Lemma 6.7 suggests a simple estimator for the gradient ∇f. Draw a random unit vector $\mathbf{u}$, and let $g(\mathbf{x}) = \frac{n}{\delta} f(\mathbf{x} + \delta \mathbf{u})\,\mathbf{u}$.

The ellipsoidal sampling estimator

The sphere estimator above is at times difficult to use: when the center of the sphere is very close to the boundary of the decision set only a very small sphere can fit completely inside. This results in a gradient estimator with large variance.

In such cases, it is useful to consider ellipsoids rather than spheres. Luckily, the generalisation to ellipsoidal sampling for gradient estimation is a simple corollary of our derivation above:

Corollary 6.8. *Consider a continuous function $f : \mathbb{R}^n \to \mathbb{R}$, an invertible matrix $A \in \mathbb{R}^{n\times n}$, and let $\mathbf{v} \sim \mathbb{B}^n$ and $\mathbf{u} \sim \mathbb{S}^n$. Define the smoothed version of f with respect to A:*

$$\hat{f}(\mathbf{x}) = \mathbf{E}[f(\mathbf{x} + A\mathbf{v})].$$

Then the following holds:

$$\nabla \hat{f}(\mathbf{x}) = n\,\mathbf{E}[f(\mathbf{x} + A\mathbf{u})A^{-1}\mathbf{u}].$$

Proof. Let $g(\mathbf{x}) = f(A\mathbf{x})$, and $\hat{g}(\mathbf{x}) = \mathbf{E}_{\mathbf{v}\in\mathbb{B}}[g(\mathbf{x}+\mathbf{v})]$.

$$\begin{aligned} n\,\mathbf{E}[f(\mathbf{x}+A\mathbf{u})A^{-1}\mathbf{u}] &= nA^{-1}\,\mathbf{E}[f(\mathbf{x}+A\mathbf{u})\mathbf{u}] \\ &= nA^{-1}\,\mathbf{E}[g(A^{-1}\mathbf{x}+\mathbf{u})\mathbf{u}] \\ &= A^{-1}\nabla\hat{g}(A^{-1}\mathbf{x}) \qquad \text{Lemma 6.7} \\ &= A^{-1}A\nabla\hat{f}(\mathbf{x}) = \nabla\hat{f}(\mathbf{x}). \end{aligned}$$

□

6.4 Online Gradient Descent without a Gradient

The simplest and historically earliest application of the BCO-to-OCO reduction outlined before is the application of the online gradient descent algorithm to the bandit setting. The FKM algorithm (named after its inventors, see bibliographic section) is outlined in algorithm 23.

For simplicity, we assume that the set $\mathcal{K}$ contains the unit ball centered at the zero vector, denoted $\mathbf{0}$. Denote $\mathcal{K}_\delta = \{\mathbf{x} \mid \frac{1}{1-\delta}\mathbf{x} \in \mathcal{K}\}$. It is left as an exercise to show that $\mathcal{K}_\delta$ is convex for any $0 < \delta < 1$ and that all balls of radius δ around points in $\mathcal{K}_\delta$ are contained in $\mathcal{K}$.

We also assume for simplicity that the adversarially chosen cost functions are bounded by one over $\mathcal{K}$, i.e., that $|\mathbf{f}_t(\mathbf{x})| \leq 1$ for all $\mathbf{x} \in \mathcal{K}$.

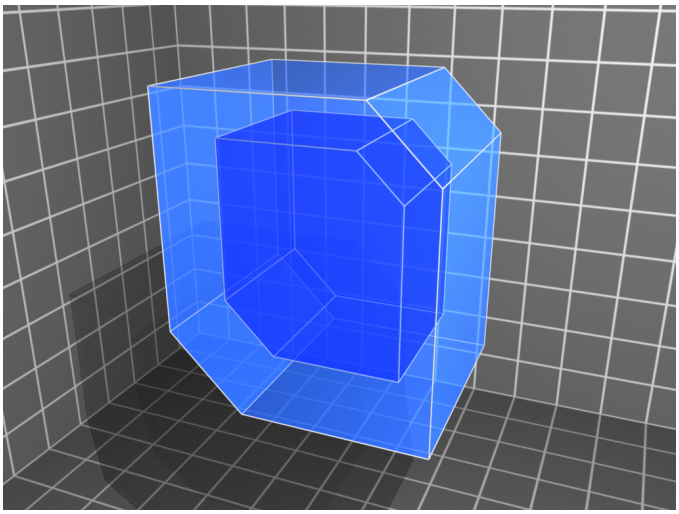

Figure 6.1: The Minkowski set $\mathcal{K}_\delta$

Algorithm 23 FKM Algorithm

1: Input: decision set $\mathcal{K}$ containing $\mathbf{0}$, set $\mathbf{x}_1 = \mathbf{0}$, parameters δ, η.
2: **for** $t = 1$ to T **do**
3: Draw $\mathbf{u}_t \in \mathbb{S}_1$ uniformly at random, set $\mathbf{y}_t = \mathbf{x}_t + \delta \mathbf{u}_t$.
4: Play $\mathbf{y}_t$, observe and incur loss $f_t(\mathbf{y}_t)$. Let $\mathbf{g}_t = \frac{n}{\delta} f_t(\mathbf{y}_t)\, \mathbf{u}_t$.
5: Update $\mathbf{x}_{t+1} = \prod_{\mathcal{K}_\delta} [\mathbf{x}_t - \eta \mathbf{g}_t]$.
6: **end for**

The FKM algorithm is an instantiation of the generic reduction from bandit convex optimization to online convex optimization with spherical gradient estimators over the set $\mathcal{K}_\delta$. It iteratively projects onto $\mathcal{K}_\delta$, in order to have enough space for spherical gradient estimation. This degrades its performance by a controlled quantity. Its regret is bounded as follows.

Theorem 6.9. *Algorithm 23 with parameters* $\eta = \frac{D}{nT^{3/4}}, \delta = \frac{1}{T^{1/4}}$ *guarantees the following expected regret bound*

$$\sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{y}_t)] - \min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}) \leq 9nDGT^{3/4} = O(T^{3/4}).$$

Proof. Recall our notation of $\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x})$. Denote

$$\mathbf{x}_\delta^\star = \prod_{\mathcal{K}_\delta}(\mathbf{x}^\star).$$

Then by properties of projections we have $\|\mathbf{x}_\delta^\star - \mathbf{x}^\star\| \leq \delta D$, where D is the diameter of $\mathcal{K}$. Thus, assuming that the cost functions $\{f_t\}$ are G-Lipschitz, we have

$$\sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{y}_t)] - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \leq \sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{y}_t)] - \sum_{t=1}^{T} f_t(\mathbf{x}_\delta^\star) + \delta TGD. \tag{6.8}$$

Denote $\hat{f}_t = \hat{f}_{\delta,t} = \mathbf{E}_{\mathbf{u}\sim\mathbb{B}}[f(\mathbf{x} + \delta\mathbf{u})]$ for shorthand. We can now bound

the regret by

$$
\begin{aligned}
&\sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{y}_t)] - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \\
&\leq \sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{x}_t)] - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) + \delta DGT && f_t \text{ is } G\text{-Lipschitz} \\
&\leq \sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{x}_t)] - \sum_{t=1}^{T} f_t(\mathbf{x}_\delta^\star) + 2\delta DGT && \text{Inequality (6.8)} \\
&\leq \sum_{t=1}^{T} \mathbf{E}[\hat{f}_t(\mathbf{x}_t)] - \sum_{t=1}^{T} \hat{f}_t(\mathbf{x}_\delta^\star) + 4\delta DGT && \text{Lemma 2.8} \\
&\leq \text{Regret}_{OGD}(\mathbf{g}_1, ..., \mathbf{g}_T) + 4\delta DGT && \text{Lemma 6.5} \\
&\leq \eta \sum_{t=1}^{T} \|\mathbf{g}_t\|^2 + \frac{D^2}{\eta} + 4\delta DGT && \text{OGD regret, Theorem 3.1} \\
&\leq \eta \frac{n^2}{\delta^2} T + \frac{D^2}{\eta} + 4\delta DGT && |\mathbf{f}_t(\mathbf{x})| \leq 1 \\
&\leq 9nDGT^{3/4}. && \eta = \frac{D}{nT^{3/4}}, \delta = \frac{1}{T^{1/4}}
\end{aligned}
$$

□

6.5 * Optimal Regret Algorithms for Bandit Linear Optimization

A special case of BCO that is of considerable interest is BLO—Bandit Linear Optimization. This setting has linear cost functions, and captures the network routing and ad placement examples discussed in the beginning of this chapter, as well as the non-stochastic MAB problem.

In this section we give near-optimal regret bounds for BLO using techniques from interior point methods for convex optimization.

The generic OGD method of the previous section suffers from three pitfalls:

1. The gradient estimators are biased, and estimate the gradient of a smoothed version of the real cost function.

2. The gradient estimators require enough "wiggle room" and are thus ill-defined on the boundary of the decision set.

3. The gradient estimates have potentially large magnitude, proportional to the distance from the boundary.

Fortunately, the first issue is non-existent for linear functions - the gradient estimators turn out to be unbiased for linear functions. In the notation of the previous chapters, we have for linear functions:

$$\hat{f}_\delta(\mathbf{x}) = \mathop{\mathbf{E}}_{\mathbf{v}\sim\mathbb{B}}[f(\mathbf{x}+\delta\mathbf{v})] = f(\mathbf{x}).$$

Thus, Lemma 6.7 gives us a stronger guarantee:

$$\mathop{\mathbf{E}}_{\mathbf{u}\in\mathbb{S}}\left[f\left(\mathbf{x}+\delta\mathbf{u}\right)\mathbf{u}\right] = \frac{\delta}{n}\nabla\hat{f}_\delta\left(\mathbf{x}\right) = \frac{\delta}{n}\nabla f(\mathbf{x}).$$

To resolve the second and third issues we use self-concordant barrier functions, a rather advanced technique from interior point methods for convex optimization.

6.5.1 Self-concordant barriers

Self-concordant barrier functions were devised in the context of interior point methods for optimization as a way of ensuring that the Newton method converges in polynomial time over bounded convex sets. In this brief introduction we survey some of their beautiful properties that will allow us to derive an optimal regret algorithm for BLO.

Definition 6.10. *Let $\mathcal{K} \in \mathbb{R}^n$ be a convex set with a nonempty interior $int(\mathcal{K})$. A function $\mathcal{R} : int(\mathcal{K}) \to \mathbb{R}$ is called ν-self-concordant if:*

1. *$\mathcal{R}$ is three times continuously differentiable and convex, and approaches infinity along any sequence of points approaching the boundary of $\mathcal{K}$.*

2. *For every $\mathbf{h} \in \mathbb{R}^n$ and $\mathbf{x} \in int(\mathcal{K})$ the following holds:*

$$|\nabla^3\mathcal{R}(\mathbf{x})[\mathbf{h},\mathbf{h},\mathbf{h}]| \le 2(\nabla^2\mathcal{R}(\mathbf{x})[\mathbf{h},\mathbf{h}])^{3/2},$$
$$|\nabla\mathcal{R}(\mathbf{x})[\mathbf{h}]| \le \nu^{1/2}(\nabla^2\mathcal{R}(\mathbf{x})[\mathbf{h},\mathbf{h}])^{1/2}$$

where the third order differential is defined as:

$$\nabla^3\mathcal{R}(\mathbf{x})[\mathbf{h},\mathbf{h},\mathbf{h}] \stackrel{\text{def}}{=} \frac{\partial^3}{\partial t_1\partial t_2\partial t_3}\mathcal{R}(\mathbf{x}+t_1\mathbf{h}+t_2\mathbf{h}+t_3\mathbf{h})\Big|_{t_1=t_2=t_3=0}$$

The Hessian of a self-concordant barrier induces a local norm at every $\mathbf{x} \in \text{int}(\mathcal{K})$, we denote this norm by $||\cdot||_{\mathbf{x}}$ and its dual by $||\cdot||_{\mathbf{x}}^*$, which are defined $\forall \mathbf{h} \in \mathbb{R}^n$ by

$$\|\mathbf{h}\|_{\mathbf{x}} = \sqrt{\mathbf{h}^\top \nabla^2 \mathcal{R}(\mathbf{x}) \mathbf{h}}, \qquad \|\mathbf{h}\|_{\mathbf{x}}^* = \sqrt{\mathbf{h}^\top (\nabla^2 \mathcal{R}(\mathbf{x}))^{-1} \mathbf{h}}.$$

We assume that $\nabla^2 \mathcal{R}(\mathbf{x})$ always has full rank. In BCO applications this is easy to ensure by adding a fictitious quadratic function to the barrier, which does not affect the overall regret by more than a constant.

Let $\mathcal{R}$ be a self-concordant barrier and $\mathbf{x} \in \text{int}(\mathcal{K})$. The *Dikin ellipsoid* is

$$\mathcal{E}_1(\mathbf{x}) := \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}} \leq 1\},$$

i.e., the $\|\cdot\|_{\mathbf{x}}$-unit ball centered around $\mathbf{x}$, is completely contained in $\mathcal{K}$.

In our next analysis we will need to bound $\mathcal{R}(\mathbf{y}) - \mathcal{R}(\mathbf{x})$ for $\mathbf{x}, \mathbf{y} \in \text{int}(\mathcal{K})$, for which the following lemma is useful:

Lemma 6.11. *Let $\mathcal{R}$ be a ν-self concordant function over $\mathcal{K}$, then for all $\mathbf{x}, \mathbf{y} \in$ int($\mathcal{K}$):*

$$\mathcal{R}(\mathbf{y}) - \mathcal{R}(\mathbf{x}) \leq \nu \log \frac{1}{1 - \pi_{\mathbf{x}}(\mathbf{y})},$$

where $\pi_{\mathbf{x}}(\mathbf{y}) = \inf\{t \geq 0 : \mathbf{x} + t^{-1}(\mathbf{y} - \mathbf{x}) \in \mathcal{K}\}$.

The function $\pi_{\mathbf{x}}(\mathbf{y})$ is called the Minkowski function for $\mathcal{K}$, and its output is always in the interval $[0, 1]$. Moreover, as y approaches the boundary of $\mathcal{K}$ then $\pi_{\mathbf{x}}(\mathbf{y}) \to 1$.

Another important property of self-concordant functions is the relationship between a point and the optimum, and the norm of the gradient at the point, according to the local norm, as given by the following lemma.

Lemma 6.12. *Let $\mathbf{x} \in$ int($\mathcal{K}$) be such that $\|\nabla \mathcal{R}(\mathbf{x})\|_{\mathbf{x}}^* \leq \frac{1}{4}$, and let $\mathbf{x}^\star = \arg\min_{\mathbf{x} \in \mathcal{K}} \mathcal{R}(\mathbf{x})$. Then*

$$\|\mathbf{x} - \mathbf{x}^\star\|_x \leq 2 \|\nabla \mathcal{R}(\mathbf{x})\|_{\mathbf{x}}^*.$$

6.5.2 A near-optimal algorithm

We have now set up all the necessary tools to derive a near-optimal BLO algorithm, presented in algorithm 24.

Algorithm 24 SCRIBLE

1: Input: decision set $\mathcal{K}$ with self concordant barrier $\mathcal{R}$, set $\mathbf{x}_1 \in \operatorname{int}(\mathcal{K})$ such that $\nabla\mathcal{R}(\mathbf{x}_1) = 0$, parameters η, δ.
2: **for** $t = 1$ to T **do**
3: Let $\mathbf{A}_t = \left[\nabla^2\mathcal{R}(\mathbf{x}_t)\right]^{-1/2}$.
4: Pick $\mathbf{u}_t \in \mathbb{S}$ uniformly, and set $\mathbf{y}_t = \mathbf{x}_t + \mathbf{A}_t\mathbf{u}_t$.
5: Play $\mathbf{y}_t$, observe and suffer loss $f_t(\mathbf{y}_t)$. let $\mathbf{g}_t = nf_t(\mathbf{y}_t)\mathbf{A}_t^{-1}\mathbf{u}_t$.
6: Update

$$\mathbf{x}_{t+1} = \operatorname*{arg\,min}_{\mathbf{x}\in\mathcal{K}_\delta} \left\{ \eta \sum_{\tau=1}^{t} \mathbf{g}_\tau^\top \mathbf{x} + \mathcal{R}(\mathbf{x}) \right\}.$$

7: **end for**

Theorem 6.13. *For appropriate choice of η, δ, the SCRIBLE algorithm guarantees*

$$\sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{y}_t)] - \min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}) \le O\left(\sqrt{T}\log T\right).$$

Proof. First, we note that $\mathbf{y}_t \in \mathcal{K}$ never steps outside of the decision set. The reason is that $\mathbf{x}_t \in \mathcal{K}$ and $\mathbf{y}_t$ lies in the Dikin ellipsoid centered at $\mathbf{x}_t$.

Further, by Corollary 6.8, we have that

$$\mathbf{E}[\mathbf{g}_t] = \nabla\hat{f}_t(\mathbf{x}_t) = \nabla f_t(\mathbf{x}_t),$$

where the latter equality follows since f_t is linear, and thus its smoothed version is identical to itself.

A final observation is that line 24 in the algorithm is an invocation of the RFTL algorithm with the self-concordant barrier $\mathcal{R}$ serving as a regularisation function. The RFTL algorithm for linear functions is a first order OCO algorithm and thus Lemma 6.5 applies.

We can now bound the regret by

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{y}_t)] - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \\
&\leq \sum_{t=1}^{T} \mathbf{E}[\hat{f}_t(\mathbf{x}_t)] - \sum_{t=1}^{T} \hat{f}_t(\mathbf{x}^\star) && \hat{f}_t = f_t,\ \mathbf{E}[\mathbf{y}_t] = \mathbf{x}_t \\
&\leq \text{Regret}_{RFTL}(\mathbf{g}_1, ..., \mathbf{g}_T) && \text{Lemma 6.5} \\
&\leq \sum_{t=1}^{T} \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{\mathcal{R}(\mathbf{x}^\star) - \mathcal{R}(\mathbf{x}_1)}{\eta} && \text{Lemma 5.3} \\
&\leq \sum_{t=1}^{T} \|\mathbf{g}_t\|_t^* \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_t + \frac{\mathcal{R}(\mathbf{x}^\star) - \mathcal{R}(\mathbf{x}_1)}{\eta}. && \text{Cauchy-Schwarz}
\end{aligned}$$

Here we use our notation from the previous chapter for the local norm $\|\mathbf{h}\|_t = \|\mathbf{h}\|_{\mathbf{x}_t} = \sqrt{\mathbf{h}^\top \nabla^2 \mathcal{R}(\mathbf{x}_t)\mathbf{h}}$.

To bound the last expression, we use Lemma 6.12, and the definition of $\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{K}} \Phi_t(\mathbf{x})$ where $\Phi_t(\mathbf{x}) = \eta \sum_{\tau=1}^{t} \mathbf{g}_\tau^\top \mathbf{x} + \mathcal{R}(\mathbf{x})$ is a self-concordant barrier. Thus,

$$\|\mathbf{x}_t - \mathbf{x}_{t+1}\|_t \leq 2\|\nabla\Phi_t(\mathbf{x}_t)\|_t^* = 2\|\nabla\Phi_{t-1}(\mathbf{x}_t) + \eta\mathbf{g}_t\|_t^* = 2\eta\|\mathbf{g}_t\|_t^*,$$

since $\nabla\Phi_{t-1}(\mathbf{x}_t) = 0$ by definition of $\mathbf{x}_t$. Recall that to use Lemma 6.12, we need $\|\nabla\Phi_t(\mathbf{x}_t)\|_t^* = \eta\|\mathbf{g}_t\|_t^* \leq \frac{1}{4}$, which is true by choice of η and since

$$\|\mathbf{g}_t\|_t^{*\,2} \leq n^2 \mathbf{u}_t^T \mathbf{A}_t^{-T} \nabla^{-2}\mathcal{R}(\mathbf{x}_t)\mathbf{A}_t^{-1}\mathbf{u}_t \leq n^2.$$

Thus,

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{y}_t)] - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \leq 2\eta \sum_{t=1}^{T} \|\mathbf{g}_t\|_t^{*\,2} + \frac{\mathcal{R}(\mathbf{x}^\star) - \mathcal{R}(\mathbf{x}_1)}{\eta} \\
&\leq 2\eta n^2 T + \frac{\mathcal{R}(\mathbf{x}^\star) - \mathcal{R}(\mathbf{x}_1)}{\eta}.
\end{aligned}$$

It remains to bound the Bregman divergence with respect to $\mathbf{x}^\star$, for which we use a similar technique as in the analysis of algorithm 23, and bound the regret with respect to $\mathbf{x}_\delta^\star$, which is the projection of $\mathbf{x}^\star$ onto $\mathcal{K}_\delta$. Using

equation (6.8), we can bound the overall regret by:

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{y}_t)] - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \\
&\leq \sum_{t=1}^{T} \mathbf{E}[f_t(\mathbf{y}_t)] - \sum_{t=1}^{T} f_t(\mathbf{x}_\delta^*) + \delta TGD && \text{equation (6.8)} \\
&= 2\eta n^2 T + \frac{\mathcal{R}(\mathbf{x}_\delta^\star) - \mathcal{R}(\mathbf{x}_1)}{\eta} + \delta TGD && \text{above derivation} \\
&\leq 2\eta n^2 T + \frac{\nu \log \frac{1}{1-\pi_{\mathbf{x}_1}(\mathbf{x}_\delta^\star)}}{\eta} + \delta TGD && \text{Lemma 6.11} \\
&\leq 2\eta n^2 T + \frac{\nu \log \frac{1}{\delta}}{\eta} + \delta TGD && \mathbf{x}_\delta^\star \in \mathcal{K}_\delta.
\end{aligned}$$

Taking $\eta = O(\frac{1}{\sqrt{T}})$ and $\delta = O(\frac{1}{T})$, the above bound implies our theorem.

□

6.6 Bibliographic Remarks

The Multi-Armed Bandit problem has history going back more than fifty years to the work of Robbins [1952], see the survey of Bubeck and Cesa-Bianchi [2012] for a much more detailed history. The non-stochastic MAB problem and the EXP3 algorithm, as well as tight lower bounds were given in the seminal paper of Auer et al. [2003]. The logarithmic gap in attainable regret for non-stochastic MAB was resolved in [Audibert and Bubeck, 2009].

Bandit Convex Optimization for the special case of linear cost functions and the flow polytope, was introduced and studied by Awerbuch and Kleinberg [2008] in the context of online routing. The full generality BCO setting was introduced by Flaxman et al. [2005], who gave the first efficient and low-regret algorithm for BCO. Tight bounds for BCO were obtained by Bubeck et al. [2015] for the one dimensional case, via an inefficient algorithm by Hazan and Li [2016], and finally with a polynomial time algorithm in Bubeck et al. [2017].

The special case in which the cost functions are linear, called Bandit Linear Optimization, received significant attention. Dani et al. [2008] gave an optimal regret algorithm up to constants depending on the dimension. Abernethy et al. [2008] gave an efficient algorithm and introduced self-concordant barriers to the bandit setting. Self-concordant barrier functions were devised in the context of polynomial-time algorithms for convex optimization in the seminal work of Nesterov and Nemirovskii [1994]. Lower bounds for regret in the bandit linear optimization setting were studied by Shamir [2015].

In this chapter we have considered the expected regret as a performance metric. Significant literature is devoted to high probability guarantees on the regret. High probability bounds for the MAB problem were given in [Auer et al., 2003], and for bandit linear optimization in [Abernethy and Rakhlin, 2009]. Other more refined metrics have been recently explored in [Dekel et al., 2012] and in the context of adaptive adversaries in [Neu et al., 2014, Yu and Mannor, 2009, Even-Dar et al., 2009, Mannor and Shimkin, 2003, Yu et al., 2009].

For a recent comprehensive text on bandit algorithms see [Lattimore and Szepesvári, 2020].

6.7 Exercises

1. Prove a lower bound on the regret of any algorithm for BCO: show that for the special case of BCO over the unit sphere, any online algorithm must incur a regret of $\Omega(\sqrt{T})$.

2. * Strengthen the above bound: show that for the special case of BLO over the d-dimensional unit simplex, with cost functions bounded in ℓ_∞ norm by one, any online algorithm must incur a regret of $\Omega(\sqrt{dT})$ as $T \to \infty$, up to logarithmic terms in T.

3. Let $\mathcal{K}$ be convex. Show that the set $\mathcal{K}_\delta$ is convex.

4. Let $\mathcal{K}$ be convex and contain the unit ball centered at zero. Show that for any point $\mathbf{x} \in \mathcal{K}_\delta$, the ball of radius δ centered at $\mathbf{x}$ is contained in $\mathcal{K}$.

5. Consider the BCO setting with H-strongly convex functions, H is known a-priori to the online learner. Show that in this case we can attain a regret bound of $\tilde{O}(T^{2/3})$.
Hint: recall that we can attain a regret bound of $O(\log T)$ in the full-information OCO with H-strongly convex functions, and recall that the notation $\tilde{O}(\cdot)$ hides constant and poly-logarithmic terms.

6. Consider the BCO setting with the following twist: at every iteration, the player is allowed to observe **two** evaluations of the function, as opposed to just one. That is, the player gives x_t, y_t, and observes $f_t(x_t), f_t(y_t)$. Regret is measured w.r.t. x_t, as usual:

$$\sum_t f_t(\mathbf{x}_t) - \min_{\mathbf{x}^\star \in \mathcal{K}} \sum_t f_t(\mathbf{x}^\star).$$

(a) Show how to construct a biased gradient estimator for f_t with arbitrary small bias and *constant* variance, that is independent of the bias.

(b) Show how to use the gradient estimator from the previous part to give an efficient algorithm for this setting that attains $O(\sqrt{T})$ regret.

Chapter 7

Projection-Free Algorithms

In many computational and learning scenarios the main bottleneck of optimization, both online and offline, is the computation of projections onto the underlying decision set (see §2.1.1). In this chapter we introduce projection-free methods for online convex optimization, that yield more efficient algorithms in these scenarios.

The motivating example throughout this chapter is the problem of matrix completion, which is a widely used and accepted model in the construction of recommendation systems. For matrix completion and related problems, projections amount to expensive linear algebraic operations and avoiding them is crucial in big data applications.

We start with a detour into classical offline convex optimization and describe the conditional gradient algorithm, also known as the Frank-Wolfe algorithm. Afterwards, we describe problems for which linear optimization can be carried out much more efficiently than projections. We conclude with an OCO algorithm that eschews projections in favor of linear optimization, in much the same flavor as its offline counterpart.

7.1 Review: Relevant Concepts from Linear Algebra

This chapter addresses rectangular matrices, which model applications such as recommendation systems naturally. Consider a matrix $X \in \mathbb{R}^{n \times m}$. A non-negative number $\sigma \in \mathbb{R}_+$ is said to be a singular value for X if there are two vectors $\mathbf{u} \in \mathbb{R}^n, \mathbf{v} \in \mathbb{R}^m$ such that

$$X^\top \mathbf{u} = \sigma \mathbf{v}, \quad X\mathbf{v} = \sigma \mathbf{u}.$$

The vectors $\mathbf{u}, \mathbf{v}$ are called the left and right singular vectors respectively. The non-zero singular values are the square roots of the eigenvalues of the matrix $XX^\top$ (and $X^\top X$). The matrix X can be written as

$$X = U\Sigma V^\top \ , \ U \in \mathbb{R}^{n\times\rho} \ , \ V^\top \in \mathbb{R}^{\rho\times m},$$

where $\rho = \min\{n, m\}$, the matrix U is an orthogonal basis of the left singular vectors of X, the matrix V is an orthogonal basis of right singular vectors, and Σ is a diagonal matrix of singular values. This form is called the singular value decomposition for X.

The number of non-zero singular values for X is called its rank, which we denote by $k \leq \rho$. The nuclear norm of X is defined as the ℓ_1 norm of its singular values, and denoted by

$$\|X\|_* = \sum_{i=1}^{\rho} \sigma_i.$$

It can be shown (see exercises) that the nuclear norm is equal to the trace of the square root of the matrix times its transpose, i.e.,

$$\|X\|_* = \mathbf{Tr}(\sqrt{X^\top X})$$

We denote by $A \bullet B$ the inner product of two matrices as vectors in $\mathbb{R}^{n\times m}$, that is

$$A \bullet B = \sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}B_{ij} = \mathbf{Tr}(AB^\top).$$

7.2 Motivation: Recommender Systems

Media recommendations have changed significantly with the advent of the Internet and rise of online media stores. The large amounts of data collected allow for efficient clustering and accurate prediction of users' preferences for a variety of media. A well-known example is the so called "Netflix challenge"—a competition of automated tools for recommendation from a large dataset of users' motion picture preferences.

One of the most successful approaches for automated recommendation systems, as proven in the Netflix competition, is matrix completion. Perhaps the simplest version of the problem can be described as follows.

The entire dataset of user-media preference pairs is thought of as a partially-observed matrix. Thus, every person is represented by a row in

the matrix, and every column represents a media item (movie). For simplicity, let us think of the observations as binary—a person either likes or dislikes a particular movie. Thus, we have a matrix $M \in \{0, 1, *\}^{n \times m}$ where n is the number of persons considered, m is the number of movies at our library, and $0/1$ and $*$ signify "dislike", "like" and "unknown" respectively:

$$M_{ij} = \begin{cases} 0, & \text{person } i \text{ dislikes movie } j \\ 1, & \text{person } i \text{ likes movie } j \\ *, & \text{preference unknown} \end{cases}.$$

The natural goal is to complete the matrix, i.e., correctly assign 0 or 1 to the unknown entries. As defined so far, the problem is ill-posed, since any completion would be equally good (or bad), and no restrictions have been placed on the completions.

The common restriction on completions is that the "true" matrix has low rank. Recall that a matrix $X \in \mathbb{R}^{n \times m}$ has rank $k < \rho = \min\{n, m\}$ if and only if it can be written as

$$X = UV \ , \ U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times m}.$$

The intuitive interpretation of this property is that each entry in M can be explained by only k numbers. In matrix completion this means, intuitively, that there are only k factors that determine a persons preference over movies, such as genre, director, actors and so on.

Now the simplistic matrix completion problem can be well-formulated as in the following mathematical program. Denote by $\|\cdot\|_{ob}$ the Euclidean norm only on the observed (non starred) entries of M, i.e.,

$$\|X\|_{ob}^2 = \sum_{M_{ij} \neq *} X_{ij}^2.$$

The mathematical program for matrix completion is given by

$$\begin{aligned} \min_{X \in \mathbb{R}^{n \times m}} \ & \frac{1}{2}\|X - M\|_{ob}^2 \\ \text{s.t.} \quad & \text{rank}(X) \leq k. \end{aligned}$$

Since the constraint over the rank of a matrix is non-convex, it is standard to consider a relaxation that replaces the rank constraint by the nuclear norm. It is known that the nuclear norm is a lower bound on the matrix

rank if the singular values are bounded by one (see exercises). Thus, we arrive at the following convex program for matrix completion:

$$\begin{aligned} &\min_{X \in \mathbb{R}^{n \times m}} \frac{1}{2} \|X - M\|_{ob}^2 \\ &\text{s.t.} \quad \|X\|_* \leq k. \end{aligned} \tag{7.1}$$

We consider algorithms to solve this convex optimization problem next.

7.3 The Conditional Gradient Method

In this section we return to the basics of convex optimization—minimization of a convex function over a convex domain as studied in chapter 2.

The conditional gradient (CG) method, or Frank-Wolfe algorithm, is a simple algorithm for minimizing a smooth convex function f over a convex set $\mathcal{K} \subseteq \mathbb{R}^n$. The appeal of the method is that it is a first order interior point method - the iterates always lie inside the convex set, and thus no projections are needed, and the update step on each iteration simply requires to minimize a linear objective over the set. The basic method is given in algorithm 25.

Algorithm 25 Conditional gradient

1: Input: step sizes $\{\eta_t \in (0,1],\ t \in [T]\}$, initial point $\mathbf{x}_1 \in \mathcal{K}$.
2: **for** $t = 1$ to T **do**
3: $\quad \mathbf{v}_t \leftarrow \arg\min_{\mathbf{x} \in \mathcal{K}} \left\{ \mathbf{x}^\top \nabla f(\mathbf{x}_t) \right\}$.
4: $\quad \mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \eta_t (\mathbf{v}_t - \mathbf{x}_t)$.
5: **end for**

Note that in the CG method, the update to the iterate $\mathbf{x}_t$ may be not be in the direction of the gradient, as $\mathbf{v}_t$ is the result of a linear optimization procedure in the direction of the negative gradient. This is depicted in figure 7.1.

The following theorem gives an essentially tight performance guarantee of this algorithm over smooth functions. Recall our notation from chapter 2: $\mathbf{x}^\star$ denotes the global minimizer of f over $\mathcal{K}$, D denotes the diameter of the set $\mathcal{K}$, and $h_t = f(\mathbf{x}_t) - f(\mathbf{x}^\star)$ denotes the suboptimality of the objective value in iteration t.

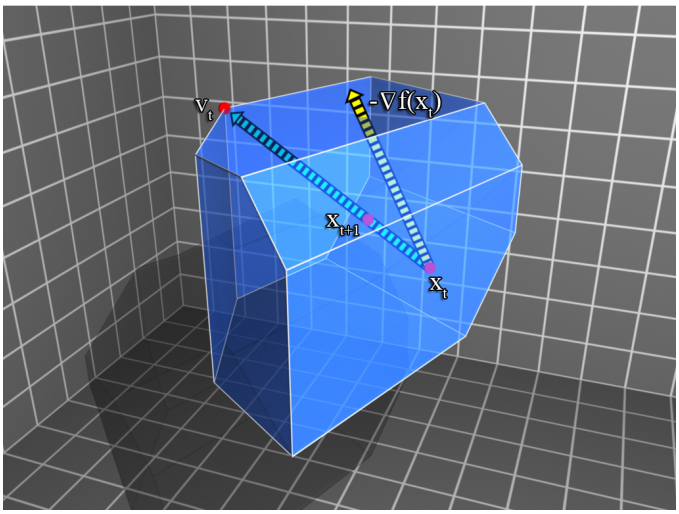

Figure 7.1: Direction of progression of the CG algorithm

Theorem 7.1. *The CG algorithm applied to β-smooth functions with step sizes $\eta_t = \min\{1, \frac{2}{t}\}$ attains the following convergence guarantee*

$$h_t \leq \frac{2\beta D^2}{t}$$

Proof. As done before in this manuscript, we denote $\nabla_t = \nabla f(\mathbf{x}_t)$. For any set of step sizes, we have

$$\begin{aligned}
&f(\mathbf{x}_{t+1}) - f(\mathbf{x}^\star) = f(\mathbf{x}_t + \eta_t(\mathbf{v}_t - \mathbf{x}_t)) - f(\mathbf{x}^\star) \\
&\leq f(\mathbf{x}_t) - f(\mathbf{x}^\star) + \eta_t(\mathbf{v}_t - \mathbf{x}_t)^\top \nabla_t + \eta_t^2 \frac{\beta}{2}\|\mathbf{v}_t - \mathbf{x}_t\|^2 && \text{smoothness} \\
&\leq f(\mathbf{x}_t) - f(\mathbf{x}^\star) + \eta_t(\mathbf{x}^\star - \mathbf{x}_t)^\top \nabla_t + \eta_t^2 \frac{\beta}{2}\|\mathbf{v}_t - \mathbf{x}_t\|^2 && \mathbf{v}_t \text{ choice} \\
&\leq f(\mathbf{x}_t) - f(\mathbf{x}^\star) + \eta_t(f(\mathbf{x}^\star) - f(\mathbf{x}_t)) + \eta_t^2 \frac{\beta}{2}\|\mathbf{v}_t - \mathbf{x}_t\|^2 && \text{convexity} \\
&\leq (1 - \eta_t)(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) + \frac{\eta_t^2 \beta}{2} D^2. && (7.2)
\end{aligned}$$

We reached the recursion $h_{t+1} \leq (1 - \eta_t)h_t + \eta_t^2 \frac{\beta D^2}{2}$, and by Lemma 7.2 we obtain,

$$h_t \leq \frac{2\beta D^2}{t}.$$

□

Lemma 7.2. *Let $\{h_t\}$ be a sequence that satisfies the recurence*

$$h_{t+1} \leq h_t(1-\eta_t) + \eta_t^2 c.$$

Then taking $\eta_t = \min\{1, \frac{2}{t}\}$ implies

$$h_t \leq \frac{4c}{t}.$$

Proof. This is proved by induction on t.

Induction base. For $t = 1$, we have

$$h_2 \leq h_1(1-\eta_1) + \eta_1^2 c = c \leq 4c.$$

Induction step.

$$\begin{aligned} h_{t+1} &\leq (1-\eta_t)h_t + \eta_t^2 c \\ &\leq \left(1 - \frac{2}{t}\right)\frac{4c}{t} + \frac{4c}{t^2} && \text{induction hypothesis} \\ &= \frac{4c}{t}\left(1 - \frac{1}{t}\right) \\ &\leq \frac{4c}{t} \cdot \frac{t}{t+1} && \tfrac{t-1}{t} \leq \tfrac{t}{t+1} \\ &= \frac{4c}{t+1}. \end{aligned}$$

□

7.3.1 Example: matrix completion via CG

As an example of an application for the conditional gradient algorithm, recall the mathematical program given by (7.1). The gradient of the objective function at point X^t is

$$\nabla f(X^t) = (X^t - M)_{ob} = \begin{cases} X^t_{ij} - M_{ij}, & (i,j) \in OB \\ 0, & \text{otherwise} \end{cases}. \tag{7.3}$$

Over the set of bounded-nuclear norm matrices, the linear optimization of line 3 in algorithm 25 becomes,

$$\begin{aligned} &\min X \bullet \nabla_t \ , \quad \nabla_t = \nabla f(X_t) \\ &\text{s.t. } \|X\|_* \leq k. \end{aligned}$$

For simplicity, let's consider square symmetric matrices, for which the nuclear norm is equivalent to the trace norm, and the above optimization problem becomes

$$\begin{aligned} &\min X \bullet \nabla_t \\ &\text{s.t. } \mathbf{Tr}(X) \leq k. \end{aligned}$$

It can be shown that this program is equivalent to the following (see exercises):

$$\begin{aligned} &\min_{\mathbf{x} \in \mathbb{R}^n} \mathbf{x}^\top \nabla_t \mathbf{x} \\ &\text{s.t. } \|\mathbf{x}\|_2^2 \leq k. \end{aligned}$$

Hence, this is an eigenvector computation in disguise! Computing the largest eigenvector of a matrix takes linear time via the power method, which also applies more generally to computing the largest singular value of rectangular matrices. With this, step 3 of algorithm 25, which amounts to mathematical program (7.1), becomes computing $v_{\max}(-\nabla f(X^t))$, the largest eigenvector of $-\nabla f(X^t)$. Algorithm 25 takes on the modified form described in Algorithm 26.

Algorithm 26 Conditional gradient for matrix completion

1: Let X^1 be an arbitrary matrix of trace k in $\mathcal{K}$.
2: **for** $t = 1$ to T **do**
3: $\quad \mathbf{v}_t = \sqrt{k} \cdot v_{\max}(-\nabla_t)$.
4: $\quad X^{t+1} = X^t + \eta_t(\mathbf{v}_t \mathbf{v}_t^\top - X^t)$ for $\eta_t \in (0, 1)$.
5: **end for**

Comparison to other gradient-based methods. How does this compare to previous convex optimization methods for solving the same matrix completion problem? As a convex program, we can apply gradient descent, or even more advantageously in this setting, stochastic gradient descent as in §3.4. Recall that the gradient of the objective function at point X^t takes the simple form (7.3). A stochastic estimate for the gradient can be attained by observing just a single entry of the matrix M, and the update itself takes constant time as the gradient estimator is sparse. However, the projection step is significantly more difficult.

In this setting, the convex set $\mathcal{K}$ is the set of bounded nuclear norm matrices. Projecting a matrix onto this set amounts to calculating the SVD of the matrix, which is similar in computational complexity to algorithms for

matrix diagonalization or inversion. The best known algorithms for matrix diagonalization are superlinear in the matrices' size, and thus impractical for large datasets that are common in applications.

In contrast, the CG method does not require projections at all, and replaces them with linear optimization steps over the convex set, which we have observed to amount to singular vector computations. The latter can be implemented to take linear time via the power method or the Lanczos algorithm (see bibliography).

Thus, the Conditional Gradient method allows for optimization of the mathematical program (7.1) with a linear-time operation (eigenvector using power method) per iteration, rather than a significantly more expensive computation (SVD) needed for gradient descent.

7.4 Projections versus Linear Optimization

The conditional gradient (Frank-Wolfe) algorithm described before does not resort to projections, but rather computes a linear optimization problem of the form

$$\arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \mathbf{x}^\top \mathbf{u} \right\}. \tag{7.4}$$

When is the CG method computationally preferable? The overall computational complexity of an iterative optimization algorithm is the product of the number of iterations and the computational cost per iteration. The CG method does not converge as well as the most efficient gradient descent algorithms, meaning it requires more iterations to produce a solution of a comparable level of accuracy. However, for many interesting scenarios the computational cost of a linear optimization step (7.4) is *significantly* lower than that of a projection step.

Let us point out several examples of problems for which we have very efficient linear optimization algorithms, whereas our state-of-the-art algorithms for computing projections are significantly slower.

Recommendation systems and matrix prediction. In the example pointed out in the preceding section of matrix completion, known methods for projection onto the spectahedron, or more generally the bounded nuclear-norm ball, require singular value decompositions, which take superlinear time via our best known methods. In contrast, the CG method requires maximal eigenvector computations which can be carried out in linear time via the power method (or the more sophisticated Lanczos algorithm).

Network routing and convex graph problems. Various routing and graph problems can be modeled as convex optimization problems over a convex set called the flow polytope.

Consider a directed acyclic graph with m edges, a source node marked s and a target node marked t. Every path from s to t in the graph can be represented by its identifying vector, that is a vector in $\{0,1\}^m$ in which the entries that are set to 1 correspond to edges of the path. The flow polytope of the graph is the convex hull of all such identifying vectors of the simple paths from s to t. This polytope is also exactly the set of all unit s–t flows in the graph if we assume that each edge has a unit flow capacity (a flow is represented here as a vector in $\mathbb{R}^m$ in which each entry is the amount of flow through the corresponding edge).

Since the flow polytope is just the convex hull of s–t paths in the graph, minimizing a linear objective over it amounts to finding a minimum weight path given weights for the edges. For the shortest path problem we have very efficient combinatorial optimization algorithms, namely Dijkstra's algorithm.

Thus, applying the CG algorithm to solve **any** convex optimization problem over the flow polytope will only require iterative shortest path computations.

Ranking and permutations. A common way to represent a permutation or ordering is by a permutation matrix. Such are square matrices over $\{0,1\}^{n\times n}$ that contain exactly one 1 entry in each row and column.

Doubly-stochastic matrices are square, real-valued matrices with non-negative entries, in which the sum of entries of each row and each column amounts to 1. The polytope that defines all doubly-stochastic matrices is called the Birkhoff-von Neumann polytope. The Birkhoff-von Neumann theorem states that this polytope is the convex hull of exactly all $n \times n$ permutation matrices.

Since a permutation matrix corresponds to a perfect matching in a fully connected bipartite graph, linear minimization over this polytope corresponds to finding a minimum weight perfect matching in a bipartite graph.

Consider a convex optimization problem over the Birkhoff-von Neumann polytope. The CG algorithm will iteratively solve a linear optimization problem over the BVN polytope, thus iteratively solving a minimum weight perfect matching in a bipartite graph problem, which is a well-studied combinatorial optimization problem for which we know of efficient algorithms. In contrast, other gradient based methods will require projections, which

are quadratic optimization problems over the BVN polytope.

Matroid polytopes. A matroid is pair (E, I) where E is a set of elements and I is a set of subsets of E called the independent sets which satisfy various interesting proprieties that resemble the concept of linear independence in vector spaces. Matroids have been studied extensively in combinatorial optimization and a key example of a matroid is the graphical matroid in which the set E is the set of edges of a given graph and the set I is the set of all subsets of E which are cycle-free. In this case, I contains all the spanning trees of the graph. A subset $S \in I$ could be represented by its identifying vector which lies in $\{0,1\}^{|E|}$ which also gives rise to the matroid polytope which is just the convex hull of all identifying vectors of sets in I. It can be shown that some matroid polytopes are defined by exponentially many linear inequalities (exponential in $|E|$), which makes optimization over them difficult.

On the other hand, linear optimization over matroid polytopes is easy using a simple greedy procedure which runs in nearly linear time. Thus, the CG method serves as an efficient algorithm to solve any convex optimization problem over matroids iteratively using only a simple greedy procedure.

7.5 The Online Conditional Gradient Algorithm

In this section we give a projection-free algorithm for OCO based on the conditional gradient method, which is projection-free and thus carries the computational advantages of the CG method to the online setting.

It is tempting to apply the CG method straightforwardly to the online appearance of functions in the OCO setting, such as the OGD algorithm in §3.1. However, it can be shown that an approach that only takes into account the last cost function is doomed to fail. The reason is that the conditional gradient method takes into account the *direction* of the gradient, and is insensitive to its *magnitude*.

Instead, we apply the CG algorithm step to the aggregate sum of all previous cost functions with added Euclidean regularization. The resulting algorithm is given formally in Algorithm 27.

We can prove the following regret bound for this algorithm. While this regret bound is suboptimal in light of the previous upper bounds we have seen, its suboptimality is compensated by the algorithm's lower computational cost.

Algorithm 27 Online conditional gradient

Input: convex set $\mathcal{K}$, T, $\mathbf{x}_1 \in \mathcal{K}$, parameters η, $\{\sigma_t\}$.
for $t = 1, 2, \ldots, T$ **do**
  Play $\mathbf{x}_t$ and observe f_t.
  Let $F_t(\mathbf{x}) = \eta \sum_{\tau=1}^{t-1} \nabla_\tau^\top \mathbf{x} + \|\mathbf{x} - \mathbf{x}_1\|^2$.
  Compute $\mathbf{v}_t = \arg\min_{\mathbf{x} \in \mathcal{K}} \{\nabla F_t(\mathbf{x}_t) \cdot \mathbf{x}\}$.
  Set $\mathbf{x}_{t+1} = (1 - \sigma_t)\mathbf{x}_t + \sigma_t \mathbf{v}_t$.
end for

Theorem 7.3. *Online conditional gradient (Algorithm 27) with parameters* $\eta = \frac{D}{2GT^{3/4}}, \sigma_t = \min\{1, \frac{2}{t^{1/2}}\}$, *attains the following guarantee*

$$\text{Regret}_T = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}^\star \in \mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \ \leq 8DGT^{3/4}$$

As a first step in analyzing Algorithm 27, consider the points

$$\mathbf{x}_t^\star = \arg\min_{\mathbf{x} \in \mathcal{K}} F_t(\mathbf{x}).$$

These are exactly the iterates of the RFTL algorithm from chapter 5, namely Algorithm 13 with the regularization being $R(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_1\|^2$, applied to cost functions with a shift, namely:

$$\tilde{f}_t = f_t(\mathbf{x} + (\mathbf{x}_t^\star - \mathbf{x}_t)).$$

The reason is that ∇_t in Algorithm 27 refers to $\nabla f_t(\mathbf{x}_t)$, whereas in the RFTL algorithm we have $\nabla_t = \nabla f_t(\mathbf{x}_t^\star)$. Notice that for any point $\mathbf{x} \in \mathcal{K}$ we have $|f_t(\mathbf{x}) - \tilde{f}_t(\mathbf{x})| \leq G\|\mathbf{x}_t - \mathbf{x}_t^\star\|$. Thus, according to Theorem 5.2, we have that

$$\begin{aligned} &\textstyle\sum_{t=1}^T f_t(\mathbf{x}_t^\star) - \sum_{t=1}^T f_t(\mathbf{x}^\star) \\ &\leq 2G \textstyle\sum_t \|\mathbf{x}_t - \mathbf{x}_t^\star\| + \sum_{t=1}^T \tilde{f}_t(\mathbf{x}_t^\star) - \sum_{t=1}^T \tilde{f}_t(\mathbf{x}^\star) \\ &\leq 2G \textstyle\sum_t \|\mathbf{x}_t - \mathbf{x}_t^\star\| + 2\eta GT + \frac{1}{\eta} D. \end{aligned} \tag{7.5}$$

Using our previous notation, denote by $h_t(\mathbf{x}) = F_t(\mathbf{x}) - F_t(\mathbf{x}_t^\star)$, and by $h_t = h_t(\mathbf{x}_t)$. The main lemma we require to proceed is the following, which relates the iterates $\mathbf{x}_t$ to the optimal point according to the aggregate function F_t.

Lemma 7.4. *The iterates* $\mathbf{x}_t$ *of Algorithm 27 satisfy for all* $t \geq 1$

$$h_t \leq 2D^2 \sigma_t.$$

Proof. As the functions F_t are 1-smooth, applying the offline Frank-Wolfe analysis technique, and in particular Equation (7.2) to the function F_t we obtain:

$$\begin{aligned} h_t(\mathbf{x}_{t+1}) &= F_t(\mathbf{x}_{t+1}) - F_t(\mathbf{x}_t^\star) \\ &\leq (1-\sigma_t)(F_t(\mathbf{x}_t) - F_t(\mathbf{x}_t^\star)) + \frac{D^2}{2}\sigma_t^2 && \text{Equation (7.2)} \\ &= (1-\sigma_t)h_t + \frac{D^2}{2}\sigma_t^2. \end{aligned}$$

In addition, by definition of F_t and h_t we have

$$\begin{aligned} & h_{t+1} \\ & = F_t(\mathbf{x}_{t+1}) - F_t(\mathbf{x}_{t+1}^\star) + \eta\nabla_{t+1}(\mathbf{x}_{t+1} - \mathbf{x}_{t+1}^\star) \\ & \leq h_t(\mathbf{x}_{t+1}) + \eta\nabla_{t+1}(\mathbf{x}_{t+1} - \mathbf{x}_{t+1}^\star) && F_t(\mathbf{x}_t^\star) \leq F_t(\mathbf{x}_{t+1}^\star) \\ & \leq \ h_t(\mathbf{x}_{t+1}) + \eta G\|\mathbf{x}_{t+1} - \mathbf{x}_{t+1}^\star\|. && \text{Cauchy-Schwarz} \end{aligned}$$

Since F_t is 1-strongly convex, we have

$$\|\mathbf{x} - \mathbf{x}_t^\star\|^2 \leq F_t(\mathbf{x}) - F_t(\mathbf{x}_t^\star).$$

Thus,

$$\begin{aligned} h_{t+1} &\leq \ h_t(\mathbf{x}_{t+1}) + \eta G\|\mathbf{x}_{t+1} - \mathbf{x}_{t+1}^\star\| \\ &\leq h_t(\mathbf{x}_{t+1}) + \eta G\sqrt{h_{t+1}} \\ &\leq h_t(1-\sigma_t) + \frac{1}{2}D^2\sigma_t^2 + \eta G\sqrt{h_{t+1}} && \text{above derivation} \\ &\leq h_t(1 - \frac{5}{6}\sigma_t) + \frac{5}{8}D^2\sigma_t^2. && \text{equation (7.6) below} \end{aligned}$$

Above we used the following derivation, that holds by choice of parameters $\eta = \frac{D}{2GT^{3/4}}$ and $\sigma_t = \min\{1, \frac{2}{t^{1/2}}\}$: since η, G, h_t are all non-negative, we have

$$\begin{aligned} \eta G\sqrt{h_{t+1}} \quad &= \left(\sqrt{D}G\eta\right)^{2/3}\left(\tfrac{G\eta}{D}\right)^{1/3}\sqrt{h_{t+1}} \\ &\leq \tfrac{1}{2}\left(\sqrt{D}G\eta\right)^{4/3} + \tfrac{1}{2}\left(\tfrac{G\eta}{D}\right)^{2/3} h_{t+1} \\ &\leq \tfrac{1}{8}D^2\sigma_t^2 + \tfrac{1}{6}\sigma_t h_{t+1} \end{aligned} \tag{7.6}$$

We now claim that the theorem follows inductively. The base of the induction holds since, for $t = 1$, the definition of F_1 implies

$$h_1 = F_1(\mathbf{x}_1) - F_1(\mathbf{x}^\star) = \|\mathbf{x}_1 - \mathbf{x}^\star\|^2 \leq D^2 \leq 2D^2\sigma_1.$$

Assuming the bound is true for t, we now show it holds for $t+1$ as well:

$$\begin{aligned} h_{t+1} &\leq h_t(1-\frac{5}{6}\sigma_t)+\frac{5}{8}D^2\sigma_t^2 \\ &\leq 2D^2\sigma_t\left(1-\frac{5}{6}\sigma_t\right)+\frac{5}{8}D^2\sigma_t^2 \\ &\leq 2D^2\sigma_t\left(1-\frac{\sigma_t}{2}\right) \\ &\leq 2D^2\sigma_{t+1}, \end{aligned}$$

as required. The last inequality follows by the definition of σ_t (see exercises). □

We proceed to use this lemma in order to prove our theorem:

Proof of Theorem 7.3. By definition, the functions F_t are 1-strongly convex. Thus, we have for $\mathbf{x}_t^\star = \arg\min_{\mathbf{x}\in\mathcal{K}} F_t(\mathbf{x})$:

$$\|\mathbf{x}-\mathbf{x}_t^\star\|^2 \leq F_t(\mathbf{x}) - F_t(\mathbf{x}_t^\star).$$

Let $\eta = \frac{D}{2GT^{3/4}}$, and notice that this satisfies the constraint of Lemma 7.4, which requires $\eta G\sqrt{h_{t+1}} \leq \frac{D^2}{2}\sigma_t^2$. In addition, $\eta < 1$ for T large enough. Hence,

$$\begin{aligned} f_t(\mathbf{x}_t) - f_t(\mathbf{x}_t^\star) &\leq G\|\mathbf{x}_t-\mathbf{x}_t^\star\| \\ &\leq G\sqrt{F_t(\mathbf{x}_t)-F_t(\mathbf{x}_t^\star)} \\ &\leq 2GD\sqrt{\sigma_t}. \qquad \text{Lemma 7.4} \end{aligned} \tag{7.7}$$

Putting everything together we obtain:

$$
\begin{aligned}
\mathit{Regret}_T(OCG) &= \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \\
&= \sum_{t=1}^{T} \left[f_t(\mathbf{x}_t) - f_t(\mathbf{x}_t^\star) + f_t(\mathbf{x}_t^\star) - f_t(\mathbf{x}^\star) \right] \\
&\leq \sum_{t=1}^{T} 2GD\sqrt{\sigma_t} + \sum_t \left[f_t(\mathbf{x}_t^\star) - f_t(\mathbf{x}^\star) \right] && \text{by (7.7)} \\
&\leq 4GDT^{3/4} + \sum_t \left[f_t(\mathbf{x}_t^\star) - f_t(\mathbf{x}^\star) \right] \\
&\leq 4GDT^{3/4} + 2G \sum_t \|\mathbf{x}_t - \mathbf{x}_t^\star\| + 2\eta GT + \frac{1}{\eta} D. && \text{by (7.5)}
\end{aligned}
$$

We thus obtain:

$$
\begin{aligned}
\mathrm{Regret}_T(OCG) &\leq 4GDT^{3/4} + 2\eta G^2 T + \frac{D^2}{\eta} \\
&\leq 4GDT^{2/3} + DGT^{1/4} + 2DGT^{3/4} \leq 8DGT^{3/4}.
\end{aligned}
$$

□

7.6 Bibliographic Remarks

The matrix completion model has been extremely popular since its inception in the context of recommendation systems [Srebro, 2004, Rennie and Srebro, 2005, Salakhutdinov and Srebro, 2010, Lee et al., 2010, Candes and Recht, 2009, Shamir and Shalev-Shwartz, 2011].

The conditional gradient algorithm was devised in the seminal paper by Frank and Wolfe [1956]. Due to the applicability of the FW algorithm to large-scale constrained problems, it has been a method of choice in recent machine learning applications, to name a few: [Hazan, 2008, Jaggi and Sulovský, 2010, Lacoste-Julien et al., 2013, Jaggi, 2013, Dudík et al., 2012, Harchaoui et al., 2012, Hazan and Kale, 2012, Shalev-Shwartz et al., 2011b, Bach et al., 2012, Tewari et al., 2011, Garber and Hazan, 2011, 2013, Bellet et al., 2014]. In the context of matrix completion and recommendation systems, several faster variants of the Frank-Wolfe method were proposed [Garber, 2016, Allen-Zhu et al., 2017]

The online conditional gradient algorithm is due to Hazan and Kale [2012]. An optimal regret algorithm, attaining the $O(\sqrt{T})$ bound, for the special case of polyhedral sets was devised in [Garber and Hazan, 2013].

Recent works consider accelerating projection-free optimization using variance reduction [Lan and Zhou, 2016, Hazan and Luo, 2016], and the case of projection-free algorithms with stochastic gradient oracles [Mokhtari et al., 2018, Chen et al., 2018, Xie et al., 2019].

For an analysis of the running time of the power and Lanczos methods for computing eigenvectors see [Kuczyński and Woźniakowski, 1992]. For modern algorithms for fast computation of the singular value decomposition see [Allen-Zhu and Li, 2016, Musco and Musco, 2015].

7.7 Exercises

1. Prove that if the singular values are smaller than or equal to one, then the nuclear norm is a lower bound on the rank, i.e., show

$$\text{rank}(X) \geq \|X\|_*.$$

2. Prove that the trace is related to the nuclear norm via

$$\|X\|_* = \mathbf{Tr}(\sqrt{XX^\top}) = \mathbf{Tr}(\sqrt{X^\top X}).$$

3. In this question we show that maximizing a linear function over the spectahedron can be reduced to a maximal eigenvector computation.

(a) Consider the the following mathematical program for a symmetric $C \in \mathbb{R}^{d\times d}$:

$$\begin{aligned} &\max X \bullet C \\ &X \in S_d = \{X \in \mathbb{R}^{d\times d} \ , \ X \succcurlyeq 0 \ , \ \mathbf{Tr}(X) \leq 1\}. \end{aligned}$$

Prove that it has the same solution as the mathematical program:

$$\begin{aligned} &\max_{\mathbf{x}\in\mathbb{R}^d} \mathbf{x}^\top C\mathbf{x} \\ &\text{s.t. } \|\mathbf{x}\|_2 \leq 1. \end{aligned}$$

(b) Show how to use eigenvector computations to maximize a general (a-symmetric) linear functions over the spectahedron.

4. Prove that for positive integers $t > 0$, any $c \in [0,1]$ and $\sigma_t = \frac{2}{t^c}$ it holds that

$$\sigma_t(1 - \frac{\sigma_t}{2}) \leq \sigma_{t+1}.$$

5. Download the MovieLens dataset from the web. Implement an online recommendation system based on the matrix completion model: implement the OCG and OGD algorithms for matrix completion. Benchmark your results.

Chapter 8

Games, Duality, and Regret

In this chapter we tie the material covered thus far to some of the most intriguing concepts in optimization and game theory. We shall use the existence of online convex optimization algorithms with sublinear regret to prove two fundamental properties: convex duality in mathematical optimization, and von Neumann's minimax theorem in game theory.

Historically, the theory of games was developed by von Neumann in the early 1930's. In an entirely different scientific thread, the theory of linear programming (LP) was advanced by Dantzig a decade later. Dantzig describes in his memoir a notable meeting between himself and von Neumann at Princeton in 1947. In this meeting, according to Dantzig, after describing the geometric and algebraic versions of linear programming, von Neumann essentially formulated and proved linear programming duality:

> "I don't want you to think I am pulling all this out of my sleeve at the spur of the moment like a magician. I have just recently completed a book with Oscar Morgenstern on the theory of games. What I am doing is conjecturing that the two problems are equivalent. The theory that I am outlining for your problem is an analogue to the one we have developed for games." [13]

At that time, the topic of discussion was not the existence and uniqueness of equilibrium in zero-sum games, which is captured by the minimax theorem. Both concepts were originally captured and proved using very different mathematical techniques: the minimax theorem was originally proved using machinery from mathematical topology, whereas linear programming duality was shown using convexity and geometric tools.

More than half a century later, Freund and Schapire tied both concepts, which were by then known to be strongly related, to regret minimization.

We shall follow their lead in this chapter, introduce the relevant concepts and give concise proofs using the machinery developed earlier in this manuscript.

The chapter can be read with basic familiarity with linear programming and little or no background in game theory. We define linear programming and zero-sum games succinctly, barely enough to prove the duality theorem and the minimax theorem. The reader is referred to the numerous wonderful texts available on linear programming and game theory for a much more thorough introduction and definitions.

8.1 Linear Programming and Duality

Linear programming is a widely successful and practical convex optimization framework. Amongst its numerous successes is the Nobel prize award given on account of its application to economics. It is a special case of the convex optimization problem from chapter 2 in which $\mathcal{K}$ is a polyhedron (i.e., an intersection of a finite set of halfspaces) and the objective function is a linear function. Thus, a linear program can be described as follows, where ($A \in \mathbb{R}^{n \times m}$):

$$\begin{array}{ll} \min & c^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} \geq b \quad . \end{array}$$

The above formulation can be transformed into several different forms via basic manipulations. For example, any LP can be transformed to an equivalent LP with the variables taking only non-negative values. This can be accomplished by writing every variable x as $x = x^+ - x^-$, with $x^+, x^- \geq 0$. It can be verified that this transformation leaves us with another LP, whose variables are non-negative, and contains at most twice as many variables (see exercises section for more details).

We are now ready to define a central notion in LP and state the duality theorem:

Theorem 8.1 (The duality theorem)**.** *Given a linear program:*

$$\begin{array}{ll} \min & c^\top \mathbf{x} \\ \textit{s.t.} & A\mathbf{x} \geq b, \\ & \mathbf{x} \geq 0, \end{array}$$

its dual program is given by:

$$\begin{aligned} \max \quad & b^\top \mathbf{y} \\ s.t. \quad & A^\top \mathbf{y} \leq c, \\ & \mathbf{y} \geq 0. \end{aligned}$$

and the objectives of both problems are either equal or unbounded.

Instead of studying duality directly, we proceed to define zero-sum games and an analogous concept to duality.

8.2 Zero-sum Games and Equilibria

The theory of games is an established research field in economic theory. We give here brief definitions of the main concepts studied in this chapter.

Let us start with an example of a zero-sum game we all know: the rock-paper-scissors game. In this game each of the two players chooses a strategy: either rock, scissors or paper. The winner is determined according to the following table, where 0 denotes a draw, -1 denotes that the row player wins, and 1 denotes a column player victory.

-	**scissors**	**paper**	**rock**
rock	-1	1	0
paper	1	0	-1
scissors	0	-1	1

Table 8.1: Example of a zero-sum game in matrix representation.

The rock-paper-scissors game is called a "zero-sum" game since one can think of the numbers as losses for the row player (loss of -1 resembles victory, 1 loss and 0 draw), in which case the column player receives a loss which is exactly the negation of the loss of the row player. Thus the sum of losses which both players suffer is zero in every outcome of the game.

Noticed that we termed one player as the "row player" and the other as the "column player" corresponding to the matrix losses. Such a matrix representation is far more general:

Definition 8.2. *A two-player zero-sum-game in normal form is given by a matrix $A \in [-1,1]^{n\times m}$. The loss for the row player playing strategy $i \in [n]$ is equal to the negative loss (reward) of the column player playing strategy $j \in [m]$ and equal to A_{ij}.*

The fact that the losses were defined in the range $[-1, 1]$ is arbitrary, as the concept of main importance we define next is invariant to scaling and shifting by a constant.

A central concept in game theory is equilibrium. There are many different notions of equilibria. In two-player zero-sum games, a pure equilibrium is a pair of strategies $(i, j) \in [n] \times [m]$ with the following property: given that the column player plays j, there is no strategy that dominates i - i.e., every other strategy $k \in [n]$ gives higher or equal loss to the row player. Equilibrium also requires that a symmetric property for strategy j holds - it is not dominated by any other strategy given that the row player plays i.

It can be shown that some games do not have a pure equilibrium as defined above, e.g., the rock-paper-scissors game. However, we can extend the notion of a strategy to a *mixed* strategy - a distribution over pure strategies. The loss of a mixed strategy is the expected loss according to the distribution over pure strategies. More formally, if the row player chooses $\mathbf{x} \in \Delta_n$ and column player chooses $\mathbf{y} \in \Delta_m$, then the expected loss of the row player (which is the negative reward to the column player) is given by:

$$\mathbf{E}[\text{loss}] = \sum_{i \in [n]} \mathbf{x}_i \sum_{j \in [m]} \mathbf{y}_j A_{ij} = \mathbf{x}^\top A \mathbf{y}.$$

We can now generalize the notion of equilibrium to mixed strategies. Given a row strategy $\mathbf{x}$, it is dominated by $\tilde{\mathbf{x}}$ with respect to a column strategy $\mathbf{y}$ if and only if

$$\mathbf{x}^\top A \mathbf{y} > \tilde{\mathbf{x}}^\top A \mathbf{y}.$$

We say that $\mathbf{x}$ is dominant with respect to $\mathbf{y}$ if and only if it is not dominated by any other mixed strategy. A pair $(\mathbf{x}, \mathbf{y})$ is an equilibrium for game A if and only if both $\mathbf{x}$ and $\mathbf{y}$ are dominant with respect to each other. It is a good exercise for the reader at this point to find an equilibrium for the rock-paper-scissors game.

At this point, some natural questions arise: Is there always an equilibrium in a given zero-sum game? Can it be computed efficiently? Are there natural repeated-game-playing strategies that reach it?

As we shall see, the answer to all questions above is affirmative. Let us rephrase these questions in a different way. Consider the optimal row strategy, i.e., a mixed strategy $\mathbf{x}$, such that the expected loss is minimized, no matter what the column player does. The optimal strategy for the row player would be:

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x} \in \Delta_n} \max_{\mathbf{y} \in \Delta_m} \mathbf{x}^\top A \mathbf{y}.$$

Notice that we use the notation $\mathbf{x}^\star \in$ rather than $\mathbf{x}^\star =$, since in general the set of strategies attaining the minimal loss over worst-case column strategies can contain more than a single strategy. Similarly, the optimal strategy for the column player would be:

$$\mathbf{y}^\star \in \arg\max_{\mathbf{y}\in\Delta_m} \min_{\mathbf{x}\in\Delta_n} \mathbf{x}^\top A\mathbf{y}.$$

Playing these strategies, no matter what the column player does, the row player would pay no more than

$$\lambda_R = \min_{\mathbf{x}\in\Delta_n} \max_{\mathbf{y}\in\Delta_m} \mathbf{x}^\top A\mathbf{y} = \max_{\mathbf{y}\in\Delta_m} \mathbf{x}^{\star\top} A\mathbf{y},$$

and column player would earn at least

$$\lambda_C = \max_{\mathbf{y}\in\Delta_m} \min_{\mathbf{x}\in\Delta_n} \mathbf{x}^\top A\mathbf{y} = \min_{\mathbf{x}\in\Delta_n} \mathbf{x}^\top A\mathbf{y}^\star.$$

With these definitions we can state von Neumann's famous minimax theorem:

Theorem 8.3 (von Neumann minimax theorem)**.** *In any zero-sum game, it holds that* $\lambda_R = \lambda_C$*.*

This theorem answers all our above questions on the affirmative. The value $\lambda^\star = \lambda_C = \lambda_R$ is called the **value of the game**, and its existence and uniqueness imply that any $\mathbf{x}^\star$ and $\mathbf{y}^\star$ in the appropriate optimality sets are an equilibrium.

We proceed to give a constructive proof of von Neumann's theorem which also yields an efficient algorithm as well as natural repeated-game playing strategies that converge to it.

8.2.1 Equivalence of von Neumann Theorem and LP duality

The von Neumann theorem is equivalent to the duality theorem of linear programming in a very strong sense, and either implies the other via simple reduction. Thus, it suffices to prove only von Neumann's theorem to prove the duality theorem.

The first part of this equivalence is shown by representing a zero-sum game as a primal-dual linear program instance, as we do now.

Observe that the definition of an optimal row strategy and value is equivalent to the following LP:

$$\begin{aligned} \min \quad & \lambda \\ \text{s.t.} \quad & \sum \mathbf{x}_i = 1 \\ & \forall i \in [m] \;.\; \mathbf{x}^\top A e_i \leq \lambda \\ & \forall i \in [n] \;.\; \mathbf{x}_i \geq 0. \end{aligned}$$

To see that the optimum of the above LP is attained at λ_R, note that the constraint $\mathbf{x}^\top A e_i \leq \lambda \quad \forall i \in [m]$ is equivalent to the constraint $\forall \mathbf{y} \in \Delta_m \;.\; \mathbf{x}^\top A \mathbf{y} \leq \lambda$, since:

$$\forall \mathbf{y} \in \Delta_m \;. \quad \mathbf{x}^\top A \mathbf{y} = \sum_{j=1}^{m} \mathbf{x}^\top A e_j \cdot \mathbf{y}_j \leq \lambda \sum_{j=1}^{m} \mathbf{y}_j = \lambda$$

The dual program to the above LP is given by

$$\begin{aligned} \max \quad & \mu \\ \text{s.t.} \quad & \sum \mathbf{y}_i = 1 \\ & \forall i \in [n] \;.\; e_i^\top A \mathbf{y} \geq \mu \\ & \forall i \in [m] \;.\; \mathbf{y}_i \geq 0. \end{aligned}$$

By similar arguments, the dual program precisely defines λ_C and $\mathbf{y}^\star$. The duality theorem asserts that $\lambda_R = \lambda_C = \lambda^\star$, which gives von Neumann's theorem.

The other direction, i.e., showing that von Neumann's theorem implies LP duality, is slightly more involved. Basically, one can convert any LP into the format of a zero-sum game. Special care is needed to ensure that the original LP is indeed feasible, as zero-sum games are always feasible and linear programs need not be. The details are left as an exercise at the end of this chapter.

8.3 Proof of von Neumann Theorem

In this section we give a proof of von Neumann's theorem using online convex optimization algorithms with sublinear regret.

The first part of the theorem, which is also known as weak duality in the LP context, is rather straightforward:

Direction 1 ($\lambda_R \geq \lambda_C$):

Proof.

$$\begin{aligned}
\lambda_R &= \min_{\mathbf{x}\in\Delta_n} \max_{\mathbf{y}\in\Delta_m} \mathbf{x}^\top A\mathbf{y} \\
&= \max_{\mathbf{y}\in\Delta_m} \mathbf{x}^{\star\top} A\mathbf{y} && \text{definition of } \mathbf{x}^\star \\
&\geq \max_{\mathbf{y}\in\Delta_m} \min_{\mathbf{x}\in\Delta_n} \mathbf{x}^\top A\mathbf{y} \\
&= \lambda_C.
\end{aligned}$$

□

The second and main direction, known as strong duality in the LP context, requires the technology of online convex optimization we have proved thus far:

Direction 2 ($\lambda_R \leq \lambda_C$):

Proof. We consider a repeated game defined by the $n \times m$ matrix A. For $t = 1, 2, ..., T$, the row player provides a mixed strategy $\mathbf{x}_t \in \Delta_n$, column player plays mixed strategy $\mathbf{y}_t \in \Delta_m$, and the loss of the row player, which equals to the reward of the column player, equals $\mathbf{x}_t^\top A\mathbf{y}_t$.

The row player generates the mixed strategies $\mathbf{x}_t$ according to an OCO algorithm — specifically using the Exponentiated Gradient algorithm 15 from chapter 5. The convex decision set is taken to be the n dimensional simplex $\mathcal{K} = \Delta_n = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x}(i) \geq 0, \sum \mathbf{x}(i) = 1\}$. The loss function at time t is given by

$$f_t(\mathbf{x}) = \mathbf{x}^\top A\mathbf{y}_t \quad (f_t \text{ is linear with respect to } \mathbf{x}) \ .$$

Spelling out the EG strategy for this particular instance, we have

$$\mathbf{x}_{t+1}(i) \leftarrow \frac{\mathbf{x}_t(i)e^{-\eta A_i \mathbf{y}_t}}{\sum_j \mathbf{x}_t(i)e^{-\eta A_j \mathbf{y}_t}} \ .$$

Then, by appropriate choice of η and Corollary 5.7, we have

$$\sum_t f_t(\mathbf{x}_t) \quad \leq \quad \min_{\mathbf{x}^\star\in\mathcal{K}} \sum_t f_t(\mathbf{x}^\star) + \sqrt{2T \log n} \ . \tag{8.1}$$

The column player plays her best response to the row player's strategy, that is:

$$\mathbf{y}_t = \arg\max_{\mathbf{y}\in\Delta_m} \mathbf{x}_t^\top A\mathbf{y}. \tag{8.2}$$

Denote the average mixed strategies by:

$$\bar{\mathbf{x}} = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{x}_\tau \quad , \quad \bar{\mathbf{y}} = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{y}_\tau \ .$$

Then, we have

$$\begin{aligned}
\lambda_R &= \min_{\mathbf{x}}\max_{\mathbf{y}}\ \mathbf{x}^\top A\mathbf{y} \\
&\le \max_{\mathbf{y}} \bar{\mathbf{x}}^\top A\mathbf{y} && \text{special case} \\
&= \frac{1}{T}\sum_t \mathbf{x}_t A\mathbf{y}^\star \\
&\le \frac{1}{T}\sum_t \mathbf{x}_t A\mathbf{y}_t && \text{by (8.2)} \\
&\le \frac{1}{T}\min_{\mathbf{x}}\sum_t \mathbf{x}^\top A\mathbf{y}_t + \sqrt{2\log n/T} && \text{by (8.1)} \\
&= \min_{\mathbf{x}} \mathbf{x}^\top A\bar{\mathbf{y}} + \sqrt{2\log n/T} \\
&\le \max_{\mathbf{y}}\min_{\mathbf{x}} \mathbf{x}^\top A\mathbf{y} + \sqrt{2\log n/T} && \text{special case} \\
&= \lambda_C + \sqrt{2\log n/T}.
\end{aligned}$$

Thus $\lambda_R \le \lambda_C + \sqrt{2\log n/T}$. As $T \to \infty$, we obtain part 2 of the theorem. □

Notice that besides the basic definitions, the only tool used in the proof is the existence of sublinear regret algorithms for online convex optimization. The fact that the regret bounds for OCO algorithms were defined without restricting the cost functions, and that they can be adversarially chosen, is crucial for the proof. The functions f_t are defined according to $\mathbf{y}_t$, which is chosen based on $\mathbf{x}_t$. Thus, the cost functions we constructed are adversarially chosen after the decision $\mathbf{x}_t$ was made by the row player.

8.4 Approximating Linear Programs

The technique in the preceding section not only proves the minimax theorem, and thus linear programming duality, but also entails an efficient algorithm. Using the equivalence of zero-sum games and linear programs, this efficient algorithm can be used to solve linear programming. We now spell out the details of this algorithm in the context of zero-sum games.

Consider the following algorithm:

Algorithm 28 Simple LP

Input: linear program in zero-sum game format, by matrix $A \in \mathbb{R}^{n \times m}$.
Let $\mathbf{x}_1 = [1/n, 1/n, ..., 1/n]$
for $t = 1$ to T **do**
    Compute $\mathbf{y}_t = \max_{\mathbf{y} \in \Delta_m} \mathbf{x}_t^\top A \mathbf{y}$
    Update $\forall i \; . \; \mathbf{x}_{t+1}(i) \leftarrow \frac{\mathbf{x}_t(i) e^{-\eta A_i \mathbf{y}_t}}{\sum_j \mathbf{x}_t(j) e^{-\eta A_j \mathbf{y}_t}}$
end for
return $\bar{\mathbf{x}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_t$

Almost immediately we obtain from the previous derivation the following:

Lemma 8.4. *The returned vector $\bar{\mathbf{x}}$ of Algorithm 28 is a $\frac{\sqrt{2 \log n}}{\sqrt{T}}$-approximate solution to the zero-sum game and linear program it describes.*

Proof. Following the exact same steps of the previous derivation, we have

$$\begin{aligned}
\max_{\mathbf{y}} \bar{\mathbf{x}}^\top A \mathbf{y} &= \frac{1}{T} \sum_t \mathbf{x}_t A \mathbf{y}^\star \\
&\leq \frac{1}{T} \sum_t \mathbf{x}_t A \mathbf{y}_t && \text{by (8.2)} \\
&\leq \frac{1}{T} \min_{\mathbf{x}} \sum_t \mathbf{x}^\top A \mathbf{y}_t + \sqrt{2 \log n / T} && \text{by (8.1)} \\
&= \min_{\mathbf{x}} \mathbf{x}^\top A \bar{\mathbf{y}} + \sqrt{2 \log n / T} \\
&\leq \max_{\mathbf{y}} \min_{\mathbf{x}} \mathbf{x}^\top A \mathbf{y} + \sqrt{2 \log n / T} && \text{special case} \\
&= \lambda^\star + \sqrt{2 \log n / T}.
\end{aligned}$$

Therefore, for each $i \in [m]$:

$$\bar{\mathbf{x}}^\top A e_i \leq \lambda^\star + \frac{\sqrt{2 \log n}}{\sqrt{T}}$$

□

Thus, to obtain an ε-approximate solution, one would need $\frac{2 \log n}{\varepsilon^2}$ iterations, each involving a simple update procedure.

8.5 Bibliographic Remarks

Game theory was founded in the late 1920's-early '30s, whose cornerstone was laid in the classic text "Theory of Games and Economic Behavior" by Neumann and Morgenstern [1944].

Linear programming is a fundamental mathematical optimization and modeling tool, dating back to the 1940's and the work of Kantorovich [1940] and Dantzig [1951]. Duality for linear programming was conceived by von Neumann, as described by Dantzig in an interview [Albers et al., 1986]. For in depth treatment of the theory of linear programming there are numerous comprehensive texts, e.g., [Bertsimas and Tsitsiklis, 1997, Matousek and Gärtner, 2007].

The beautiful connection between low-regret algorithms and solving zero-sum games was discovered by Freund and Schapire [1999]. More general connections of convergence of low-regret algorithms to equilibria in games were studied by Hart and Mas-Colell [2000], and more recently in [Even-dar et al., 2009, Roughgarden, 2015].

Approximation algorithms that arise via simple Lagrangian relaxation techniques were pioneered by Plotkin et al. [1995]. See also the survey [Arora et al., 2012] and more recent developments that give rise to sublinear time algorithms [Clarkson et al., 2012, Hazan et al., 2011].

8.6 Exercises

1. Prove that equilibrium strategy pairs in zero-sum games are not unique. That is, construct a zero-sum game for which there is more than one equilibrium.

2. In this question we prove a special case of Sion's generalization to the minimax theorem. Let $f : X \times Y \mapsto \mathbb{R}$ be a real valued function on $X \times Y$, where X, Y are bounded, closed and convex sets in Euclidean space $\mathbb{R}^d$. Let f be convex-concave, i.e.,

(a) For every $\mathbf{y} \in Y$, the function $f(\cdot, \mathbf{y}) : X \mapsto \mathbb{R}$ is convex.

(b) For every $\mathbf{x} \in X$, the function $f(\mathbf{x}, \cdot) : Y \mapsto \mathbb{R}$ is concave. Prove that

$$\min_{\mathbf{x}\in X} \max_{\mathbf{y}\in Y} f(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{y}\in Y} \min_{\mathbf{x}\in X} f(\mathbf{x}, \mathbf{y})$$

3. Read Adler's exposition on the equivalence of linear programming and zero sum games [Adler, 2013]. Explain how to convert a linear program to a zero-sum game.

4. Consider a repeated zero-sum game over a matrix A in which both players change their mixed strategies according to a low-regret algorithm over the linear cost/reward functions of the game. Prove that the average value of the game approaches that of an equilibrium of the game given by A.

5. * Write a semidefinite program as a zero-sum game. Write down an algorithm for approximating the solution of a semidefinite program using OCO algorithms, and sketch an analysis of its correctness and performance bound.

Chapter 9

Learning Theory, Generalization, and Online Convex Optimization

In our treatment of online convex optimization so far we have only implicitly discussed learning theory. The framework of OCO was shown to capture applications such as learning classifiers online, prediction with expert advice, online portfolio selection and matrix completion, all of which have a learning aspect. We have introduced the metric of regret and gave efficient algorithms to minimize regret in various settings. We have also argued that minimizing regret is a meaningful approach for many online prediction problems. However, the relation to other theories of learning was not discussed thus far.

In this section we draw a formal and strong connection between OCO and the theory of statistical learning. We begin by giving the basic definitions of statistical learning theory, and proceed to describe how the applications studied in this manuscript relate to this model. We then continue to show how regret minimization in the setting of online convex optimization gives rise to computationally efficient statistical learning algorithms.

9.1 Statistical Learning Theory

The theory of statistical learning addresses the problem of learning a concept from examples. A concept is a mapping from domain $\mathcal{X}$ to labels $\mathcal{Y}$, denoted $C : \mathcal{X} \mapsto \mathcal{Y}$.

As an example, consider the problem of optical character recognition. In

this setting, the domain $\mathcal{X}$ can be all $n \times n$ bitmap images, the label set $\mathcal{Y}$ is the Latin (or other) alphabet, and the concept C maps a bitmap into the character depicted in the image.

Statistical theory models the problem of learning a concept by allowing access to labelled examples from the target distribution. The learning algorithm has access to pairs, or samples, from an unknown distribution

$$(\mathbf{x}, y) \sim \mathcal{D} \quad , \quad \mathbf{x} \in \mathcal{X} \ , \ y \in \mathcal{Y}.$$

The goal is to be able to predict y as a function of $\mathbf{x}$, i.e., to **learn** a hypothesis, or a mapping from $\mathcal{X}$ to $\mathcal{Y}$, denoted $h : \mathcal{X} \mapsto \mathcal{Y}$, with small error with respect to the distribution $\mathcal{D}$. In the case that the label set is binary $\mathcal{Y} = \{0, 1\}$, or discrete such as in optical character recognition, the *generalization error* of an hypothesis h with respect to distribution $\mathcal{D}$ is given by

$$\text{error}(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) \neq y].$$

More generally, the goal is to learn a hypothesis that minimizes the loss according to a (usually convex) loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$. In this case the generalization error of a hypothesis is defined as:

$$\text{error}(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(h(\mathbf{x}), y)].$$

We henceforth consider learning algorithms $\mathcal{A}$ that observe a sample from the distribution $\mathcal{D}$, denoted $S \sim \mathcal{D}^m$ for a sample of m examples, $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$, and produce a hypothesis $\mathcal{A}(S) : \mathcal{X} \mapsto \mathcal{Y}$ based on this sample.
The goal of statistical learning can thus be summarised as follows:

> *Given access to i.i.d. samples from an arbitrary distribution over $\mathcal{X} \times \mathcal{Y}$ corresponding to a certain concept, learn a hypothesis $h : \mathcal{X} \mapsto \mathcal{Y}$ which has arbitrarily small generalization error with respect to a given loss function.*

9.1.1 Overfitting

In the problem of optical character recognition the task is to recognize a character from a given image in bitmap format. To model it in the statistical learning setting, the domain $\mathcal{X}$ is the set of all $n \times n$ bitmap images for some integer n. The label set $\mathcal{Y}$ is the latin alphabet, and the concept C maps a bitmap into the character depicted in the image.

Consider the naive algorithm which fits the perfect hypothesis for a given sample, in this case set of bitmaps. Namely, $\mathcal{A}(S)$ is the hypothesis which correctly maps any given bitmap input $\mathbf{x}_i$ to its correct label y_i, and maps all unseen bitmaps to the character "1."

Clearly, this hypothesis does a very poor job of generalizing from experience - all images that have not been observed yet will be classified without regard to their properties, surely an erroneous classification most times. However - the training set, or observed examples, are perfectly classified by this hypothesis!

This disturbing phenomenon is called "overfitting," a central concern in machine learning. Before continuing to add the necessary components in learning theory to prevent overfitting, we turn our attention to a formal statement of when overfitting can appear.

9.1.2 No free lunch?

The following theorem shows that learning, as stated in the goal of statistical learning theory, is impossible without restricting the hypothesis class being considered. For simplicity, we consider the zero-one loss in this section.

Theorem 9.1 (No Free Lunch Theorem)**.** *Consider any domain $\mathcal{X}$ of size $|\mathcal{X}| = 2m > 4$, and any algorithm $\mathcal{A}$ which outputs a hypothesis $\mathcal{A}(S)$ given a sample S of size m. Then there exists a concept $C : \mathcal{X} \to \{0, 1\}$ and a distribution $\mathcal{D}$ such that:*

- *The generalization error of the concept C is zero.*
- *With probability at least $\frac{1}{10}$, the error of the hypothesis generated by $\mathcal{A}$ is at least* $\text{error}(A(S)) \geq \frac{1}{10}$.

The proof of this theorem is based on the probabilistic method, a useful technique for showing the existence of combinatorial objects by showing that the probability they exist in some distributional setting is bounded away from zero. In our setting, instead of explicitly constructing a concept C with the required properties, we show it exists by a probabilistic argument.

Proof. We show that for any learning algorithm, there is some learning task (i.e., "hard" concept) that it will not learn well. Formally, take $\mathcal{D}$ to be the uniform distribution over $\mathcal{X}$. Our proof strategy will be to show the following inequality, where we take a uniform distribution over all concepts $\mathcal{X} \mapsto \{0, 1\}$

$$Q \stackrel{def}{=} \mathop{\mathbf{E}}_{C:\mathcal{X}\to\{0,1\}}[\mathop{\mathbf{E}}_{S\sim\mathcal{D}^m}[\text{error}(\mathcal{A}(S))]] \geq \frac{1}{4}.$$

After showing this step, we will use Markov's Inequality to conclude the theorem.

We proceed by using the linearity property of expectations, which allows us to swap the order of expectations, and then conditioning on the event that $\mathbf{x} \in S$.

$$\begin{aligned} Q &= \mathop{\mathbf{E}}_{S}[\mathop{\mathbf{E}}_{C}[\mathop{\mathbf{E}}_{\mathbf{x}\in\mathcal{X}}[\mathcal{A}(S)(\mathbf{x}) \neq C(\mathbf{x})]]] \\ &= \mathop{\mathbf{E}}_{S,\mathbf{x}}[\mathop{\mathbf{E}}_{C}[\mathcal{A}(S)(\mathbf{x}) \neq C(\mathbf{x})|\mathbf{x} \in S]\Pr[\mathbf{x} \in S]] \\ &+ \mathop{\mathbf{E}}_{S,\mathbf{x}}[\mathop{\mathbf{E}}_{C}[\mathcal{A}(S)(\mathbf{x}) \neq C(\mathbf{x})|\mathbf{x} \notin S]\Pr[\mathbf{x} \notin S]]. \end{aligned}$$

All terms in the above expression, and in particular the first term, are non-negative and at least 0. Also note that since the domain size is $2m$ and the sample is of size $|S| \leq m$, we have $\Pr(\mathbf{x} \notin S) \geq \frac{1}{2}$. Finally, observe that $\Pr[\mathcal{A}(S)(\mathbf{x}) \neq C(\mathbf{x})] = \frac{1}{2}$ for all $\mathbf{x} \notin S$ since we are given that the "true" concept C is chosen uniformly at random over all possible concepts. Hence, we get that:

$$Q \geq 0 + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4},$$

which is the intermediate step we wanted to show. The random variable $\mathbf{E}_{S\sim\mathcal{D}^m}[\text{error}(\mathcal{A}(S))]$ attains values in the range $[0, 1]$. Since its expectation is at least $\frac{1}{4}$, the event that it attains a value of at least $\frac{1}{4}$ is non-empty. Thus, there exists a concept such that

$$\mathop{\mathbf{E}}_{S\sim\mathcal{D}^m}[\text{error}(\mathcal{A}(S))] \geq \frac{1}{4}$$

where, as assumed beforehand, $\mathcal{D}$ is the uniform distribution over $\mathcal{X}$.

We now conclude with Markov's Inequality: since the expectation above over the error is at least one-fourth, the probability over examples such that the error of $\mathcal{A}$ over a random sample is at least one-tenth is at least

$$\Pr_{S\sim\mathcal{D}^m}\left(\text{error}(\mathcal{A}(S)) \geq \frac{1}{10}\right) \geq \frac{\frac{1}{4} - \frac{1}{10}}{1 - \frac{1}{10}} > \frac{1}{10}.$$

□

9.1.3 Examples of learning problems

The conclusion of the previous theorem is that the space of possible concepts being considered in a learning problem needs to be restricted for any meaningful guarantee. Thus, learning theory concerns itself with concept classes, also called hypothesis classes, which are sets of possible hypotheses from which one would like to learn. We denote the concept (hypothesis) class by $\mathcal{H} = \{h : \mathcal{X} \mapsto \mathcal{Y}\}$.

Common examples of learning problems that can be formalized in this model and the corresponding definitions include:

- Optimal character recognition: In the problem of optical character recognition the domain $\mathcal{X}$ consists of all $n \times n$ bitmap images for some integer n, the label set $\mathcal{Y}$ is a certain alphabet, and the concept C maps a bitmap image into the character depicted in it. A common (finite) hypothesis class for this problem is the set of all decision trees with bounded depth.

- Text classification: In the problem of text classification the domain is a subset of Euclidean space, i.e., $\mathcal{X} \subseteq \mathbb{R}^d$. Each document is represented in its bag-of-words representation, and d is the size of the dictionary. The label set $\mathcal{Y}$ is binary, where one indicates a certain classification or topic, e.g.,"Economics", and zero others.

  A commonly used hypothesis class for this problem is the set of all bounded-norm vectors in Euclidean space $\mathcal{H} = \{h_{\mathbf{w}}\ ,\ \mathbf{w} \in \mathbb{R}^d\ ,\ \|\mathbf{w}\|_2^2 \leq \omega\}$ such that $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$. The loss function is chosen to be the hinge loss, i.e., $\ell(\hat{y}, y) = \max\{0, 1 - \hat{y}y\}$.

- Recommendation systems: recall the online convex optimization formulation of this problem in section 7.2. A statistical learning formulation for this problem is very similar. The domain is a direct sum of two sets $\mathcal{X} = \mathcal{X}_1 \oplus \mathcal{X}_2$. Here $\mathbf{x}_1 \in \mathcal{X}_1$ is a certain media item, and every person is an item $\mathbf{x}_2 \in \mathcal{X}_2$. The label set $\mathcal{Y}$ is binary, where one indicates a positive sentiment for the person to the particular media item, and zero a negative sentiment.

  A commonly considered hypothesis class for this problem is the set of all mappings $\mathcal{X}_1 \times \mathcal{X}_2 \mapsto \mathcal{Y}$ that, when viewed as a matrix in $\mathbb{R}^{|\mathcal{X}_1| \times |\mathcal{X}_2|}$, have bounded algebraic rank.

9.1.4 Defining generalization and learnability

We are now ready to give the fundamental definition of statistical learning theory, called Probably Approximately Correct (PAC) learning:

Definition 9.2 (PAC learnability)**.** *A hypothesis class $\mathcal{H}$ is PAC learnable with respect to loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ if the following holds. There exists an algorithm $\mathcal{A}$ that accepts $S_T = \{(\mathbf{x}_t, y_t),\ t \in [T]\}$ and returns hypothesis $\mathcal{A}(S_T) \in \mathcal{H}$ that satisfies: for any $\varepsilon, \delta > 0$ there exists a sufficiently large natural number $T = T(\varepsilon, \delta)$, such that for any distribution $\mathcal{D}$ over pairs $(\mathbf{x}, y)$ and T samples from this distribution, it holds that with probability at least $1 - \delta$*

$$\text{error}(\mathcal{A}(S_T)) \leq \varepsilon.$$

A few remarks regarding this definition:

- The set S_T of samples from the underlying distribution is called the training set. The error in the above definition is called the **generalization error**, as it describes the overall error of the concept as generalized from the observed training set. The behavior of the number of samples T as a function of the parameters ε, δ and the concept class is called the **sample complexity** of $\mathcal{H}$.

- The definition of PAC learning says nothing about computational efficiency. Computational learning theory usually requires, in addition to the definition above, that the algorithm $\mathcal{A}$ is efficient, i.e., polynomial running time with respect to $\varepsilon, \log \frac{1}{\delta}$ and the representation of the hypothesis class. The representation size for a discrete set of concepts is taken to be the logarithm of the number of hypotheses in $\mathcal{H}$, denoted $\log |\mathcal{H}|$.

- If the hypothesis $\mathcal{A}(S_T)$ returned by the learning algorithm belongs to the hypothesis class $\mathcal{H}$, as in the definition above, we say that $\mathcal{H}$ is **properly learnable**. More generally, $\mathcal{A}$ may return hypothesis from a different hypothesis class, in which case we say that $\mathcal{H}$ is **improperly learnable**.

The fact that the learning algorithm can learn up to *any* desired accuracy $\varepsilon > 0$ is called the **realizability assumption** and greatly reduces the generality of the definition. It amounts to requiring that a hypothesis with near-zero error belongs to the hypothesis class. In many cases, concepts are only approximately learnable by a given hypothesis class, or inherent noise in the problem prohibits realizability (see exercises).

This issue is addressed in the definition of a more general learning concept, called **agnostic learning**:

Definition 9.3 (agnostic PAC learnability)**.** *The hypothesis class $\mathcal{H}$ is agnostically PAC learnable with respect to loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ if the following holds. There exists an algorithm $\mathcal{A}$ that accepts $S_T = \{(\mathbf{x}_t, y_t),\ t \in [T]\}$ and returns hypothesis $\mathcal{A}(S_T)$ that satisfies: for any $\varepsilon, \delta > 0$ there exists a sufficiently large natural number $T = T(\varepsilon, \delta)$ such that for any distribution $\mathcal{D}$ over pairs $(\mathbf{x}, y)$ and T samples from this distribution, it holds that with probability at least $1 - \delta$*

$$\text{error}(\mathcal{A}(S_T)) \leq \min_{h \in \mathcal{H}} \{\text{error}(h)\} + \varepsilon.$$

With these definitions, we can state the fundamental theorem of statistical learning theory for finite hypothesis classes:

Theorem 9.4 (PAC learnability of finite hypothesis classes)**.** *Every finite concept class $\mathcal{H}$ is agnostically PAC learnable with sample complexity that is* $\text{poly}(\varepsilon, \delta, \log |\mathcal{H}|)$.

In the following sections we prove this theorem, and in fact a more general statement that holds also for certain infinite hypothesis classes. The complete characterization of which infinite hypothesis classes are learnable is a deep and fundamental question, whose complete answer was given by Vapnik and Chervonenkis (see bibliography). The question of which (finite or infinite) hypothesis classes are **efficiently** PAC learnable, especially in the improper sense, is still at the forefront of learning theory today.

9.2 Agnostic Learning using Online Convex Optimization

In this section we show how to use online convex optimization for agnostic PAC learning. Following the paradigm of this manuscript, we describe and analyze a reduction from agnostic learning to online convex optimization. The reduction is formally described in Algorithm 29.

Algorithm 29 Reduction: Learning ⇒ OCO

1: Input: OCO algorithm $\mathcal{A}$, convex hypothesis class $\mathcal{H} \subseteq \mathbb{R}^d$, convex loss function ℓ.
2: Let $h_1 \leftarrow \mathcal{A}(\emptyset)$.
3: **for** $t = 1$ to T **do**
4: Draw labeled example $(\mathbf{x}_t, y_t) \sim \mathcal{D}$.
5: Let $f_t(h) = \ell(h(\mathbf{x}_t), y_t)$.
6: Update
$$h_{t+1} = \mathcal{A}(f_1, ..., f_t).$$
7: **end for**
8: Return $\bar{h} = \frac{1}{T}\sum_{t=1}^{T} h_t$.

For this reduction we assumed that the concept (hypothesis) class is a convex subset of Euclidean space. A similar reduction can be carried out for discrete hypothesis classes (see exercises). In fact, the technique we explore below will work for any hypothesis set $\mathcal{H}$ that admits a low regret algorithm, and can be generalized to infinite hypothesis classes that are known to be learnable.

Let $h^\star = \arg\min_{h\in\mathcal{H}}\{\text{error}(h)\}$ be the hypothesis in the class $\mathcal{H}$ that minimizes the generalization error. Using the assumption that $\mathcal{A}$ guarantees sublinear regret, our simple reduction implies PAC learning, as given in the following theorem.

Theorem 9.5. *Let $\mathcal{A}$ be an OCO algorithm whose regret after T iterations is guaranteed to be bounded by $\text{Regret}_T(\mathcal{A})$. Then for any $\delta > 0$, with probability at least $1 - \delta$, it holds that*

$$\text{error}(\bar{h}) \leq \text{error}(h^\star) + \frac{\text{Regret}_T(\mathcal{A})}{T} + \sqrt{\frac{8\log(\frac{2}{\delta})}{T}}.$$

In particular, for $T = O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta} + T_\varepsilon(\mathcal{A}))$, where $T_\varepsilon(\mathcal{A})$ is the integer T such that $\frac{\text{Regret}_T(\mathcal{A})}{T} \leq \varepsilon$, we have

$$\text{error}(\bar{h}) \leq \text{error}(h^*) + \varepsilon.$$

How general is the theorem above? In the previous chapters we have described and analyzed OCO algorithms with regret guarantees that behave asymptotically as $O(\sqrt{T})$ or better. This translates to sample complexity of $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ (see exercises), which is known to be tight for certain scenarios.

To prove this theorem we need some tools from probability theory, such as the concentration inequalities that we survey next.

9.2.1 Reminder: measure concentration and martingales

Let us briefly discuss the notion of a martingale in probability theory. For intuition, it is useful to recall the simple random walk. Let X_i be a Rademacher random variable which takes values

$$X_i = \begin{cases} 1, & \text{with probability} \quad \frac{1}{2} \\ \\ -1, & \text{with probability} \quad \frac{1}{2} \end{cases} .$$

A simple symmetric random walk is described by the sum of such random variables, depicted in figure 9.1. Let $X = \sum_{i=1}^{T} X_i$ be the position after T steps of this random walk. The expectation and variance of this random variable are $\mathbf{E}[X] = 0$, $\mathrm{Var}(X) = T$.

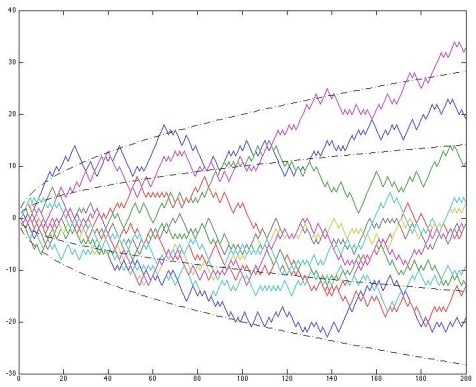

Figure 9.1: Symmetric random walk: 12 trials of 200 steps. The black dotted lines show the functions $\pm\sqrt{x}$ and $\pm 2\sqrt{x}$, respectively.

The phenomenon of measure concentration addresses the probability of a random variable to attain values within range of its standard deviation. For the random variable X, this probability is much higher than one would expect using only the first and second moments. Using only the variance, it follows from Chebychev's inequality that

$$\Pr\left[|X| \geq c\sqrt{T}\right] \leq \frac{1}{c^2}.$$

However, the event that $|X|$ is centred around $O(\sqrt{T})$ is in fact much tighter,

and can be bounded by the Hoeffding-Chernoff lemma as follows

$$\Pr[|X| \geq c\sqrt{T}] \leq 2e^{\frac{-c^2}{2}} \quad \text{Hoeffding-Chernoff lemma.} \tag{9.1}$$

Thus, deviating by a constant from the standard deviation decreases the probability exponentially, rather than polynomially. This well-studied phenomenon generalizes to sums of weakly dependent random variables and martingales, which are important for our application.

Definition 9.6. *A sequence of random variables $X_1, X_2, ...$ is called a* martingale *if it satisfies:*

$$\mathbf{E}[X_{t+1}|X_t, X_{t-1}...X_1] = X_t \quad \forall\, t > 0.$$

A similar concentration phenomenon to the random walk sequence occurs in martingales. This is captured in the following theorem by Azuma.

Theorem 9.7 (Azuma's inequality)**.** *Let $\left\{X_i\right\}_{i=1}^{T}$ be a martingale of T random variables that satisfy $|X_i - X_{i+1}| \leq 1$. Then:*

$$\Pr\left[|X_T - X_0| > c\right] \leq 2e^{\frac{-c^2}{2T}}.$$

By symmetry, Azuma's inequality implies,

$$\Pr\left[X_T - X_0 > c\right] = \Pr\left[X_0 - X_T > c\right] \leq e^{\frac{-c^2}{2T}}. \tag{9.2}$$

9.2.2 Analysis of the reduction

We are ready to prove the performance guarantee for the reduction in Algorithm 29. Assume for simplicity that the loss function ℓ is bounded in the interval $[0, 1]$, i.e.,

$$\forall \hat{y}, y \in \mathcal{Y}\ ,\ \ell(\hat{y}, y) \in [0, 1].$$

Proof of Theorem 9.5. We start by defining a sequence of random variables that form a martingale. Let

$$Z_t \stackrel{\text{def}}{=} \text{error}(h_t) - \ell(h_t(\mathbf{x}_t), y_t), \quad X_t \stackrel{\text{def}}{=} \sum_{i=1}^{t} Z_i.$$

Let us verify that $\{X_t\}$ is indeed a bounded martingale. Notice that by definition of error(h), we have that

$$\mathop{\mathbf{E}}_{(\mathbf{x},y)\sim\mathcal{D}}[Z_t|X_{t-1}] = \text{error}(h_t) - \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(h_t(\mathbf{x}), y)] = 0.$$

Thus, by the definition of Z_t,

$$\mathbf{E}[X_{t+1}|X_t, ...X_1] \quad = \mathbf{E}[Z_{t+1}|X_t] + X_t = X_t.$$

In addition, by our assumption that the loss is bounded, we have that (see exercises)

$$|X_t - X_{t-1}| = |Z_t| \leq 1. \tag{9.3}$$

Therefore we can apply Azuma's theorem to the martingale $\{X_t\}$, or rather its consequence (9.2), and get

$$\Pr[X_T > c] \leq e^{\frac{-c^2}{2T}}.$$

Plugging in the definition of X_T, dividing by T and using $c = \sqrt{2T\log(\frac{2}{\delta})}$:

$$\Pr\left[\frac{1}{T}\sum_{t=1}^{T}\text{error}(h_t) - \frac{1}{T}\sum_{t=1}^{T}\ell(h_t(\mathbf{x}_t), y_t) > \sqrt{\frac{2\log(\frac{2}{\delta})}{T}}\right] \leq \frac{\delta}{2}. \tag{9.4}$$

A similar martingale can be defined for $h^\star$ rather than h_t, and repeating the analogous definitions and applying Azuma's inequality we get:

$$\Pr\left[\frac{1}{T}\sum_{t=1}^{T}\text{error}(h^\star) - \frac{1}{T}\sum_{t=1}^{T}l(h^\star(\mathbf{x}_t), y_t) < -\sqrt{\frac{2\log(\frac{2}{\delta})}{T}}\right] \leq \frac{\delta}{2}. \tag{9.5}$$

For notational convenience, let us use the following notation:

$$\Gamma_1 = \frac{1}{T}\sum_{t=1}^{T}\text{error}(h_t) - \frac{1}{T}\sum_{t=1}^{T}\ell(h_t(\mathbf{x}_t), y_t),$$

$$\Gamma_2 = \frac{1}{T}\sum_{t=1}^{T}\text{error}(h^\star) - \frac{1}{T}\sum_{t=1}^{T}l(h^\star(\mathbf{x}_t), y_t).$$

Next, observe that

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\text{error}(h_t) - \text{error}(h^\star) \\
&= \Gamma_1 - \Gamma_2 + \frac{1}{T}\sum_{t=1}^{T}\ell(h_t(\mathbf{x}_t), y_t) - \frac{1}{T}\sum_{t=1}^{T}\ell(h^\star(\mathbf{x}_t), y_t) \\
&\leq \frac{\text{Regret}_T(\mathcal{A})}{T} + \Gamma_1 - \Gamma_2,
\end{aligned}$$

where in the last inequality we have used the definition $f_t(h) = \ell(h(\mathbf{x}_t), y_t)$. From the above and Inequalities (9.4), (9.5) we get

$$\begin{aligned}
&\Pr\left[\frac{1}{T}\sum_{t=1}^{T}\text{error}(h_t) - \text{error}(h^\star) > \frac{\text{Regret}_T(\mathcal{A})}{T} + 2\sqrt{\frac{2\log(\frac{2}{\delta})}{T}}\right] \\
&\leq \Pr\left[\Gamma_1 - \Gamma_2 > 2\sqrt{\frac{2\log(\frac{1}{\delta})}{T}}\right] \\
&\leq \Pr\left[\Gamma_1 > \sqrt{\frac{2\log(\frac{1}{\delta})}{T}}\right] + \Pr\left[\Gamma_2 \leq -\sqrt{\frac{2\log(\frac{1}{\delta})}{T}}\right] \\
&\leq \delta. \qquad \text{Inequalities (9.4), (9.5)}
\end{aligned}$$

By convexity we have that $\text{error}(\bar{h}) \leq \frac{1}{T}\sum_{t=1}^{T}\text{error}(h_t)$. Thus, with probability at least $1-\delta$,

$$\text{error}(\bar{h}) \leq \frac{1}{T}\sum_{t=1}^{T}\text{error}(h_t) \leq \text{error}(h^\star) + \frac{\text{Regret}_T(\mathcal{A})}{T} + \sqrt{\frac{8\log(\frac{2}{\delta})}{T}}.$$

□

9.3 Learning and Compression

Thus far we have considered finite and certain infinite hypothesis classes, and shown that they are efficiently learnable if there exists an efficient regret-minimization algorithm for a corresponding OCO setting.

In this section we describe yet another property which is sufficient for PAC learnability: the ability to compress the training set. This property is particularly easy to state and use, especially for infinite hypothesis classes. It does not, however, imply efficient algorithms.

Intuitively, if a learning algorithm is capable to express an hypothesis using a small fraction of the training set, we will show that it generalizes well to unseen data. For simplicity, we only consider learning problems that satisfy a variant of the realizablility assumption, i.e., the compression scheme generates a hypothesis that attains zero error.

More formally, we define the notion of a compression scheme for a given learning problem as follows. The definition and theorem henceforth can be generalized to allow for loss functions, but for simplicity, consider only the zero-one loss function for this section.

Definition 9.8. *(Compression Scheme) A distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ admits a compression scheme of size k, realized by an algorithm $\mathcal{A}$, if the following holds. For any $T > k$, let $S_T = \{(\mathbf{x}_t, y_t),\ t \in [T]\}$ be a sample from $\mathcal{D}$. There exists an $S' \subseteq S_T$, $|S'| = k$, such that the algorithm $\mathcal{A}$ accepts the set of k examples S', and returns a hypothesis $\mathcal{A}(S') \in \{\mathcal{X} \mapsto \mathcal{Y}\}$, which satsifies:*

$$\underset{S_T}{\text{error}}(\mathcal{A}(S')) = 0.$$

The main conclusion of this section is that a learning problem that admits a compression scheme of size k is PAC learnable with sample complexity proportional to k. This is formally given the following theorem.

Theorem 9.9. *Let $\mathcal{D}$ be a data distribution that admits a compression scheme of size k realized by algorithm $\mathcal{A}$. Then with probability at least $1 - \delta$ over the choice of a training set $|S_T| = T$, it holds that*

$$\text{error}(\mathcal{A}(S_T)) \leq \frac{8k \log \frac{T}{\delta}}{T}.$$

Proof. Denote by $\text{error}_S(h)$ the error of an hypothesis h on a sample S of i.i.d. examples, where the sample is taken independently of h. Since the examples are chosen independently, the probability that a hypothesis with $\text{error}(h) > \varepsilon$ has $\text{error}_S(h) = 0$ is at most $(1 - \varepsilon)^{|S|}$. Denote the event of h satisfying these two conditions by $h \in \text{bad}$.

Consider a compression scheme for distribution $\mathcal{D}$ of size k, realized by $\mathcal{A}$, and a sample of size $|S_T| = T \gg k$. By definition of a compression scheme, the hypothesis returned by $\mathcal{A}$ is based on k examples chosen from the set $S' \subseteq S_T$. We can bounds the probability of the event that $\text{error}_{S_T}(\mathcal{A}(S')) = 0$ and $\text{error}(\mathcal{A}(S')) > \varepsilon$, denoted by $\mathcal{A}(S') \in \text{bad}$, as follows,

$$\begin{aligned} &\Pr[\mathcal{A}(S') \in \text{bad}] \\ &= \textstyle\sum_{S' \subseteq S_T, |S'|=k} \Pr[\mathcal{A}(S') \in \text{bad}] \cdot \Pr[S'] \quad \text{law of total probability} \\ &\leq \tbinom{T}{k}(1 - \varepsilon)^T. \end{aligned}$$

For $\varepsilon = \frac{8k \log \frac{T}{\delta}}{T}$, we have that

$$\binom{T}{k}(1 - \varepsilon)^T \leq T^k e^{-\varepsilon T} \leq \delta.$$

Since the compression scheme is guaranteed to return a hypothesis such that $\text{error}_{S_T}(\mathcal{A}(S')) = 0$, this implies that with probability at least $1 - \delta$, the hypothesis $\mathcal{A}(S')$ has $\text{error}(\mathcal{A}(S')) \leq \varepsilon$. □

An important example of the use of compression schemes to bound the generalization error is for the hypothesis class of hyperplanes in $\mathbb{R}^d$. It is left as an exercise to show that this hypothesis class admits a compression scheme of size d.

9.4 Bibliographic Remarks

The foundations of statistical and computational learning theory were put forth in the seminal works of Vapnik [1998] and Valiant [1984] respectively. There are numerous comprehensive texts on statistical and computational learning theory, see e.g., [Kearns and Vazirani, 1994], and the recent text [Shalev-Shwartz and Ben-David, 2014].

Reductions from the online to the statistical (a.k.a. "batch") setting were initiated by Littlestone [Littlestone, 1989]. Tighter and more general bounds were explored in [Cesa-Bianchi et al., 2006, Cesa-Bianchi and Gentile, 2008, Zhang, 2005].

The probabilistic method is attributed to Paul Erdos, see the illuminating text of Alon and Spencer [Alon and Spencer, 1992].

The relationship between compression and PAC learning was studied in the seminal work of Littlestone and Warmuth [1986]. For more on the relationship and historical connections between statistical learning and compression see the inspiring chapter in [Wigderson, 2019]. More recently Moran and Yehudayoff [2016], David et al. [2016] show that compression is equivalent to learnability in general supervised learning tasks and give quantitative bounds for this relationship.

The use of compression for proving generalization error bounds has been applied in [Hanneke et al., 2019] for regression and in [Gottlieb et al., 2018, Kontorovich et al., 2017] for nearest neighbor classification. Another application is the recent work of Bousquet et al. [2020] which gives optimal generalization error bounds for support vector machines using compression.

9.5 Exercises

1. Strengthen the no free lunch Theorem 9.1 to show the following: For any $\varepsilon > 0$, there exists a finite domain $\mathcal{X}$, such that for any learning algorithm $\mathcal{A}$ which given a sample S produces hypothesis $\mathcal{A}(S)$, there exists a distribution D and a concept $C : X \mapsto \{0, 1\}$ such that

(a) $\text{error}(C) = 0$

(b) $\mathbf{E}_{S\sim\mathcal{D}^m}[\text{error}(\mathcal{A}(S))] \geq \frac{1}{2} - \varepsilon$.

2. Let $\mathcal{A}$ be an agnostic learning algorithm for the finite hypothesis class $\mathcal{H} : \mathcal{X} \mapsto \mathcal{Y}$ and the zero-one loss. Consider any concept $C : X \mapsto Y$ which is realized by $\mathcal{H}$, and the concept $\hat{C}$ which is obtained by replacing the label associated with each domain entry $x \in X$ randomly with probability $\varepsilon_0 > 0$ every time x is sampled independently. That is:

$$\hat{C}(x) = \begin{cases} 1, & \text{with probability } \frac{\varepsilon_0}{2} \\ 0, & \text{with probability } \frac{\varepsilon_0}{2} \\ C(x), & \text{otherwise} \end{cases}$$

Prove that $\mathcal{A}$ can ε-approximate the concept $\hat{C}$: that is, show that $\mathcal{A}$ can be used to produce a hypothesis $h_{\mathcal{A}}$ that has error

$$\underset{D}{\text{error}}(h_{\mathcal{A}}) \leq \frac{1}{2}\varepsilon_0 + \varepsilon$$

with probability at least $1 - \delta$ for every ε, δ with sample complexity polynomial in $\frac{1}{\varepsilon}, \log\frac{1}{\delta}, \log|H|$.

3. Prove inequality 9.3.

4. (Sample complexity of SVM)
Consider the class of hypothesis given by hyperplanes in Euclidean space with bounded norm

$$\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^d \ , \ \|\mathbf{x}\|_2 \leq \lambda\}.$$

Give an algorithm to PAC-learn this class with respect to the hinge loss function using reduction (29). Analyze the resulting computational and sample complexity.

5. Show how to use a modification of reduction 29 to learn a finite (non-convex) hypothesis class efficiently, i.e., without enumerating over all hypothesis. For this

question, success probability of $\frac{1}{2}$ is sufficient.
Hint: instead of returning $\bar{h}$, consider returning a hypothesis at random.

6. Consider the hypothesis class of all axis-aligned rectangles in the plane. That is, consider all hypothesis parametrized by four real numbers, a_x, b_x, a_y, b_y, such that

$$h_{a_x,b_x,a_y,b_y}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x}_1 \in [a_x, b_x], \mathbf{x}_2 \in [a_y, b_y] \\ 0, & o/w \end{cases} .$$

Prove that this hypothesis class admits a compression scheme of size 4.

7. * Consider the hypothesis class of all hyperplanes in $\mathbb{R}^d$. This class is parameterized by all vectors in the unit sphere, such that

$$\forall \mathbf{y} \in \mathbb{S}_d \ , \ h_{\mathbf{y}}(\mathbf{x}) = \text{sign}(\mathbf{x}^\top \mathbf{y}).$$

Prove that this class has a compression scheme of size d.

Chapter 10

Learning in Changing Environments

In online convex optimization the decision maker iteratively makes a decision without knowledge of the future, and pays a cost based on her decision and the observed outcome. The algorithms that we have studied thus far are designed to perform nearly as well as the best single decision in hindsight. The performance metric we have advocated for, average *regret* of the online player, approaches zero as the number of game iterations grows.

In scenarios in which the outcomes are sampled from some (unknown) distribution, regret minimization algorithms effectively "learn" the environment and approach the optimal strategy. This was formalized in chapter 9. However, if the underlying distribution changes, no such claim can be made.

Consider for example the online shortest path problem we have studied in the first chapter. It is a well observed fact that traffic in networks exhibits changing cyclic patterns. A commuter may choose one path from home to work on a weekday, but a completely different path on the weekend when traffic patterns are different. Another example is the stock market: in a bull market the investor may want to purchase technology stocks, but in a bear market perhaps they would shift their investments to gold or government bonds.

When the environment undergoes many changes, standard regret may not be the best measure of performance. In changing environments, the online convex optimization algorithms we have studied thus far for strongly convex or exp-concave loss functions exhibit undesirable "static" behavior, and converge to a fixed solution.

In this chapter we introduce and study a generalization of the concept

of regret called *adaptive* regret, to allow for a changing prediction strategy. We start with examining the notion of adapting in the problem of prediction from expert advice. We then continue to the more challenging setting of online convex optimization, and derive efficient algorithms for minimizing this more refined regret metric.

10.1 A Simple Start: Dynamic Regret

Before giving the main performance metric studied in this chapter, we consider the first natural approach: measuring regret w.r.t. any sequence of decisions. Clearly, in general it is impossible to compete with an arbitrary changing benchmark. However, it is possible to give a refined analysis that shows what happens to the regret of an online convex optimization algorithm vs. changing decisions.

More precisely, define the *dynamic regret* of an OCO algorithm with respect to a sequence $\mathbf{u}_1, \ldots, \mathbf{u}_T$ as:

$$\text{DynamicRegret}_T(\mathcal{A}, \mathbf{u}_1, \ldots, \mathbf{u}_T) \stackrel{\text{def}}{=} \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{u}_t)$$

To analyze the dynamic regret, some measure of the complexity of the sequence $\mathbf{u}_1, \ldots, \mathbf{u}_T$ is necessary. Let $\mathcal{P}(\mathbf{u}_1, \ldots, \mathbf{u}_T)$ be the path length of the comparison sequence defined as

$$\mathcal{P}(\mathbf{u}_1, \ldots, \mathbf{u}_T) = \sum_{t=1}^{T-1} \|\mathbf{u}_t - \mathbf{u}_{t+1}\| + 1.$$

It is natural to expect the regret to scale with the path length, as indeed the following theorem shows. For a fixed comparator $\mathbf{u}_t = \mathbf{x}^\star$, the path length is one, and thus Theorem 10.1 recovers the $O(\sqrt{T})$ standard regret bound. For simplicity, we assume that the time horizon T is known ahead of time, and so is the path length of the comparator sequence, although these limitations can be removed (see bibliographic section).

Theorem 10.1. *Online Gradient Descent (algorithm 8) with step size* $\eta \approx \sqrt{\frac{\mathcal{P}(\mathbf{u}_1,\ldots,\mathbf{u}_T)}{T}}$ *guarantees the following dynamic regret bound:*

$$\text{DynamicRegret}_T(\mathcal{A}, \mathbf{u}_1, \ldots, \mathbf{u}_T) = O(\sqrt{T\mathcal{P}(\mathbf{u}_1, \ldots, \mathbf{u}_T)})$$

Proof. Using our notation, and following the steps of the proof of Theorem 3.1,

$$\|\mathbf{x}_{t+1} - \mathbf{u}_t\|^2 \ \leq \|\mathbf{y}_{t+1} - \mathbf{u}_t\|^2 = \|\mathbf{x}_t - \mathbf{u}_t\|^2 + \eta^2\|\nabla_t\|^2 - 2\eta\nabla_t^\top(\mathbf{x}_t - \mathbf{u}_t).$$

Thus,

$$2\nabla_t^\top(\mathbf{x}_t - \mathbf{u}_t) \quad \leq \tfrac{\|\mathbf{x}_t - \mathbf{u}_t\|^2 - \|\mathbf{x}_{t+1} - \mathbf{u}_t\|^2}{\eta} + \eta G^2$$

Using convexity and summing this inequality across time we get

$$\begin{aligned}
&2\left(\sum_{t=1}^T f_t(\mathbf{x}_t) - f_t(\mathbf{u}_t)\right) \leq 2\sum_{t=1}^T \nabla_t^\top(\mathbf{x}_t - \mathbf{x}^\star) \\
&\leq \sum_{t=1}^T \frac{\|\mathbf{x}_t - \mathbf{u}_t\|^2 - \|\mathbf{x}_{t+1} - \mathbf{u}_t\|^2}{\eta} + \eta G^2 T \\
&= \frac{1}{\eta}\sum_{t=1}^T \Big(\|\mathbf{x}_t\|^2 - \|\mathbf{x}_{t+1}\|^2 + 2\mathbf{u}_t^\top(\mathbf{x}_{t+1} - \mathbf{x}_t)\Big) + \eta G^2 T \\
&\leq \frac{2}{\eta}\left(D^2 + \sum_{t=2}^T \mathbf{x}_t^\top(\mathbf{u}_{t-1} - \mathbf{u}_t) + \mathbf{u}_T^\top\mathbf{x}_{T+1} - \mathbf{u}_1^\top\mathbf{x}_1\right) + \eta G^2 T \\
&\leq \frac{3}{\eta}\left(D^2 + D\sum_{t=2}^T \|\mathbf{u}_{t-1} - \mathbf{u}_t\|\right) + \eta G^2 T \qquad\qquad \mathbf{u}_t \in \mathcal{K} \\
&\leq \frac{3D^2}{\eta}\mathcal{P}(\mathbf{u}_1, ..., \mathbf{u}_T) + \eta G^2 T.
\end{aligned}$$

The theorem now follows by choice of η. □

This simple modification to the analysis of online gradient descent naturally extends to online mirror descent, as well as to other notions of path distance of the comparison sequence.

We now turn to another metric of performance that requires more advanced methods than we have seen thus far. This metric can be shown to be more general than dynamic regret, in the sense that the bounds we prove also imply low dynamic regret.

10.2 The Notion of Adaptive Regret

The main performance metric we consider in this chapter is designed to measure the performance of a decision maker in a changing environment. It is formally given in the following definition.

Definition 10.2. *The adaptive regret of an online convex optimization algorithm $\mathcal{A}$ is defined as the maximum regret it achieves over any contiguous time interval. Formally,*

$$\begin{aligned}\text{AdaptiveRegret}_T(\mathcal{A}) &\stackrel{def}{=} \sup_{I=[r,s]\subseteq[T]} \left\{ \sum_{t=r}^{s} f_t(\mathbf{x}_t) - \min_{x_I^* \in \mathcal{K}} \sum_{t=r}^{s} f_t(\mathbf{x}_I^*) \right\} \\ &= \sup_{I=[r,s]\subseteq[T]} \left\{ \text{Regret}_{[r,s]}(\mathcal{A}) \right\}.\end{aligned}$$

As opposed to standard regret, the power of this definition stems from the fact that the comparator is allowed to change. In fact, it is allowed to change indefinitely with every interval of time.

For an algorithm with low adaptive regret, as opposed to standard regret, how would its performance guarantee differ in a changing environment? Consider the problem of portfolio selection, for which time can be divided into disjoint segments with different characteristics: bear market in the first $T/2$ iterations and bull market in the last $T/2$ iterations. A (standard) sublinear regret algorithm is only required to converge to the average of both optimal portfolios, clearly an undesirable outcome. However, an algorithm with sublinear adaptive regret bounds would *necessarily* converge to the optimal portfolio in both intervals.

Not only does this definition make intuitive sense, but it generalizes other natural notions. For example, consider an OCO setting that can be divided into k intervals, such that in each a different comparator is optimal. Then an adaptive regret guarantee of $\text{AdaptiveRegret}_T = o(T)$ would translate to overall regret of $k \times \text{AdaptiveRegret}_{T/k}$ compared to the best k-shifting comparator[14].

10.2.1 Weakly and strongly adaptive algorithms

The Online Gradient Descent algorithm over general convex losses, with step sizes $O(\frac{1}{\sqrt{t}})$, attains an adaptive regret guarantee of

$$\text{AdaptiveRegret}_T(OGD) = O(\sqrt{T}),$$

and this bound is tight. This is a simple consequence of the analysis we have already seen in chapter 3, and left as an exercise. Unfortunately this guarantee is meaningless for intervals of length $o(\sqrt{T})$.

Recall that for strongly convex loss functions, the OGD algorithm with the optimal learning rate schedule attains $O(\log T)$ regret. However, it does

not attain any non-trivial adaptive regret guarantee: its adaptive regret can be as large as $\Omega(T)$, and this is also left as an exercise.

An OCO algorithm $\mathcal{A}$ is said to be *strongly adaptive* if its adaptive regret can be bounded by its regret over the interval up to logarithmic terms in T, i.e.

$$\text{AdaptiveRegret}_T(\mathcal{A}) = O(\text{Regret}_I(\mathcal{A}) \cdot \log^{O(1)} T).$$

The natural question is thus: are there algorithms that attain the optimal regret guarantee, and simultaneously the optimal adaptive regret guarantee? As we shall see, the answer is affirmative in a strong sense: we shall describe and analyze algorithms that are optimal in both metrics. Furthermore, these algorithms can be implemented with small computational overhead over the non-adaptive methods we have already studied.

10.3 Tracking the Best Expert

Consider the fundamental problem studied in the first chapter of this text, prediction from expert advice, but with a small twist. Instead of a static best expert, consider the setting in which different experts are the "best expert" in different time intervals. More precisely, consider the situation in which time $[T]$ can be divided into k disjoint intervals such that each admits a different "locally best" expert. Can we learn to track the best expert?

This tracking problem was historically the first motivation to study adaptivity in online learning. Indeed, as shown by Herbster and Warmuth (see bibliographic section), there is a natural algorithm that attains optimal regret bounds.

The Fixed Share algorithm, describe in Algorithm 30, is a variant of the Hedge Algorithm 1. On top of the familiar multiplicative updates, it adds a uniform exploration term whose purpose is to avoid the weight of any expert from becoming too small. This provably allows a regret bound that tracks the best expert in any interval.

In line with the notation we have used throughout this manuscript, we denote decisions in a convex decision set by $\mathbf{x} \in \mathcal{K}$. An expert i suggests decision $\mathbf{x}_t^i$, and suffers loss according to a convex loss function denoted $f_t(\mathbf{x}_t^i)$. The main performance guarantee for the Fixed Share algorithm is given in the theorem below.

Algorithm 30 Fixed Share

Input: parameter $\delta < \frac{1}{2}$. Initialize $\forall i \in [N], p_i^1 = \frac{1}{N}$.
for $t = 1$ to T **do**
  Play $\mathbf{x}_t = \sum_{i=1}^N p_t^i \mathbf{x}_t^i$.
  After receiving f_t, update for $1 \le i \le N$

$$\hat{p}_{t+1}^i = \frac{p_t^i e^{-\alpha f_t(\mathbf{x}_t^i)}}{\sum_{j=1}^N p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}}$$

  Fixed-share step:

$$p_{t+1}^i = (1-\delta)\hat{p}_{t+1}^i + \frac{\delta}{N}$$

end for

Theorem 10.3. *Given a sequence of α-exp-concave loss functions, the Fixed-Share algorithm with $\delta = \frac{1}{2T}$ guarantees*

$$\sup_{I=[r,s]\subseteq[T]} \left\{ \sum_{t=r}^{s} f_t(\mathbf{x}_t) - \min_{i^* \in [N]} \sum_{t=r}^{s} f_t(\mathbf{x}_t^{i^*}) \right\} = O\left(\frac{1}{\alpha} \log NT\right).$$

Notice that this is a different guarantee than adaptive regret as per Definition 10.2, as the decision set is discrete. However, it is a crucial component in the adaptive algorithms we will explore in the next section.

As a direct conclusion from this theorem, it can be shown (see exercises) that if the best expert changes k times in a sequence of length T, the overall regret compared to the best expert in every interval is bounded by

$$O\left(k \log \frac{NT}{k}\right).$$

To prove this theorem, we start with the following lemma, which is a fine-grained analysis of the multiplicative weights properties:

Lemma 10.4. *For all $1 \le i < N$,*

$$f_t(\mathbf{x}_t) - f_t(\mathbf{x}_t^i) \le \alpha^{-1}(\log \hat{p}_{t+1}^i - \log \hat{p}_t^i - \log(1-\delta)).$$

Proof. Using the α-exp concavity of f_t,

$$e^{-\alpha f_t(\mathbf{x}_t)} \quad = \quad e^{-\alpha f_t(\sum_{j=1}^N p_t^j \mathbf{x}_t^j)} \ge \sum_{j=1}^N p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}.$$

Taking the natural logarithm,

$$f_t(\mathbf{x}_t) \leq -\alpha^{-1} \log \sum_{j=1}^{N} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}$$

Hence,

$$\begin{aligned} f_t(\mathbf{x}_t) - f_t(\mathbf{x}_t^i) &\leq \alpha^{-1}(\log e^{-\alpha f_t(\mathbf{x}_t^i)} - \log \textstyle\sum_{j=1}^{N} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}) \\ &= \alpha^{-1} \log \frac{e^{-\alpha f_t(\mathbf{x}_t^i)}}{\sum_{j=1}^{N} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}} \\ &= \alpha^{-1} \log \left(\frac{1}{p_t^i} \cdot \frac{p_t^i e^{-\alpha f_t(\mathbf{x}_t^i)}}{\sum_{j=1}^{N} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}} \right) \\ &= \alpha^{-1} \log \frac{\hat{p}_{t+1}^i}{p_t^i} = \alpha^{-1}(\log \hat{p}_{t+1}^i - \log p_t^i) \end{aligned}$$

The proof is completed observing that:

$$\begin{aligned} \log p_t^i &= \log \left((1-\delta)\hat{p}_t^i + \tfrac{\delta}{N} \right) \\ &\geq \log \hat{p}_t^i + \log(1-\delta). \end{aligned}$$

□

Theorem 10.3 can now be derived as a corollary:

Theorem 10.3. By summing up over the interval $I = [r, s]$, and using the lower bound on p_t^i, we have

$$\begin{aligned} & \textstyle\sum_{t \in I} f_t(\mathbf{x}_t) - \sum_{t \in I} f_t(\mathbf{x}_t^i) \\ \leq & \textstyle\sum_{t \in I} \alpha^{-1}(\log \hat{p}_{t+1}^i - \log \hat{p}_t^i - \log(1-\delta)) \\ \leq & \tfrac{1}{\alpha} \left[\log \tfrac{1}{\hat{p}_r^i} - |I| \log(1-\delta) \right] \\ \leq & \tfrac{1}{\alpha} \left[\log \tfrac{N}{\delta} + 2\delta |I| \right] && \hat{p}_r^i \geq \frac{\delta}{N}, \delta < \frac{1}{2} \\ \leq & \tfrac{1}{\alpha} \log 2NT + \tfrac{1}{\alpha} && \delta = \frac{1}{2T} \end{aligned}$$

□

10.4 Efficient Adaptive Regret for Online Convex Optimization

The Fixed-Share algorithm described in the previous section is extremely practical and efficient for discrete sets of experts. However, to exploit the full power of OCO we require an efficient algorithm for continuous decision sets.

Consider for example the problems of online portfolio selection and online shortest paths: naïvely applying the Fixed-Share algorithm is computationally inefficient. Instead, we seek an algorithm which takes advantage of the efficient representation of these problems in the language of convex programming.

We present such a method called FLH, or Follow the Leading History. The basic idea is to think of different online convex optimization algorithms starting at different time points as experts, and apply a version of Fixed Share to these experts.

Algorithm 31 Follow the Leading History

Let $\mathcal{A}$ be an OCO algorithm. Initialize $p_1^1 = 1$
for $t = 1$ to T **do**
  Set $\forall j \leq t$, $\mathbf{x}_t^j \leftarrow \mathcal{A}(f_j, ..., f_{t-1})$
  Play $\mathbf{x}_t = \sum_{j=1}^{t} p_t^j \mathbf{x}_t^j$.
  After receiving f_t, update for $1 \leq i \leq t$

$$\hat{p}_{t+1}^i = \frac{p_t^i e^{-\alpha f_t(\mathbf{x}_t^i)}}{\sum_{j=1}^{t} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}}$$

  Mixing step: set $p_{t+1}^{t+1} = \frac{1}{t+1}$ and

$$\forall i \neq t+1 \ , \ p_{t+1}^i = \left(1 - \frac{1}{t+1}\right) \hat{p}_{t+1}^i.$$

end for

The main performance guarantee is given in the following theorem.

Theorem 10.5. *Let $\mathcal{A}$ be an OCO algorithm for α-exp-concave loss function with* $\text{Regret}_T(\mathcal{A})$*. Then,*

$$\text{AdaptiveRegret}_T(FLH) \leq \text{Regret}_T(\mathcal{A}) + O(\frac{1}{\alpha} \log T).$$

In particular, taking $\mathcal{A} \equiv ONS$ guarantees

$$\text{AdaptiveRegret}_T = O(\frac{1}{\alpha}\log T).$$

Notice that FLH invokes $\mathcal{A}$ at iteration t at most T times. Hence its running time is bounded by T times that of $\mathcal{A}$. This can still be prohibitive as the number of iterations grows large. In the next section, we show how the ideas from this algorithm can give rise to an efficient adaptive algorithm with only $O(\log T)$ computational overhead and slightly worse regret bounds.

The analysis of FLH is very similar to that of Fixed Share, with the main subtleties due to the fact that the time horizon T is not assumed to be known ahead of time, and thus the number of experts varies with time.

Instead of giving the full analysis, which is deferred to the exercises, we give a simplified version of FLH which does assume a-priory knowledge of T, and whose analysis can be directly reduced to that of Theorem 10.3.

Algorithm 32 Simple-FLH

1: Let $\mathcal{A}$ be an OCO algorithm. Set $N = T, \delta = \frac{1}{2T}$.
2: **for** $t = 1$ to T **do**
3: For all $i \le t$, set $\mathbf{x}_t^i \leftarrow \mathcal{A}(f_j, ..., f_{t-1})$. Otherwise, set $\mathbf{x}_t^i = \mathbf{0}$.
4: Apply the Fixed Share algorithm with expert predictions $\mathbf{x}_t^i$.
5: **end for**

The simplified version of FLH is given in Algorithm 32, and it guarantees the following adaptive regret bound.

Theorem 10.6. *Algorithm 32 guarantees:*

$$\text{AdaptiveRegret}_T(\textit{Simple-FLH}) \le \text{Regret}_T(\mathcal{A}) + O(\frac{1}{\alpha}\log T).$$

Proof. Applying Theorem 10.3 to the experts defined in Simple FLH, guarantees for every interval in time $I = [r, s]$, and by choice of N, for every $i \le s$,

$$\sum_{t\in I} f_t(\mathbf{x}_t) - \sum_{t\in I} f_t(\mathbf{x}_t^i) \le \frac{1}{\alpha}\log 2NT + 1 = O(\frac{1}{\alpha}\log T).$$

In particular, consider the sequence of predictions given by the r'th expert, for which we have

$$\sum_{t\in I} f_t(\mathbf{x}_t^r) = \text{Regret}_{s-r+1}(\mathcal{A}) \le \text{Regret}_T(\mathcal{A}).$$

The theorem now follows since this holds for every iterval $I \subseteq [T]$. □

10.5 * Computationally Efficient Methods

In the previous section we studied adaptive regret, introduced and analyzed an algorithm that attains near optimal adaptive regret bounds. However, FLH suffers from a significant computational and memory overhead: it requires maintaining $O(T)$ copies of an online convex optimization algorithm. This computational overhead, which is proportional to the number of iterations, can be prohibitive in many applications. In this section our goal is to implement the algorithmic template of FLH efficiently and using little space.

To be more precise, henceforth denote the running time per iteration of algorithm $\mathcal{A}$ as $V_t(\mathcal{A})$. Recall that at time t, FLH stores all predictions $\{\mathbf{x}_t^i \mid i \in [t]\}$ and has to compute weights for all of them. This requires running time of at least $O(V_t(\mathcal{A}) \cdot t)$.

The FLH2 algorithm, described in Algorithm 33, significantly cuts down this running time to being only logarithmic in the current time iteration parameter t. To achieve this, FLH2 applies a pruning method to cut down the number of active online algorithms from t to $O(\log t)$. However, its adaptive regret guarantee is slightly worse, and suffers a multiplicative factor of $O(\log T)$ as compared to FLH.

Algorithm 33 FLH2

Let $\mathcal{A}$ be an OCO algorithm. Initialize $p_1^1 = 1, S_1 = \{1\}$
for $t = 1$ to T **do**
  Set $\forall j \in S_t$, $\mathbf{x}_t^j \leftarrow \mathcal{A}(f_j, ..., f_{t-1})$
  Play $\mathbf{x}_t = \sum_{j \in S_t} p_t^j \mathbf{x}_t^j$.
  After receiving f_t, perform update for $i \in S_t$:

$$\hat{p}_{t+1}^i = \frac{p_t^i e^{-\alpha f_t(\mathbf{x}_t^i)}}{\sum_{j \in S_t} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}}$$

  Pruning: set $S_{t+1} \leftarrow \text{Prune}(S_t) \cup \{t+1\}$. Set $\hat{p}_{t+1}^{t+1}$ to $\frac{1}{t}$, and update:

$$\forall i \in S_{t+1} \ . \ p_{t+1}^i = \frac{\hat{p}_{t+1}^i}{\sum_{j \in S_{t+1}} \hat{p}_{t+1}^j}$$

end for

Before giving the exact details of this pruning method, we state the performance guarantee for FLH2.

Theorem 10.7. *Given an OCO algorithm $\mathcal{A}$ with regret* $\text{Regret}_T(\mathcal{A})$ *and running time $V_T(A)$, algorithm $FLH2$ guarantees: $V_T(FLH2) \leq V_T(\mathcal{A}) \log T$ and*

$$\text{AdaptiveRegret}_T(FLH2) \leq \text{Regret}_T(\mathcal{A}) \log T + O(\frac{1}{\alpha} \log^2 T).$$

The main conclusion from this theorem is obtained by using FLH2 with $\mathcal{A}$ being the ONS algorithm from chapter 4. This gives adaptive regret of $O(\frac{1}{\alpha} \log^2 T)$ and running time which is polynomial in natural parameters of the problem and poly-logarithmic in the number of iterations.

Before diving into the analysis, we explain the main new ingredient. At the heart of this algorithm is a new method for incorporating history. We will show that it suffices to store only $O(\log t)$ experts at time t, rather than all t experts as in FLH.

At time t, there is a working set S_t of experts. In FLH, this set can be thought of to contain $E^1, \cdots, E^t$, where each E^i is the algorithm $\mathcal{A}$ starting from iteration i. For the next round, a new expert E^{t+1} is added to get S_{t+1}. The complexity and regret of FLH is directly related to the cardinality of these sets.

The key to decreasing the sizes of the sets S_t is to also *remove* (or *prune*) some experts. Once an expert is removed, it is never used again. The algorithm will perform the multiplicative update and mixing steps (steps 5 and 7 in algorithm 33) only on the working set of experts.

The problem of maintaining the set of active experts can be thought of as the following abstract data streaming problem. Suppose the integers $1, 2, \cdots$ are being "processed" in a streaming fashion. At time t, we have "read" the positive integers up to t and maintain a very small subset of them in S_t. At time t we create S_{t+1} from S_t: we are allowed to add to S_t only the integer $t+1$, and remove some integers already in S_t. Our aim is to maintain a set S_t which satisfies:

1. For every positive $s \leq t$, $[s, (s+t)/2] \cap S_t \neq \emptyset$.
2. For all t, $|S_t| = O(\log T)$.
3. For all t, $S_{t+1} \backslash S_t = \{t+1\}$.

The first property of the sets S_t intuitively means that S_t is "well spread out" in a logarithmic scale. This is depicted in Figure 10.1. The second property ensures computational efficiency.

Indeed, the procedure "Prune" maintains S_t with these exact properties, and is detailed after we prove Theorem 10.7.

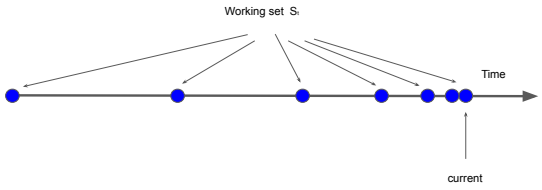

Figure 10.1: Illustration of the working set S_t

We proceed to prove the main theorem. We start with an analogue of Lemma 10.4.

Proposition 10.8. *The following holds for all $i \in S_t$,*

1. $f_t(\mathbf{x}_t) - f_t(\mathbf{x}_t^i) \leq \alpha^{-1}(\log \hat{p}_{t+1}^i - \log \hat{p}_t^i + \log \frac{t-1}{t})$
2. $f_t(\mathbf{x}_t) - f_t(\mathbf{x}_t^t) \leq \alpha^{-1}(\log \hat{p}_{t+1}^t + \log t)$

Proof. Using the α-exp concavity of f_t -

$$e^{-\alpha f_t(\mathbf{x}_t)} = e^{-\alpha f_t(\sum_{j \in S_t} p_t^j \mathbf{x}_t^j)} \geq \sum_{j \in S_t} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}$$

Taking the natural logarithm,

$$f_t(\mathbf{x}_t) \leq -\alpha^{-1} \log \sum_{j \in S_t} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}$$

Hence,

$$\begin{aligned} f_t(\mathbf{x}_t) - f_t(\mathbf{x}_t^i) &\leq \alpha^{-1}(\log e^{-\alpha f_t(\mathbf{x}_t^i)} - \log \textstyle\sum_{j \in S_t} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}) \\ &= \alpha^{-1} \log \frac{e^{-\alpha f_t(\mathbf{x}_t^i)}}{\sum_{j \in S_t} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}} \\ &= \alpha^{-1} \log \left(\frac{1}{p_t^i} \cdot \frac{p_t^i e^{-\alpha f_t(\mathbf{x}_t^i)}}{\sum_{j \in S_t} p_t^j e^{-\alpha f_t(\mathbf{x}_t^j)}} \right) \\ &= \alpha^{-1} \log \frac{\hat{p}_{t+1}^i}{p_t^i} \end{aligned}$$

To complete the proof, we note the following two facts that are analogous to the ones used in Claim 10.4:

1. For $1 \leq i < t$, $\log p_t^i \geq \log \hat{p}_t^i + \log \frac{t-1}{t}$
2. $\log p_t^t \geq -\log t$

Proving these facts is left as an exercise.

□

Using this we can prove the following Lemma.

Lemma 10.9. *Consider some time interval $I = [r, s]$. Suppose that E^r was in the working set S_t, for all $t \in I$. Then the regret incurred in I is at most $\frac{1}{\alpha}\log(s) + \text{Regret}_T(\mathcal{A})$.*

Proof. Consider the regret in I with respect to expert E^r,

$$\begin{aligned}
&\textstyle\sum_{t=r}^{s}(f_t(\mathbf{x}_t) - f_t(\mathbf{x}_t^r)) \\
&= (f_r(\mathbf{x}_r) - f_r(\mathbf{x}_r^r)) + \textstyle\sum_{t=r+1}^{s}(f_t(\mathbf{x}_t) - f_t(\mathbf{x}_t^r)) \\
&\leq \alpha^{-1}\big(\log \hat{p}_{r+1}^r + \log r + \textstyle\sum_{t=r+1}^{s}(\log \hat{p}_{t+1}^r - \log \hat{p}_t^r + \log \frac{t}{t-1})\big) \quad \text{Claim 10.8} \\
&= \alpha^{-1}(\log r + \log \hat{p}_{s+1}^r + \textstyle\sum_{t=r+1}^{s} \log \frac{t}{t-1}) \\
&= \alpha^{-1}(\log(s) + \log \hat{p}_{s+1}^r)
\end{aligned}$$

Since $\hat{p}_{s+1}^r \leq 1$, $\log \hat{p}_{s+1}^r \leq 0$. This implies that the regret w.r.t. expert E^r is bounded by $\alpha^{-1}\log(s)$. Since E^r has regret bounded by $\text{Regret}_I(\mathcal{A}) \leq \text{Regret}_T(\mathcal{A})$ over I, the conclusion follows. □

Given the properties of S_t, we can show that in any interval the regret incurred is small.

Lemma 10.10. *For any interval I the regret incurred by the FLH2 is at most*
$(\frac{1}{\alpha}\log(s) + \text{Regret}_T(\mathcal{A}))(\log_2 |I| + 1)$.

Proof. Let $|I| \in [2^q, 2^{q+1})$, and denote for simplicity $R_T = \frac{1}{\alpha}\log(s) + \text{Regret}_T(\mathcal{A})$. We will prove by induction on q.

base case: For $q = 0$ the regret is bounded by

$$f_r(\mathbf{x}_r) \leq \text{Regret}_T(\mathcal{A}) \leq R_T$$

induction step: By the properties of the S_t's, there is an expert E^i in the pool such that $i \in [r, (r+s)/2]$. This expert E^i entered the pool at time i and stayed throughout $[i, s]$. By Lemma 10.9, the algorithm incurs regret at most $R_T = \frac{1}{\alpha}\log(s) + \text{Regret}_T(\mathcal{A})$ in $[i, s]$.

The interval $[r, i-1]$ has size at most $\frac{|I|}{2} \in [2^{q-1}, 2^q)$, and by induction the algorithm has regret of at most $R_T \cdot q$ on this interval. This gives a total of $R_T(q+1)$ regret on I. □

We can now prove Theorem 10.7:

Theorem 10.7. The running time of FLH2 is bounded by $|S_t| \cdot V_T(\mathcal{A})$. Since $|S_t| = O(\log t)$, we can bound the running time by $O(V_T(\mathcal{A}) \log T)$. This fact, together with Lemma 10.10, completes the proof. □

10.5.1 The pruning method

We now explain the pruning procedure used to maintain the set $S_t \subseteq \{1, 2, ..., t\}$.

We specify the *lifetime* of integer i - if $i = r2^k$, where r is odd, then the lifetime of i is $2^{k+2} + 1$. Suppose the lifetime of i is m. Then for any time $t \in [i, i + m]$, integer i is *alive* at t. The set S_t is simply the set of all integers that are alive at time t. Obviously, at time t, the only integer added to S_t is t - this immediately proves Property (3). We now prove the other properties.

Proof. (Property (1)) We need to show that some integer in $[s, (s + t)/2]$ is alive at time t. This is trivially true when $t - s < 2$, since $t - 1, t \in S_t$. Let 2^ℓ be the largest power of 2 such that $2^\ell \leq (t - s)/2$. There is some integer $x \in [s, (s + t)/2]$ such that $2^\ell | x$. The lifetime of x is larger than $2^\ell \times 2 + 1 > t - s$, so x is alive at t. □

Proof. (Property (2)) For each $0 \leq k \leq \lfloor \log t \rfloor$, let us count the number of integers of the form $r2^k$ (r odd) alive at t. The lifetimes of these integers are $2^{k+2} + 1$. The only integers alive lie in the interval $[t - 2^{k+2} - 1, t]$. Since all of these integers of this form are separated by gaps of size at least 2^k, there are at most a constant number of such integers alive at t. In total, the size of S_t is $O(\log t)$. □

10.6 Bibliographic Remarks

Dynamic regret bounds for online gradient descent were proposed by Zinkevich [2003], and further studied in [Besbes et al., 2015]. It was shown in [Zhang et al., 2018] that adaptive regret bounds imply dynamic regret bounds.

The study of learning in changing environments can be traced to the seminal work of Herbster and Warmuth [1998] in the context of tracking for the problem of prediction from expert advice. Their technique was later extended to tracking of experts from a small pool [Bousquet and Warmuth, 2003].

The problem of tracking a large set of experts efficiently was studied using the Fixed-Share technique in [Singer, 1998, Kozat and Singer, 2007, György et al., 2005].

The deviation from Fixed-Share to the FLH technique and the notion of adaptive regret were introduced in Hazan and Seshadhri [2007]. These techniques were subject of later study and extensions [Adamskiy et al., 2016, Zhang et al., 2019]. Daniely et al. [2015] study adaptive regret for weakly convex loss functions and introduced the term "strongly adaptive", which differentiates the weakly and strongly convex settings. They note that FLH is a strongly adaptive algorithm.

The use of an exponential look-back for prediction has roots in information theory [Willems and Krom, 1997, Shamir and Merhav, 2006]. Efficient methods for streaming, that were used in this chapter to maintaining a small set of active experts, were studied in the steaming algorithms literature [Gopalan et al., 2007].

Adaptive regret algorithms were motivated by applications involving changing environments, such as the portfolio selection problem. More recently they were applied for time series prediction [Anava et al., 2013] and the control of dynamical systems [Gradu et al., 2020].

10.7 Exercises

1. Consider an OCO setting that can be divided into k intervals, such that in each a different comparator is optimal. Let $\mathcal{A}$ be an algorithm that has an adaptive regret guarantee of $\text{AdaptiveRegret}_T(\mathcal{A}) = o(T)$. Prove that the regret of $\mathcal{A}$ vs. the best k-shifting comparator is bounded by $k \times \text{AdaptiveRegret}_T$.

2. Prove that the OGD algorithm for convex functions, with step sizes $O(\frac{1}{\sqrt{t}})$, has an adaptive regret guarantee of $O(\sqrt{T})$, and that this is tight. Prove that the lazy version of OGD, from chapter 5, behaves differently and has an adaptive regret bound of $\Omega(T)$.

3. Prove that the OGD algorithm with step size $O(\frac{1}{t})$ for strongly convex functions, has adaptive regret which is lower bounded by $\Omega(T)$.

4. Consider the problem of prediction from expert advice with α-exp-concave loss functions, where the best expert switches k times. That is, time can be divided into k segments $I_1, ..., I_k$, such that the best expert in each segment is different. Show that the regret of the Fixed Share algorithm vs. the best k-switching expert (a strategy that is allowed to change experts k times) is bounded by

$$O\left(k \log \frac{NT}{k}\right).$$

5. Spell out a choice of δ parameter for the Fixed Share algorithm that does not require knowledge of the number of iterations T in advance. Prove an analogue of Theorem 10.3 with your choice of δ.

6. Complete the proof of Theorem 10.5.

7. Prove the following facts used in the proof of Proposition 10.8:

(a) For $1 \leq i < t$, $\log p_t^i \geq \log \hat{p}_t^i + \log \frac{t-1}{t}$

(b) $\log p_t^t \geq -\log t$

8. Implement meta-algorithms FLH and FLH2 with the ONS algorithm, and apply the resulting method on the portfolio selection problem. Benchmark your results and compare to ONS and OGD.

Chapter 11

Boosting and Regret

In this chapter we consider a fundamental methodology of machine learning: *boosting*. In the statistical learning setting, roughly speaking, boosting refers to the process of taking a set of rough "rules of thumb" and combining them into a more accurate predictor.

Consider for example the problem of Optical Character Recognition (OCR) in its simplest form: given a set of bitmap images depicting hand-written postal-code digits, classify those that contain the digit "1" from those of "0".

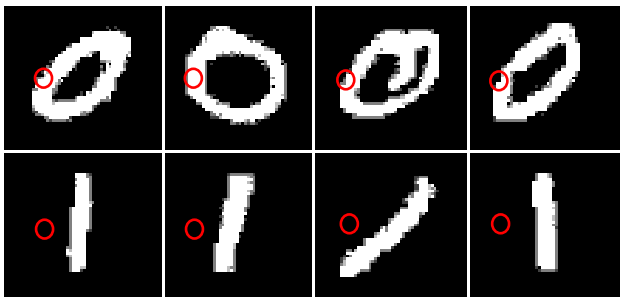

Figure 11.1: Distinguishing zero versus one from a single pixel

Seemingly, discerning the two digits seems a difficult task taking into

account the different styles of handwriting, inconsistent styles even for the same person, label errors in the training data, etc. However, an inaccurate rule of thumb is rather easy to produce: in the bottom-left area of the picture we'd expect many more dark bits for "1"s than if the image depicts a "0". This is, of course, a rather inaccurate classifier. It does not consider the alignment of the digit, thickness of the handwriting, and numerous other factors. Nevertheless, as a rule of thumb - we'd expect better-than-random performance and some correlation with the ground truth.

The inaccuracy of the crude single-bit predictor is compensated by its simplicity. It is not hard to implement a classifier based upon this rule of thumb, which is very efficient indeed. The natural and fundamental question which now arises is: can several such rules of thumb be combined into a single, accurate and efficient classifier?

In the rest of this chapter we shall formalize this question in the statistical learning theory framework. We then proceed to use the technology developed in this manuscript, namely regret minimization algorithms for online convex optimization, to answer this question in the affirmative. Our development will be somewhat non-standard: we'll describe a black-box reduction from regret-minimization to boosting. This allows any of the OCO methods previously discussed in this text to be used as the main component of a boosting algorithm.

11.1 The Problem of Boosting

Throughout this chapter we use the notation and definitions of chapter 9 on learning theory, and focus on statistical learnability rather than agnostic learnability. More formally, we assume the so called "realizability assumption", which states that for a learning problem over hypothesis class $\mathcal{H}$ there exists some $h^\star \in \mathcal{H}$ such that its generalization error is zero, or formally $\text{error}(h^\star) = 0$.

Using the notations of the previous chapter, we can define the following seemingly weaker notion than statistical learnability.

Definition 11.1 (Weak learnability)**.** *The concept class $\mathcal{H} : \mathcal{X} \mapsto \mathcal{Y}$ is said to be γ-weakly-learnable if the following holds. There exists an algorithm $\mathcal{A}$ that accepts $S_m = \{(\mathbf{x}, y)\}$ and returns an hypothesis in $\mathcal{A}(S_m) \in \mathcal{H}$ that satisfies:*
for any $\delta > 0$ there exists $m = m(\delta)$ large enough such that for any distribution $\mathcal{D}$ over pairs $(\mathbf{x}, y)$, for $y = h^\star(\mathbf{x})$, and m samples from this distribution,

it holds that with probability $1 - \delta$*,*

$$\text{error}(\mathcal{A}(S_m)) \leq \frac{1}{2} - \gamma$$

This is an apparent weakening of the definition of statistical learnability that we have described in chapter 9: the error is not required to approach zero. The standard case of statistical learning in the context of boosting is called "strong learnability". An algorithm that achieves weak learning is referred to as a weak learner, and respectively we can refer to a strong learner as an algorithm that attains statistical learning, i.e., allows for generalization error arbitrarily close to zero, for a certain concept class.

The central question of boosting can now be formalized: are weak learning and strong learning equivalent? In other words, is there an (efficient?) procedure that has access to a weak oracle for a concept class, and returns a strong learner for the class?

Miraculously, the answer is affirmative, and gives rise to one of the most effective paradigms in machine learning.

11.2 Boosting by Online Convex Optimization

In this section we describe a *reduction* from OCO to boosting. The template is similar to the one we have used in chapter 9: using one of the numerous algorithms for online convex optimization we have explored in this manuscript, as well as access to a weak learner, we create a procedure for strong learning.

11.2.1 Simplification of the setting

Our derivation focuses on simplicity rather than generality. As such, we make the following assumptions:

1. We restrict ourselves to the classical setting of binary classification. Boosting to real-valued losses is also possible, but outside our scope. Thus, we assume the loss function to be the zero-one loss, that is:

$$\ell(\hat{y}, y) = \begin{cases} 0, & y = \hat{y} \\ \\ 1, & 0/w \end{cases}$$

2. We assume that the concept class is realizable, i.e., there exists an $h^\star \in \mathcal{H}$ such that $\text{error}(h^\star) = 0$. There are results on boosting in

the agnostic learning setting, these are surveyed in the bibliographic section.

3. We denote the distribution over examples $\mathcal{X} \times \mathcal{Y} = \{(x, y)\}$, where $y = h^\star(\mathbf{x})$, as a point in $\Delta_\mathcal{X}$. That is, a point $\mathbf{p} \in \Delta_\mathcal{X}$ is a non-negative vector that integrates to one over all examples. For simplicity, we think of $\mathcal{X}, \mathcal{Y}$ as a finite, and therefore $\mathbf{p} \in \Delta_m$ belongs to the m dimensional simplex, i.e., is a discrete distribution over m elements.

4. We henceforth denote the weak learning algorithm by $\mathcal{W}$, and denote by $\mathcal{W}(\mathbf{p}, \delta)$ a call to the weak learning algorithm over distribution $\mathbf{p}$ that satisfies
$$\Pr[\operatorname*{error}_{\mathbf{p}}(\mathcal{W}(\mathbf{p}, \delta)) \geq \frac{1}{2} - \gamma] \leq \delta.$$

With these assumptions and definitions we are ready to prove the main result: a reduction from weak learning to strong learning using an online convex optimization algorithm with a sublinear regret bound. Essentially, our task would be to find a hypothesis which attains zero error on a given sample.

11.2.2 Algorithm and analysis

Pseudocode for the boosting algorithm is given in Algorithm 34. This reduction accepts as input a γ-weak learner and treats it as a black box, returning a function which we'll prove is a strong learner.

The reduction also accepts as input an online convex optimization algorithm denoted $\mathcal{A}^{OCO}$. The underlying decision set for the OCO algorithm is the m-dimensional simplex, where m is the sample size. Thus, its decisions are distributions over examples. The cost functions are linear, and assign a value of zero or one, depending on whether the current hypothesis errs on a particular example. Hence, the cost at a certain iteration is the expected error of the current hypothesis (chosen by the weak learner) over the distribution chosen by the low-regret algorithm.

It is important to note that the final hypothesis $\bar{h}$ which the algorithm outputs does not necessarily belong to $\mathcal{H}$ - the initial hypothesis class we started off with.

Theorem 11.2. *Algorithm 34 returns a hypothesis $\bar{h}$ such that with probability at least $1 - \delta$,*
$$\operatorname*{error}_{S}(\bar{h}) = 0.$$

Algorithm 34 Reduction from Boosting to OCO

Input: $\mathcal{H}, \delta$, OCO algorithm $\mathcal{A}^{OCO}$, γ-weak learning algorithm $\mathcal{W}$, sample $S_m \sim \mathcal{D}$.
Set T such that $\frac{1}{T}\text{Regret}_T(A^{OCO}) \leq \frac{\gamma}{2}$
Set distribution $\mathbf{p}_1 = \frac{1}{m}\mathbf{1} \in \Delta_m$ to be the uniform distribution.
for $t = 1, 2 \dots T$ **do**
Find hypothesis $h_t \leftarrow \mathcal{W}(\mathbf{p}_t, \frac{\delta}{2T})$
Define the loss function $f_t(\mathbf{p}) = \mathbf{r}_t^\top \mathbf{p}$, where the vector $\mathbf{r}_t \in \mathbb{R}^m$ is defined as

$$\mathbf{r}_t(i) = \begin{cases} 1, & h_t(\mathbf{x}_i) = y_i \\ 0, & o/w \end{cases}$$

Update $\mathbf{p}_{t+1} \leftarrow \mathcal{A}^{OCO}(f_1, ..., f_t)$
end for
return $\bar{h}(\mathbf{x}) = \text{sign}(\sum_{t=1}^T h_t(\mathbf{x}))$

Proof. Given $h \in \mathcal{H}$, we denote its empirical error on the sample S, weighted by the distribution $\mathbf{p} \in \delta_m$, by:

$$\underset{S,\mathbf{p}}{\text{error}}(h) = \sum_{i=1}^m \mathbf{p}(i) \cdot \mathbf{1}_{h(\mathbf{x}_i) \neq y_i}.$$

Notice that by definition of $\mathbf{r}_t$ we have $\mathbf{r}_t^\top \mathbf{p}_t = 1 - \text{error}_{S,\mathbf{p}_t}(h_t)$. Since h_t is the output of a γ-weak-learner on the distribution $\mathbf{p}_t$, we have for all $t \in [T]$,

$$\begin{aligned} \Pr[\mathbf{r}_t^\top \mathbf{p}_t \leq \frac{1}{2} + \gamma] &= \Pr[1 - \underset{S,\mathbf{p}_t}{\text{error}}(h_t) \leq \frac{1}{2} + \gamma] \\ &= \Pr[\underset{S,\mathbf{p}_t}{\text{error}}(h_t) \geq \frac{1}{2} - \gamma] \\ &\leq \frac{\delta}{2T}. \end{aligned}$$

This applies for each t separately, and by the union bound we have

$$\Pr[\frac{1}{T}\sum_{t=1}^T \mathbf{r}_t^\top \mathbf{p}_t \geq \frac{1}{2} + \gamma] \geq 1 - \delta$$

Denote by $S_\phi \subseteq S$ be the set of all missclassified examples by $\bar{h}$. Let $\mathbf{p}^*$

the uniform distribution over S_ϕ.

$$\begin{aligned}
\sum_{t=1}^{T} \mathbf{r}_t^\top \mathbf{p}^* &= \sum_{t=1}^{T} \frac{1}{|S_\phi|} \sum_{(\mathbf{x},y)\in S_\phi} \mathbf{1}_{h_t(\mathbf{x})=y} \\
&= \frac{1}{|S_\phi|} \sum_{(\mathbf{x},y)\in S_\phi} \sum_{t=1}^{T} \mathbf{1}_{h_t(\mathbf{x}_j)=y_j} \\
&\leq \frac{1}{|S_\phi|} \sum_{(\mathbf{x},y)\in S_\phi} \frac{T}{2} && \bar{h}(\mathbf{x}_j) \neq y_j \\
&= \frac{T}{2}.
\end{aligned}$$

Combining the previous two observations, we have with probability at least $1-\delta$ that

$$\begin{aligned}
\frac{1}{2}+\gamma \qquad & \leq \tfrac{1}{T}\textstyle\sum_{t=1}^{T} \mathbf{r}_t^\top \mathbf{p}_t \\
& \leq \tfrac{1}{T}\textstyle\sum_{t=1}^{T} \mathbf{r}_t^\top \mathbf{p}^* + \tfrac{1}{T}\mathrm{Regret}_T(\mathcal{A}^{OCO}) \quad \text{low regret of } \mathcal{A}^{OCO} \\
& \leq \tfrac{1}{2} + \tfrac{1}{T}\mathrm{Regret}_T(\mathcal{A}^{OCO}) \\
& \leq \tfrac{1}{2} + \tfrac{\gamma}{2}.
\end{aligned}$$

This is a contradiction. We conclude that a distribution $\mathbf{p}^*$ cannot exist, and thus all examples in S are classified correctly.

□

11.2.3 AdaBoost

A special case of the template reduction we have described is obtained when the OCO algorithm is taken to be the Multiplicative Updates method we have come to know in the manuscript.

Corollary 5.7 gives a bound of $O(\sqrt{T \log m})$ on the regret of the EG algorithm in our context. This bounds T in Algorithm 34 by $O(\frac{1}{\gamma^2} \log m)$.

Closely related is the AdaBoost algorithm, which is one of the most useful and successful algorithms in Machine Learning at large (see bibliography). Unlike the Boosting algorithm that we have analyzed, AdaBoost doesn't have to know in advance the parameter γ of the weak learners. Pseudo code for the AdaBoost algorithm is given in 35.

Algorithm 35 AdaBoost

Input: $\mathcal{H}, \delta$, γ-weak-learner $\mathcal{W}$, sample $S_m \sim \mathcal{D}$.
Set $\mathbf{p}_1 \in \Delta_m$ be the uniform distribution over S_m.
for $t = 1, 2 \ldots T$ **do**
Find hypothesis $h_t \leftarrow \mathcal{W}(\mathbf{p}_t, \frac{\delta}{T})$
Calculate $\varepsilon_t = \text{error}_{S,\mathbf{p}_t}(h_t)$, $\alpha_t = \frac{1}{2}\log(\frac{1-\varepsilon_t}{\varepsilon_t})$
Update,

$$\mathbf{p}_{t+1}(i) = \frac{\mathbf{p}_t(i)e^{-\alpha_t y_i h_t(i)}}{\sum_{j=1}^{m} \mathbf{p}_t(j)e^{-\alpha_t y_j h_t(j)}}$$

end for
return $\bar{h}(\mathbf{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}))$

11.2.4 Completing the picture

In our discussion so far we have focused only on the empirical error over a sample. To show generalization and complete the Boosting theorem, one must show that zero empirical error on a large enough sample implies ε generalization error on the underlying distribution.

Notice that the hypothesis returned by the Boosting algorithms does not belong to the original concept class. This presents a challenge for certain methods of proving generalization error bounds that are based on measure concentration over a fixed hypothesis class.

Both issues are resolved using the implication that compression implies generalization, as given in Theorem 9.9. We sketch the argument below, and the precise derivation is left as an exercise.

Roughly speaking, boosting algorithm 34 runs on m examples for $T = O(\frac{\log m}{\gamma^2})$ rounds, returns a final hypothesis $\bar{h}$ that is the majority vote of T hypothesis, and classifies correctly all m examples of the training set.

Suppose that the weak learning algorithm has sample complexity of size $k(\gamma, \delta)$: given $k = k(\gamma, \delta)$ examples, it returns a hypothesis with generalization error at most $\frac{1}{2} - \gamma$ with probability at least $1 - \delta$. Further, suppose the original training set of m examples was sampled from distribution $\mathcal{D}$.

Since $\bar{h}$ classifies correctly the entire training set, it follows that the distribution $\mathcal{D}$ has a compression scheme of size

$$Tk = O\left(\frac{k(\gamma, \frac{\delta}{T})\log m}{\gamma^2}\right).$$

Therefore, using Theorem 9.9, we have that,

$$\operatorname*{error}_{\mathcal{D}}(\bar{h}) \leq O\left(\frac{k \log^2 \frac{m}{\delta}}{\gamma^2 m}\right).$$

Now one can obtain an arbitrary small generalization error by choosing m as a function of k, δ, γ. Notice that this argument makes an assumption only about the sample complexity of the weak learning algorithm, rather than the hypothesis class $\mathcal{H}$.

11.3 Bibliographic Remarks

The theoretical question of Boosting and posed and addressed in the work of Schapire [1990], Freund [1995]. The AdaBoost algorithm was proposed in the seminal paper of Freund and Schapire [1997]. The latter paper also contains the essential ingredients for the reduction from general low-regret algorithms to boosting.

Boosting has had significant impact on theoretical and practical data analysis as described by the statistician Leo Breiman [Olshen, 2001]. For a much more comprehensive survey of Boosting theory and applications see the recent book [Schapire and Freund, 2012].

The theory for agnostic boosting is more recent, and several different definitions and settings exist, see [Kalai et al., 2008, Kalai and Servedio, 2005, Kanade and Kalai, 2009, Feldman, 2009, Ben-David et al., 2001], the most general of which is perhaps by Kanade and Kalai [2009].

A unified framework for realizable and agnostic boosting, for both the statistical and online settings, is given in [Brukhim et al., 2020].

The theory of boosting has been extended to real valued learning via the theory of gradient boosting [Friedman, 2002]. More recently it was extended to online learning [Leistner et al., 2009, Chen et al., 2012, 2014, Beygelzimer et al., 2015a,b, Agarwal et al., 2019, Jung et al., 2017, Jung and Tewari, 2018, Brukhim and Hazan, 2020].

11.4 Exercises

1. Describe a boosting algorithm based on the online gradient descent algorithm. Give a bound on its running time.

2. Download the MNIST dataset from the web. Implement weak learners to differentiate a pair of digits of your choice according to a single bit. Implement AdaBoost and apply it to your weak learners. Summarize your results and conclusions.

3. * Consider the problem of *agnostic boosting*, in which the existence of a zero-error hypothesis is not assumed.

(a) Write down an alternative definition of a weak learning algorithm for the agnostic setting.

(b) Write down a reasonable goal for a boosting algorithm.

(c) Write down an analogue of Theorem 11.2 for agnostic boosting (without proof).

4. Compute the number of samples required to achieve a generalization error of ε, using boosting algorithm 34 and a weak learning algorithm with sample complexity bound $k(\gamma, \delta)$.

Chapter 12

Online Boosting

This text considers online optimization and learning, and it is a natural question to ask whether the technique of boosting has an analogue in the online world? What is a "weak learner" in online convex optimization, and how can one strengthen it? This is the subject of this chapter, and we shall see that boosting can be extremely powerful and useful in the setting of online convex optimization.

12.1 Motivation: Learning from a Huge Set of Experts

Recall the classical problem of prediction from expert advice from the first chapter of this text. A learner iteratively makes decisions and receives loss according to an arbitrarily chosen loss function. For its decision making, the learner is assisted by a pool of experts. Classical algorithms such as the Hedge algorithm 1, guarantees a regret bound of $O(\sqrt{T \log N})$, where N is the number of experts, and this is known to be tight.

However, in many problems of interest, the class of experts is too large to efficiently manipulate. This is particularly evident in contextual learning, as formally defined below, where the experts are *policies* – functions mapping contexts to action. In such instances, even if a regret bound of $O(\sqrt{T \log N})$ is meaningful, the algorithms achieving this bound are computationally inefficient; their running time is linear in N. This linear dependence is many times unacceptable: the effective number of policies mapping contexts to actions is exponential in the number of contexts.

The boosting approach to address this computational intractability is motivated by the observation that it is often possible to design simple *rules-*

of-thumb that perform slightly better than random guesses. Analogously to the weak learning oracles from chapter 11, We propose that the learner has access to an "online weak learner" - a computationally cheap mechanism capable of guaranteeing multiplicatively approximate regret against a base hypotheses class.

In the rest of this chapter we describe efficient algorithms that when provided weak learners, compete with the convex hull of the base hypotheses class with near-optimal regret.

12.1.1 Example: boosting online binary classification

As a more precise example to the motivation we just surveyed, we formalize online boosting for binary prediction from expert advice. At iteration t, a set of experts denoted $h \in \mathcal{H}$, observe a context $\mathbf{a}_t$, and predict a binary outcome $h(\mathbf{a}_t) \in \{-1, 1\}$. The loss of each expert is taken to be the binary loss, $-h(\mathbf{a}_t) \cdot y_t$ for a true label $y_t \in \{-1, 1\}$. The Hedge algorithm from the first chapter applies to this problem, and guarantees a regret of $O(\sqrt{T \log |\mathcal{H}|})$ for a finite $\mathcal{H}$. However, the case in which $\mathcal{H}$ is extremely large, maintaining the weights is prohibitive computationally.

A weak online learner $\mathcal{W}$ in this setting is an algorithm which is guaranteed to attain at most a factor γ loss from the best expert in class, for some $\gamma \in [0, 1]$, up to an additive sublinear regret term. Formally, for any sequence of contexts and labels $\{\mathbf{a}_t, y_t\}$,

$$\sum_{t=1}^{T} y_t \cdot \mathcal{W}(\mathbf{a}_t) \leq \gamma \cdot \min_{h \in \mathcal{H}} \sum_{t=1}^{T} y_t \cdot h(\mathbf{a}_t) + \text{Regret}_T(\mathcal{W}).$$

The online boosting question can now be phrased as follows: given access to a weak online learning algorithm $\mathcal{W}$, can we design an efficient online algorithm $\mathcal{A}$ that guarantees vanishing regret over $\mathcal{H}$? More formally, let

$$\text{Regret}_T(\mathcal{A}) = \sum_{t=1}^{T} y_t \cdot \mathcal{W}(\mathbf{a}_t) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} y_t \cdot h(\mathbf{a}_t).$$

Can we design an algorithm $\mathcal{A}$ that has $\frac{\text{Regret}_T(\mathcal{A})}{T} \mapsto 0$, without explicit access to $\mathcal{H}$? As we will see, the answer to this question is affirmative in a strong sense: boosting does have an online analogue which is a powerful technique in online learning. In the next section we describe a more powerful notion of boosting that applies to the full generality of online convex optimization. This in turn implies an affirmative answer to this question for online binary classification.

12.1.2 Example: personalized article placement

In the problem of matching articles to visitors of a web-page on the Internet, a number of articles are available to be placed in a given web-page for a particular visitor. The goal of the decision maker, in this case the article placer, is to find the most relevant article that will maximize the probability of a visitor click.

It is usually the case that context is available, in the form of user profile, preferences surfing history and so forth. This context is invaluable in terms of placing the most relevant article. Thus, the decision of the article-placer is to choose from a *policy*: a mapping from context to article.

The space of policies is significantly larger than the space of articles and space of contexts: its size is the power of articles to the cardinality of contexts. This motivates the use of online learning algorithms whose computational complexity is independent of the number of experts.

The natural formulation of this problem is not binary prediction, but rather multi-class prediction. Formulating this problem in the language of online convex optimization is left as an exercise.

12.2 The Contextual Learning Model

Boosting in the context of online convex optimization is most useful for the contextual learning problem which we now describe.

Let us consider the familiar OCO setting over a general convex decision set $\mathcal{K} \subseteq \mathbb{R}^d$, and adversarially chosen convex loss functions $f_1, ..., f_t : \mathcal{K} \mapsto \mathbb{R}$. Boosting is particularly important in settings that we have a very large number of possible experts that makes running one of the algorithms we have considered thus far infeasible. Concretely, suppose we have access to a hypothesis class $\mathcal{H} \subseteq \{\mathbf{a}\} \mapsto \mathcal{K}$, that given a sequence of contexts $\mathbf{a}_1, ..., \mathbf{a}_t$, produces a new point $h(\mathbf{a}_{t+1}) \in \mathcal{K}$.

We have studied numerous methods capable of minimizing regret for this setting in this text, all assumed that we have access to the set $\mathcal{H}$, and depend on its diameter in some way.

To avoid this dependence, we consider an alternative access model to $\mathcal{H}$. A weak learner for the OCO setting is defined as follows.

Definition 12.1. *An online learning algorithm $\mathcal{W}$ is a γ-**weak OCO learner (WOCL)** for $\mathcal{H}$ and $\gamma \in (0, 1)$, if for any sequence of contexts $\{\mathbf{a}_t\}$ and* **linear** *loss functions $f_1, ..., f_T$, for which $\max_{\mathbf{x}\in\mathcal{K}} f_t(\mathbf{x}) - \min_{\mathbf{y}\in\mathcal{K}} f_t(\mathbf{y}) \leq 1$*

[15], *we have*

$$\sum_{t=1}^{T} f_t(\mathcal{W}(\mathbf{a}_t)) \leq \gamma \cdot \min_{h \in \mathcal{H}} \sum_{t=1}^{T} f_t(h(\mathbf{a}_t)) + (1-\gamma) \sum_{t=1}^{T} f_t(\bar{\mathbf{x}}) + \text{Regret}_T(\mathcal{W}), \tag{12.1}$$

where $\bar{\mathbf{x}} = \int_{\mathbf{x} \in \mathcal{K}} \mathbf{x}$ *is the center of mass of* $\mathcal{K}$.

This definition differs in two aspects from the types of regret minimization guarantees we have seen thus far. For one, the algorithm competes with a γ-multiple of the best comparator in hindsight, and is "weak" in this precise manner.

Secondly, a multiplicative guarantee is not invariant for a constant shift. This is the reason for the existence of an additional component, $\sum_t f_t(\bar{\mathbf{x}})$, in the regret bound. This can be thought of as the cost of a random, or naive predictor. A weak learner must, at the very least, perform better than this naive and non-anticipating predictor!

It is convenient to henceforth assume that the loss functions are shifted such that $f_t(\bar{\mathbf{x}}) = 0$. Under this assumption, we can rephrase γ-WOCL as

$$\sum_{t=1}^{T} f_t(\mathcal{W}(\mathbf{a}_t)) \leq \gamma \cdot \min_{h \in \mathcal{H}} \sum_{t=1}^{T} f_t(h(\mathbf{a}_t)) + \text{Regret}_T(\mathcal{W}). \tag{12.2}$$

12.3 The Extension Operator

The main difficulty is coping with the approximate guarantee that the WOCL provides. Therefore the algorithm we describe henceforth scales the predictions returned by the weak learner by a factor of $\frac{1}{\gamma}$. This means that the scaled decisions do not belong to the original decision set anymore, and need to be projected back.

Here lies the main challenge. First, we assume that the loss functions $f \in \mathcal{F}$ are defined over all of $\mathbb{R}^d$ to enable valid decisions outside of $\mathcal{K}$. Next, we need to be able to project onto $\mathcal{K}$ without increasing the cost. It can be seen that some natural families of functions, i.e., linear functions, do not admit any such projection. To remedy this situation, we define the extension operator of a function over a convex domain $\mathcal{K}$ as follows.

First, denote the Euclidean distance function to a set $\mathcal{K}$ as (see also section 13.2),

$$\mathbf{Dist}(\cdot, \mathcal{K}) \ , \ \mathbf{Dist}(\mathbf{x}, \mathcal{K}) = \min_{\mathbf{y} \in \mathcal{K}} \|\mathbf{y} - \mathbf{x}\|.$$

Definition 12.2 ($(\mathcal{K}, \kappa, \delta)$-extension)**.** *The extension operator over $\mathcal{K} \subseteq \mathbb{R}^d$ is defined as:*

$$X_{\mathcal{K},\kappa,\delta}[f] : \mathbb{R}^d \mapsto \mathbb{R} \quad , \quad X[f] = S_\delta[f + \kappa \cdot \mathbf{Dist}(\cdot, \mathcal{K})],$$

where the smoothing operator S_δ was defined as per Lemma 2.8.

The important take-away from these operators is the following lemma, whose importance is crucial in the OCO boosting algorithm 36, as it projects infeasible points that are obtained from the weak learners to the feasible domain.

Lemma 12.3. *The $(\mathcal{K}, \kappa, \delta)$-extension of a function $\hat{f} = X[f]$ satisfies the following:*

1. *For every point $\mathbf{x} \in \mathcal{K}$, we have $\|\hat{f}(\mathbf{x}) - f(\mathbf{x})\|_2 \leq \delta G$.*

2. *The projection of a point, whose gradient is bounded by G, onto $\mathcal{K}$ improves the $(\mathcal{K}, \kappa, \delta)$-extension function, for $\kappa = G$, value up to a small term,*

$$\hat{f}\left(\prod_{\mathcal{K}}(\mathbf{x})\right) \leq \hat{f}(\mathbf{x}) + \delta G.$$

Proof. 1. Since $\mathbf{Dist}(\mathbf{x}, \mathcal{K}) = 0$ for all $x \in \mathcal{K}$, this follows immediately from Lemma 2.8.

2. Denote $\mathbf{x}_\pi = \prod_{\mathcal{K}}(\mathbf{x})$ for brevity. Then

$$\begin{aligned}
&\hat{f}(\mathbf{x}_\pi) - \hat{f}(\mathbf{x}) \\
&\leq f(\mathbf{x}_\pi) - f(\mathbf{x}) - \kappa \mathbf{Dist}(\mathbf{x}, \mathcal{K}) + \delta G && \text{part 1} \\
&\leq f(\mathbf{x}_\pi) - f(\mathbf{x}) - \kappa \|\mathbf{x} - \mathbf{x}_\pi\| + \delta G \\
&= \nabla f(\mathbf{x})(\mathbf{x} - \mathbf{x}_\pi) - \kappa \|\mathbf{x} - \mathbf{x}_\pi\| + \delta G \\
&\leq \|\nabla f(\mathbf{x})\| \|\mathbf{x} - \mathbf{x}_\pi\| - \kappa \|\mathbf{x} - \mathbf{x}_\pi\| + \delta G && \text{Cauchy-Schwartz} \\
&\leq G \|\mathbf{x} - \mathbf{x}_\pi\| - \kappa \|\mathbf{x} - \mathbf{x}_\pi\| + \delta G \\
&\leq \delta G && \text{choice of } \kappa.
\end{aligned}$$

□

12.4 The Online Boosting Method

The online boosting algorithm we describe in this section is closely related to the online Frank-Wolfe algorithm from chapter 7. Not only does it deliver in boosting WOCL to strong learning, but it gives an even stronger guarantee: low regret over the convex hull of the hypothesis class.

Algorithm 36 efficiently converts a weak online learning algorithm into an OCO algorithm with vanishing regret in a black-box manner. The idea is to apply the weak learning algorithm on linear functions that are gradients of the loss. The algorithm then recursively applies another weak learner on the gradients of the residual loss, and so forth.

Algorithm 36 Boosting for Online Convex Optimization

Input: N copies of the γ-WOCL $\mathcal{W}^1, \mathcal{W}^2, \ldots, \mathcal{W}^N$, parameters $\eta_1, ..., \eta_T, \delta, \kappa = G$.
for $t = 1$ **to** T **do**
  Receive context $\mathbf{a}_t$, choose $\mathbf{x}_t^0 = \mathbf{0}$ arbitrarily.
  for $i = 1$ **to** N **do**
    Define $\mathbf{x}_t^i = (1 - \eta_i)\mathbf{x}_t^{i-1} + \eta_i \frac{1}{\gamma} \mathcal{W}^i(\mathbf{a}_t)$.
  end for
  Predict $\mathbf{x}_t = \prod_{\mathcal{K}}[\mathbf{x}_t^N]$, suffer loss $f_t(\mathbf{x}_t)$.
  Obtain loss function f_t, create $\hat{f}_t = X_{\mathcal{K},\kappa,\delta}[f_t]$.
  for $i = 1$ **to** N **do**
    Define and pass to $\mathcal{W}^i$ the linear loss function f_t^i,

$$f_t^i(\mathbf{x}) = \nabla \hat{f}_t(\mathbf{x}_t^{i-1}) \cdot \mathbf{x}.$$

  end for
end for

However, the Frank-Wolfe method is not applied directly to the loss functions, but rather to a proxy loss which defined using the extension operation in 12.2. Importantly, algorithm 36 has a running time that is independent of $|\mathcal{H}|$.

Notice that if $\gamma = 1$, the algorithm stills gives a significant advantage as compared to the weak learner: the regret guarantee is vs. the convex hull of $\mathcal{H}$, as compared to the best single hypothesis.

Theorem 12.4 (Main)**.** *The predictions* $\mathbf{x}_t$ *generated by Algorithm 36 with*

$\delta = \sqrt{\frac{D^2}{\gamma N}}, \eta_i = \min\{\frac{2}{i}, 1\}$ *satisfy*

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t) \; - \min_{h^\star \in \mathbf{CH}(\mathcal{H})} \sum_{t=1}^{T} f_t(h^\star(\mathbf{a}_t)) \leq \frac{5dGDT}{\gamma\sqrt{N}} + \frac{2GD}{\gamma}\text{Regret}_T(\mathcal{W}).$$

Remark 1: It is possible to obtain tighter bounds by a factor of the dimension, and other constant terms, using a more sophisticated smoothing operator. References for these tighter results are given in the bibliographic section at the end of this chapter.

Remark 2: The regret bound of Theorem 12.4 is nearly as good as we could hope for. The first term approaches zero as the number of weak learners N grows. The second term is sublinear as the regret of the weak learner. It is scaled by a factor of $\frac{1}{\gamma}$, which we can expect due to the approximate guarantee of the weak learner.

Before proving the theorem, let us define some notations we use. The algorithm defines the extension of the loss functions as

$$\hat{f}_t = X[f_t] = S_\delta[f_t + G \cdot \mathbf{Dist}(\mathbf{x}, \mathcal{K})].$$

We apply the setting of $\kappa = G$, as required by Lemma 12.3, and by Lemma 2.8, $\hat{f}_t$ is $\frac{dG}{\delta}$-smooth. Also, denote by $\mathbf{CH}(\mathcal{H}) = \{\sum_{h\in\mathcal{H}} \mathbf{p}_h h | \mathbf{p} \in \Delta_\mathcal{H}\}$ the convex hull of the set $\mathcal{H}$, and let

$$h^\star = \operatorname*{arg\,min}_{h^\star \in \mathbf{CH}(\mathcal{H})} \sum_{t=1}^{T} f_t(h^\star(\mathbf{a}_t))$$

to be the best hypothesis in the convex hull of $\mathcal{H}$ in hindsight, i.e., the best convex combination of hypothesis from $\mathcal{H}$. Notice that since the loss functions are generally convex and non-linear, this convex combination is not necessarily a singleton. We define $\mathbf{x}_t^\star = h^\star(\mathbf{a}_t)$ as the decisions of this hypothesis.

The main crux of the proof is given by the following lemma.

Lemma 12.5. *For smoothed loss functions $\{\hat{f}_t\}$ that are β-smooth and $\hat{G}$ Lipschitz, it holds that*

$$\sum_{t=1}^{T} \hat{f}_t(\mathbf{x}_t^N) \; - \sum_{t=1}^{T} \hat{f}_t(\mathbf{x}_t^\star) \leq \frac{2\beta D^2 T}{\gamma^2 N} + \frac{\hat{G}D}{\gamma}\text{Regret}_T(\mathcal{W}).$$

Proof. Define for all $i = 0, 1, 2, \ldots, N$,

$$\Delta_i = \sum_{t=1}^{T} \left(\hat{f}_t(\mathbf{x}_t^i) - \hat{f}_t(\mathbf{x}_t^\star) \right).$$

Recall that $\hat{f}_t$ is β smooth by our assumption. Therefore:

$$\begin{aligned}
\Delta_i &= \sum_{t=1}^{T} \left[\hat{f}_t(\mathbf{x}_t^{i-1} + \eta_i(\frac{1}{\gamma}\mathcal{W}^i(\mathbf{a}_t) - \mathbf{x}_t^{i-1})) - \hat{f}_t(\mathbf{x}_t^\star) \right] \\
&\leq \sum_{t=1}^{T} \Big[\hat{f}_t(\mathbf{x}_t^{i-1}) - \hat{f}_t(\mathbf{x}_t^\star) + \eta_i \nabla \hat{f}_t(\mathbf{x}_t^{i-1}) \cdot (\frac{1}{\gamma}\mathcal{W}^i(\mathbf{a}_t) - \mathbf{x}_t^{i-1}) \\
&\quad + \frac{\eta_i^2 \beta}{2} \|\frac{1}{\gamma}\mathcal{W}^i(\mathbf{a}_t) - \mathbf{x}_t^{i-1}\|^2 \Big].
\end{aligned}$$

By using the definition and linearity of f_t^i, we have

$$\begin{aligned}
\Delta_i &\leq \sum_{t=1}^{T} \left[\hat{f}_t(\mathbf{x}_t^{i-1}) - \hat{f}_t(\mathbf{x}_t^\star) + \eta_i(f_t^i(\frac{1}{\gamma}\mathcal{W}^i(\mathbf{a}_t)) - f_t^i(\mathbf{x}_t^{i-1})) + \frac{\eta_i^2 \beta D^2}{2\gamma^2} \right] \\
&= \Delta_{i-1} + \sum_{t=1}^{T} \eta_i(\frac{1}{\gamma} f_t^i(\mathcal{W}^i(\mathbf{a}_t)) - f_t^i(\mathbf{x}_t^{i-1})) + \sum_{t=1}^{T} \frac{\eta_i^2 \beta D^2}{2\gamma^2}.
\end{aligned}$$

Now, note the following equivalent restatement of the WOCL guarantee, which again utilizes linearity of f_t^i to conclude: linear loss on a convex combination of a set is equal to the same convex combination of the linear loss applied to individual elements.

$$\begin{aligned}
\frac{1}{\gamma} \sum_{t=1}^{T} f_t^i(\mathcal{W}^i(\mathbf{a}_t)) &\leq \min_{h^\star \in \mathcal{H}} \sum_{t=1}^{T} f_t^i(h^\star(\mathbf{a}_t)) + \frac{\hat{G} D \mathrm{Regret}_T(\mathcal{W})}{\gamma} \\
&= \min_{h^\star \in \mathbf{CH}(\mathcal{H})} \sum_{t=1}^{T} f_t^i(h^\star(\mathbf{a}_t)) + \frac{\hat{G} D \mathrm{Regret}_T(\mathcal{W})}{\gamma}.
\end{aligned}$$

Using the above and that $h^\star \in \mathbf{CH}(\mathcal{H})$, we have

$$\begin{aligned}
&\Delta_i \\
&\leq \Delta_{i-1} + \sum_{t=1}^{T} [\eta_i \nabla \hat{f}_t(\mathbf{x}_t^{i-1}) \cdot (\mathbf{x}_t^\star - \mathbf{x}_t^{i-1}) + \frac{\eta_i^2 \beta D^2}{2\gamma^2}] + \eta_i \frac{\hat{G} D}{\gamma} \mathrm{Regret}_T(\mathcal{W}) \\
&\leq \Delta_{i-1}(1 - \eta_i) + \frac{\eta_i^2 \beta D^2 T}{2\gamma^2} + \eta_i R_T.
\end{aligned}$$

where the last inequality uses the convexity of $\hat{f}_t$ and we denote $R_T = \frac{\hat{G}D}{\gamma}\text{Regret}_T(\mathcal{W})$. We thus have the recurrence

$$\Delta_i \leq \Delta_{i-1}(1-\eta_i) + \eta_i^2 \frac{\beta D^2 T}{2\gamma^2} + \eta_i R_T.$$

Denoting $\hat{\Delta}_i = \Delta_i - R_T$, we are left with

$$\hat{\Delta}_i \leq \hat{\Delta}_{i-1}(1-\eta_i) + \eta_i^2 \frac{\beta D^2 T}{2\gamma^2}.$$

This is a recursive relation that can be simplified by applying Lemma 7.2 from chapter 7. We obtain that $\hat{\Delta}_N \leq \frac{2\beta D^2 T}{\gamma^2 N}$. □

We are ready to prove the main guarantee of Algorithm 36.

Proof of Theorem 12.4. Using both parts of Lemma 12.3 in succession, we have

$$\begin{aligned}\sum_{t=1}^{T} f_t(\mathbf{x}_t) \ - \sum_{t=1}^{T} f_t(\mathbf{x}_t^\star) &\leq \sum_{t=1}^{T} \hat{f}_t(\mathbf{x}_t) \ - \sum_{t=1}^{T} \hat{f}_t(\mathbf{x}_t^\star) + 2\delta GT \\ &\leq \sum_{t=1}^{T} \hat{f}_t(\mathbf{x}_t^N) \ - \sum_{t=1}^{T} \hat{f}_t(\mathbf{x}_t^\star) + 3\delta GT.\end{aligned}$$

Next, recall by Lemma 2.8, that $\hat{f}_t$ is $\frac{dG}{\delta}$-smooth. By applying Lemma 12.5, and optimizing δ, we have

$$\begin{aligned}\sum_{t=1}^{T} f_t(\mathbf{x}_t) \ - \sum_{t=1}^{T} f_t(\mathbf{x}_t^\star) &\leq 3\delta GT + \frac{2dGD^2T}{\delta\gamma^2 N} + \frac{\hat{G}D}{\gamma}\text{Regret}_T(\mathcal{W}) \\ &= \frac{5\sqrt{d}GDT}{\gamma\sqrt{N}} + \frac{\hat{G}D}{\gamma}\text{Regret}_T(\mathcal{W}) \\ &\leq \frac{5dGDT}{\gamma\sqrt{N}} + \frac{\hat{G}D}{\gamma}\text{Regret}_T(\mathcal{W}),\end{aligned}$$

where the last inequality is only to obtain a nicer expression.

It remains to bound $\hat{G}$, and we claim that $\hat{G} \leq 2G$. To see this, notice that the function $\mathbf{Dist}(\mathbf{x}, \mathcal{K})$ is 1-Lipschitz, since

$$\begin{aligned}&\mathbf{Dist}(\mathbf{x}, \mathcal{K}) - \mathbf{Dist}(\mathbf{y}, \mathcal{K}) \\ &= \|\mathbf{x} - \Pi_{\mathcal{K}}(\mathbf{x})\| - \|\mathbf{y} - \Pi_{\mathcal{K}}(\mathbf{y})\| \\ &\leq \|\mathbf{x} - \Pi_{\mathcal{K}}(\mathbf{y})\| - \|\mathbf{y} - \Pi_{\mathcal{K}}(\mathbf{y})\| && \Pi_{\mathcal{K}}(\mathbf{y}) \in \mathcal{K} \\ &\leq \|\mathbf{x} - \mathbf{y}\|. && \Delta\text{-inequality}\end{aligned}$$

Thus, by the definition of the extension operator and the functions f_t^i, we have that $\|\nabla f_t^i(\mathbf{x}_t^i)\| = \|\nabla \hat{f}_t(\mathbf{x}_t^i)\| \leq 2G$. □

12.5 Bibliographic Remarks

The theory of boosting, which we have surveyed in chapter 11, originally applied to binary classification problems. Boosting for real-valued regression was studied in the theory of gradient boosting by Friedman [2002].

Online boosting, for both the classification and regression settings was studied much later [Leistner et al., 2009, Chen et al., 2012, 2014, Beygelzimer et al., 2015a,b, Agarwal et al., 2019, Jung et al., 2017, Jung and Tewari, 2018, Brukhim and Hazan, 2020]. The relationship to the Frank-Wolfe method was explicit in these works, and also studied in [Freund et al., 2017, Wang et al., 2015]. A framework which encapsulates both agnostic and realizable boosting, for both offline and online settings, is given in [Brukhim et al., 2020].

Boosting for the full online convex optimization setting, with a multiplicative approximation and general convex decision set, was obtained in [Hazan and Singh, 2021]. The latter also give tighter bounds by a factor of the dimension than those presented in this text using a more sophisticated smoothing technique known as the Moreau-Yoshida regularization [Beck, 2017].

The contextual experts and bandits problems have been proposed by Langford and Zhang [2008] as a decision making framework with large number of policies. In the online setting, several works study the problem with emphasis on efficient algorithms given access to an optimization oracle [Rakhlin and Sridharan, 2016, Syrgkanis et al., 2016a,b, Rakhlin and Sridharan, 2016]. For surveys on contextual bandit algorithms and applications of this model see [Zhou, 2015, Bouneffouf and Rish, 2019].

12.6 Exercises

1. Derive, from Algorithm 36, an algorithm for online binary classification. Spell out its regret guarantee.

2. Formulate the online personalized article placement problem in the language of online convex optimization (what is the decision set?). Define a weak learner in this context, and the final boosting guarantee from Theorem 12.4.

3. * Define the Moreau-Yoshida regularization (or smoothing operator) as:

$$M_\delta[f](\mathbf{x}) = \inf_{\mathbf{y}\in\mathbb{R}^d} \left\{ f(\mathbf{y}) + \frac{1}{2\delta}\|\mathbf{x}-\mathbf{y}\|^2 \right\}.$$

Prove that given any G-Lipschitz f, the smoothed function $\hat{f}_\delta = M_\delta[f]$ satisfies:

(a) $\hat{f}_\delta$ is $\frac{1}{\delta}$-smooth, and G-Lispchitz.

(b) $\left|\hat{f}_\delta(\mathbf{x}) - f(\mathbf{x})\right| \leq \frac{\delta G^2}{2}$ for all $\mathbf{x}\in\mathbb{R}^d$.

4. * Use the Moreau-Yoshida regularization to improve the bounds of Theorem 12.4 by a factor of the dimension.

Chapter 13

Blackwell Approachability and Online Convex Optimization

The history of adversarial prediction started with the seminal works of mathematicians David Blackwell and James Hannan. In most of the text thus far, we have presented the viewpoint of sequential prediction and loss minimization, taken by Hannan. This was especially true in chapter 5, as the FPL algorithm dates back to his work. In this chapter we turn to a dual view of regret minimization, called "Blackwell approachability". Approachability theory originated in the work of Blackwell, and was discovered simultaneously to that of Hannan. A short historical account is surveyed in the bibliographic materials at the end of this chapter.

For decades the relationship between regret minimization in general convex games and Blackwell approachability was not fully understood. The common thought was, in fact, that Blackwell approachability is a stronger notion. In this chapter we show that approachability and online convex optimization are equivalent in a strong sense: an algorithm for one task implies an algorithm for the other with no loss of computational efficiency.

As a side benefit to this equivalence, we deduce a proof of Blackwell's approachability theorem using the existence of online convex optimization algorithms. This proof applies to a more general version of approachabilty, over general vector games, and comes with rates of convergence that are borrowed from the OCO algorithms we have already studied.

While previous chapters had a practical motivation and introduced methods for online learning, this chapter is purely theoretical, and devoted to give

an alternate viewpoint of online convex optimization from a game theoretic perspective.

13.1 Vector-Valued Games and Approachability

Von Neumann's minimax theorem, that we have studied in chapter 8, establishes a central result in the theory of two-player zero-sum games by providing a prescription to both players. This prescription is in the form of a pair of optimal mixed strategies: each strategy attains the optimal worst-case value of the game without knowledge of the opponent's strategy. However, the theorem fundamentally requires that both players have a utility function that can be expressed as a *scalar*.

In 1956, in response to von Neumann's result, David Blackwell posed an intriguing question: what guarantee can we hope to achieve when playing a two-player game with a *vector-valued payoff*?

A vector-valued game is defined similarly to zero-sum games as we have defined in Definition 8.2, with reward/loss vectors replacing the scalar rewards/losses.

Definition 13.1. *A two-player vector game is given by a set of $n \times m$ vectors $\{\mathbf{u}(i,j) \in \mathbb{R}^d\}$. The reward vector for the row player playing strategy $i \in [n]$, and column player playing strategy $j \in [m]$, is given by the vector $\mathbf{u}(i,j) \in \mathbb{R}^d$.*

Similar to scalar games, we can define mixed strategies as distributions over pure strategies, and denote the expected reward vector for playing mixed strategies by

$$\forall \mathbf{x} \in \Delta_n, \mathbf{y} \in \Delta_m \ . \ \mathbf{u}(\mathbf{x},\mathbf{y}) = \mathop{\mathbf{E}}_{i \sim \mathbf{x}, j \sim \mathbf{y}} \left[\mathbf{u}(i,j)\right].$$

We henceforth consider more general vector games than originally considered in the literature. The additional generality allows for uncountably many strategies for both players, and allows the strategies to originate from bounded convex and closed sets in Euclidean space.

Definition 13.2. *A generalized two-player vector game is given by a set of vectors $\{\mathbf{u} \in \mathbb{R}^d\}$, and two bounded convex and closed decision sets $\mathcal{K}_1, \mathcal{K}_2$. The reward vector for the row player playing strategy $\mathbf{x} \in \mathcal{K}_1$, and column player playing strategy $\mathbf{y} \in \mathcal{K}_2$, is given by the vector $\mathbf{u}(\mathbf{x},\mathbf{y}) \in \mathbb{R}^d$.*

The goal of a zero-sum game is clear: to guarantee a certain loss/reward. What should be the vector game generalization? Blackwell proposed to ask "can we guarantee that our vector payoff lies in some closed convex set S?"

It is left as an exercise at the end of this chapter to show that an immediate analogue of Von Neumann's theorem does not exist: there is no single mixed strategy that ensures the vector payoff lies in a given set. However, this does not rule out an asymptotic notion, if we allow the game to repeat indefinitely, and ask whether there exists a strategy to ensure that the *average* reward vector lies in a certain set, or at least approaches it in terms of Euclidean distance. This is exactly the solution concept that Blackwell proposed as defined formally below.

Using the notation we have used throughout this text, we denote the (Euclidean) distance to a bounded, closed and convex set S as

$$\mathbf{Dist}(\mathbf{w}, S) = \min_{\mathbf{x} \in S} \|\mathbf{w} - \mathbf{x}\|.$$

Definition 13.3. *Given a generalized vector game $\mathcal{K}_1, \mathcal{K}_2, \{\mathbf{u}(\cdot,\cdot)\}$, we say that a set $S \subseteq \mathbb{R}^d$ is* **approachable** *if there exists some algorithm $\mathcal{A}$, called an* **approachability algorithm**, *which iteratively selects points $\mathcal{K}_1 \ni \mathbf{x}_t \leftarrow \mathcal{A}(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1})$, such that, for any sequence $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T \in \mathcal{K}_2$, we have*

$$\mathbf{Dist}\Big(\tfrac{1}{T}\textstyle\sum_{t=1}^{T} \mathbf{u}(\mathbf{x}_t, \mathbf{y}_t), S\Big) \to 0 \quad \textit{as} \quad T \to \infty.$$

Under this notion, we can now allow the player to implement an adaptive strategy for a repeated version of the game, and we require that the average reward vector comes arbitrarily close to S. Blackwell's theorem characterizes which sets in Euclidean space are approachable. We give it below in generalized form,

Theorem 13.4 (Blackwell's Approachability Theorem)**.** *For any vector game $\mathcal{K}_1, \mathcal{K}_2, \{\mathbf{u}(\cdot,\cdot)\}$, the closed, bounded and convex set $S \subseteq \mathbb{R}^d$ is approachable if and only if the following condition holds:*

$$\forall \mathbf{y} \in \mathcal{K}_2 \ , \ \exists \mathbf{x} \in \mathcal{K}_1 \ , s.t. \ \mathbf{u}(\mathbf{x}, \mathbf{y}) \in S.$$

The approachability condition spelled out in the equation above is both necessary and sufficient. The necessity of this condition is left as an exercise, and the more interesting implication is that any set that satisfies this condition is, in fact, approachable. Our reductions henceforth give an explicit proof of Blackwell's theorem, and we leave it as an exercise to draw the explicit conclusion of this theorem from the first efficient reduction.

The relationship between Blackwell approachability in vector games and OCO may not be evident at this point. However, we proceed to show that the two notions are in fact algorithmically equivalent.

In the next section we show that any algorithm for OCO can be efficiently converted to an approachability algorithm for vector games. Following this, we show the other direction as well: an approachability algorithm for vector games gives an OCO algorithm with no loss of efficiency!

13.2 From Online Convex Optimization to Approachability

In this section we give an efficient reduction from OCO to approachability. Namely, assume that we have an OCO algorithm denoted $\mathcal{A}$, that attains sublinear regret. Our goal is to design a Blackwell approachability algorithm for a given vector game and closed, bounded convex set S. Thus, the reduction in this section shows that OCO is a stronger notion than approachabiliy. This direction is perhaps the more surprising one, and was discovered more recently, see bibliographic section for an historical account of this development.

Since we are looking to approach a given set, it is natural to consider minimizing the distance of our reward vector to the set. Recall we denote the (Euclidean) distance to a set as $\mathbf{Dist}(\mathbf{w}, S) = \min_{\mathbf{x}\in S} \|\mathbf{w} - \mathbf{x}\|$. The support function of closed convex set S is given by

$$h_S(\mathbf{w}) = \max_{\mathbf{x}\in S}\{\mathbf{w}^\top \mathbf{x}\}.$$

Notice that this function is convex, since it is a maximum over linear functions.

Lemma 13.5. *The distance to a set can be written as*

$$\mathbf{Dist}(\mathbf{u}, S) = \max_{\|\mathbf{w}\|\le 1} \left\{\mathbf{w}^\top \mathbf{u} - h_S(\mathbf{w})\right\}. \tag{13.1}$$

Proof. Using the definition of the support function,

$$\begin{aligned}
&\max_{\|\mathbf{w}\|\le 1} \left\{\mathbf{w}^\top \mathbf{u} - h_S(\mathbf{w})\right\} \\
&= \max_{\|\mathbf{w}\|\le 1} \left\{\mathbf{w}^\top \mathbf{u} - \max_{\mathbf{x}\in S} \mathbf{w}^\top \mathbf{x}\right\} \\
&= \max_{\|\mathbf{w}\|\le 1} \min_{\mathbf{x}\in S} \left\{\mathbf{w}^\top \mathbf{u} - \mathbf{w}^\top \mathbf{x}\right\} && \text{negation} \\
&= \min_{\mathbf{x}\in S} \max_{\|\mathbf{w}\|\le 1} \left\{\mathbf{w}^\top \mathbf{u} - \mathbf{w}^\top \mathbf{x}\right\} && \text{minimax theorem} \\
&= \min_{\mathbf{x}\in S} \|\mathbf{x} - \mathbf{u}\| \\
&= \mathbf{Dist}(\mathbf{u}, S).
\end{aligned}$$

□

Blackwell's theorem characterizes approachable sets: it is necessary and sufficient to be able to find a best response $\mathbf{x} \in \mathcal{K}_1$, to any $\mathbf{y} \in \mathcal{K}_2$, such that $\mathbf{u}(\mathbf{x}, \mathbf{y}) \in S$. To proceed with the reduction, we need an equivalent condition stated formally as follows.

Lemma 13.6. *For a generalized vector game $\mathcal{K}_1, \mathcal{K}_2, \{\mathbf{u}\}$, the following to conditions are equivalent:*

1. *There exists a feasible best response,*

$$\forall \mathbf{y} \in \mathcal{K}_2 \ , \ \exists \mathbf{x} \in \mathcal{K}_1 \ , s.t. \ \mathbf{u}(\mathbf{x}, \mathbf{y}) \in S.$$

2. *For all $\mathbf{w} \in \mathbb{R}^d, \|\mathbf{w}\| \leq 1$, there exists $\mathbf{x} \in \mathcal{K}_1$ such that*

$$\forall \mathbf{y} \in \mathcal{K}_2 \quad , \quad \mathbf{w}^\top \mathbf{u}(\mathbf{x}, \mathbf{y}) - h_S(\mathbf{w}) \leq 0.$$

Proof. Consider the scalar zero sum game

$$\min_{\mathbf{x}} \max_{\mathbf{y}} \mathbf{Dist}(\mathbf{u}(\mathbf{x}, \mathbf{y}), S) = \lambda.$$

Blackwell's theorem asserts that $\lambda = 0$ if and only if S is approachable. Using Sion's generalization to the Von Neumann minimax theorem from chapter 8,

$$\begin{aligned} \lambda &= \min_{\mathbf{x}} \max_{\mathbf{y}} \mathbf{Dist}(\mathbf{u}(\mathbf{x}, \mathbf{y}), S) \\ &= \min_{\mathbf{x}} \max_{\mathbf{y}} \max_{\|\mathbf{w}\| \leq 1} \left\{ \mathbf{w}^\top \mathbf{u}(\mathbf{x}, \mathbf{y}) - h_S(\mathbf{w}) \right\} \quad \text{Lemma 13.5} \\ &= \max_{\|\mathbf{w}\| \leq 1} \min_{\mathbf{x}} \max_{\mathbf{y}} \left\{ \mathbf{w}^\top \mathbf{u}(\mathbf{x}, \mathbf{y}) - h_S(\mathbf{w}) \right\} \quad \text{minimax} \end{aligned}$$

Thus, the second statement of the lemma is satisfied if and only if $\lambda = 0$.

□

As mentioned previously, the necessity of Blackwell's condition is left as an exercise. To prove sufficiency, we assume that form (2) of Blackwell's condition as in Lemma 13.6 is satisfied. Formally, we henceforth assume that the vector game and set S are equipped with a best response oracle $\mathcal{O}$, such that

$$\forall \mathbf{y} \in \mathcal{K}_2 \ , \quad \mathbf{w}^\top \mathbf{u}(\mathcal{O}(\mathbf{w}), \mathbf{y}) - h_S(\mathbf{w}) \leq 0. \tag{13.2}$$

We proceed with the formal proof of sufficiency, constructively specified in Algorithm 37. Notice that in this reduction, the functions f_t are concave, and the OCO algorithm is used for maximization.

Algorithm 37 OCO to Approachability reduction

Input: generalized vector game $\mathcal{K}_1, \mathcal{K}_2, \{\mathbf{u}(\cdot,\cdot)\}$, set S, best response oracle $\mathcal{O}$, OCO algorithm $\mathcal{A}$
Set $\mathcal{K} = \mathbb{B} \in \mathbb{R}^d$ to be the unit Euclidean ball, as decision set for $\mathcal{A}$.
for $t = 1, \ldots, T$ **do**
    set $f_t(\mathbf{w}) = \mathbf{w}^\top \mathbf{u}_{t-1} - h_S(\mathbf{w})$
    Query $\mathcal{A}$: $\mathbf{w}_t \leftarrow \mathcal{A}(f_1, \ldots, f_{t-1})$
    Query $\mathcal{O}$: $\mathbf{x}_t \leftarrow \mathcal{O}(\mathbf{w}_t)$
    Observe $\mathbf{y}_t$ and let $\mathbf{u}_t = \mathbf{u}(\mathbf{x}_t, \mathbf{y}_t)$
end for
return $\bar{\mathbf{u}}_T = \frac{1}{T}\sum_{t=1}^T \mathbf{u}(\mathbf{x}_t, \mathbf{y}_t)$

Theorem 13.7. *Algorithm 37, with input OCO algorithm $\mathcal{A}$, returns the vector $\bar{\mathbf{u}}_T = \frac{1}{T}\sum_{t=1}^T \mathbf{u}(\mathbf{x}_t, \mathbf{y}_t)$ that approaches the set S at a rate of*

$$\mathbf{Dist}(\bar{\mathbf{u}}_T, S) \leq \frac{\mathrm{Regret}_T(\mathcal{A})}{T}$$

Proof. Notice that equation (13.2) implies that for $\mathbf{w}_t$ as defined in the algorithm, we have

$$\forall \mathbf{y} \in \mathcal{K}_2 \quad , \quad \mathbf{w}_t^\top \mathbf{u}(\mathcal{O}(\mathbf{w}_t), \mathbf{y}) - h_S(\mathbf{w}_t) \leq 0.$$

This implies that for any t,

$$f_t(\mathbf{w}_t) = \mathbf{w}_t^\top \mathbf{u}(\mathcal{O}(\mathbf{w}_t), \mathbf{y}_t) - h_S(\mathbf{w}_t) \leq 0. \tag{13.3}$$

Therefore we have, using Lemma 13.5,

$$\begin{aligned}
\mathbf{Dist}(\bar{\mathbf{u}}_T, S) &= \max_{\|\mathbf{w}\| \leq 1} \left\{ \mathbf{w}^\top \bar{\mathbf{u}}_T - h_S(\mathbf{w}) \right\} \\
&= \max_{\mathbf{w}^\star \in \mathcal{K}} \tfrac{1}{T} \textstyle\sum_t f_t(\mathbf{w}^\star) && \text{definition of } f_t \\
&\leq \tfrac{1}{T} \textstyle\sum_t f_t(\mathbf{w}_t) + \frac{\mathrm{Regret}_T(\mathcal{A})}{T} && \text{OCO guarantee of } \mathcal{A} \\
&\leq \frac{\mathrm{Regret}_T(\mathcal{A})}{T} && \text{equation (13.3)}
\end{aligned}$$

□

This theorem explicitly relates OCO with approachability, and since we have already proved the existence of efficient OCO algorithms in this text, it can be used to formally prove Blackwell's theorem. Completing the details is left as an exercise.

13.3 From Approachability to Online Convex Optimization

In this subsection we show the converse reduction: given an approachability algorithm, we design an OCO algorithm with no loss of computational efficiency. This direction was essentially shown by Blackwell for discrete decision problems, as described in more detail in the bibliographic section. We prove it here in the full generality of OCO.

Formally, given an approachability algorithm $\mathcal{A}$, denote by $\mathbf{Dist}_T(\mathcal{A})$ an upper bound on its rate of convergence to the set S as a function of the number of iterations T. That is, for a given vector game, denote by $\bar{\mathbf{u}}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{u}(\mathbf{x}_t, \mathbf{y}_t)$ the average reward vector. Then $\mathcal{A}$ guarantees

$$\mathbf{Dist}(\bar{\mathbf{u}}_T, S) \leq \mathbf{Dist}_T(\mathcal{A}) \ , \ \lim_{T \mapsto \infty} \mathbf{Dist}_T(\mathcal{A}) = 0.$$

Given an approachability algorithm $\mathcal{A}$, we henceforth create an OCO algorithm with vanishing regret.

13.3.1 Cones and polar cones

Approachability is in a certain geometric sense a dual to OCO. To see this, we require several geometric notions, that are explicitly required for the reduction from approachability to OCO.

For a given convex set $\mathcal{K} \subseteq \mathbb{R}^d$, we define its cone as the set of all vectors in $\mathcal{K}$ multiplied by a non-negative scalar,

$$\text{cone}(\mathcal{K}) = \{c \cdot \mathbf{x} \mid \mathbf{x} \in \mathcal{K}, 0 \leq c \in \mathbb{R}\}.$$

The notion of a convex cone is not strictly required for the proofs below, but they are commonly used in the context of approachability. The polar set to a given set $\mathcal{K} \subseteq \mathbb{R}^d$ is defined to be

$$\mathcal{K}^0 \stackrel{\text{def}}{=} \{\mathbf{y} \in \mathbb{R}^d \ \text{ s.t. } \ \forall \mathbf{x} \in \mathcal{K} \ , \ \mathbf{x}^\top \mathbf{y} \leq 0\}.$$

It is left as an exercise to prove that $\mathcal{K}^0$ is a convex set, and that for cones, the polar to the polar is the original set.

Henceforth we need the extension of a convex set defined as follows. Denote by $1 \oplus \mathcal{K}$ as the direct sum of the scalar one and the set $\mathcal{K}$, i.e., all vectors of the form $\tilde{\mathbf{x}} = 1 \oplus \mathbf{x}$ for $\mathbf{x} \in \mathcal{K}$. Denote the bounded polar extension of a set $\mathcal{K}$ by

$$Q(\mathcal{K}) = (1 \oplus \mathcal{K})^0.$$

That is, we take all points in the polar set to the direct sum $1 \oplus \mathcal{K}$.

This definition of the polar set gives rise to the following quantitative characterization.

Lemma 13.8. *Let $\mathbf{y} \in \mathbb{R}^{d+1}$ be such that $\mathbf{Dist}(\mathbf{y}, Q(\mathcal{K})) \leq \varepsilon$. Then, denoting by D the diameter of $\mathcal{K}$,*

$$\forall \tilde{\mathbf{x}} \in 1 \oplus \mathcal{K} \ , \ \mathbf{y}^\top \tilde{\mathbf{x}} \leq \varepsilon(D+1).$$

Proof. By definition of distance to a set, we have that $\mathbf{Dist}(\mathbf{y}, Q(\mathcal{K})) \leq \varepsilon$, implies the existence of a point $\mathbf{z} \in Q(\mathcal{K})$ such that $\|\mathbf{y} - \mathbf{z}\| \leq \varepsilon$. Thus, for all $\tilde{\mathbf{x}} \in 1 \oplus \mathcal{K}$, we have

$$\begin{aligned}
\mathbf{y}^\top \tilde{\mathbf{x}} &= (\mathbf{y} - \mathbf{z} + \mathbf{z})^\top \tilde{\mathbf{x}} \\
&\leq \|\mathbf{y} - \mathbf{z}\| \|\tilde{\mathbf{x}}\| + \mathbf{z}^\top \tilde{\mathbf{x}} && \text{Cauchy-Schwartz} \\
&\leq \varepsilon \|\tilde{\mathbf{x}}\| + \mathbf{z}^\top \tilde{\mathbf{x}} && \|\mathbf{y} - \mathbf{z}\| \leq \varepsilon \\
&\leq \varepsilon \|\tilde{\mathbf{x}}\| + 0 && \tilde{\mathbf{x}} \in 1 \oplus \mathcal{K}, \mathbf{z} \in (1 \oplus \mathcal{K})^0 \\
&\leq \varepsilon(1 + D).
\end{aligned}$$

□

13.3.2 The reduction

Algorithm 38 takes as an input a Blackwell approachability algorithm that guarantees, under the necessary and sufficient condition, convergence to a given set. It also takes as an input a set $\mathcal{K}$ for OCO.

The reduction considers a vector game with decision sets $\mathcal{K}, \mathcal{F}$ and approachability set $S = Q(\mathcal{K})$, and generates a sequence of decisions that guarantee low regret as we prove next.

Since this reduction creates the approachability set S as a function of $\mathcal{K}$, we need to prove that indeed the set S is approachable. We show this in the next subsection.

Theorem 13.9. *The reduction defined in Algorithm 38, for any input algorithm $\mathcal{A}$, produces an OLO algorithm $\mathcal{L}$ such that*

$$\text{Regret}(\mathcal{L}) \leq T(D+1) \cdot \mathbf{Dist}_T(\mathcal{A}).$$

Proof. The approachability algorithm guarantees $\mathbf{Dist}(\bar{\mathbf{u}_T}, S) \leq \mathbf{Dist}_T(\mathcal{A})$. Using the definition of S and Lemma 13.8 we have

Algorithm 38 Conversion of Approachability Alg. $\mathcal{A}$ to Online Convex Optimization Alg. $\mathcal{L}$

Input: closed, bounded and convex decision set $\mathcal{K} \subset \mathbb{R}^d$, approachability oracle $\mathcal{A}$.
Let: vector game w. $\mathcal{K}_1 = \mathcal{K}$, $\mathcal{K}_2 = \mathcal{F}$, and set $S := Q(\mathcal{K})$.
for $t = 1, \ldots, T$ **do**
    Query $\mathcal{A}$: $\mathbf{x}_t \leftarrow \mathcal{A}(f_1, \ldots, f_{t-1})$
    Let: $\mathcal{L}(f_1, \ldots, f_{t-1}) := \mathbf{x}_t$
    Receive: cost function f_t
    Construct reward vector $\mathbf{u}(\mathbf{x}_t, f_t) := \nabla_t^\top \mathbf{x}_t \oplus (-\nabla_t)$
end for

$$\begin{aligned}
&\forall \tilde{\mathbf{x}} \in Q(\mathcal{K}) \ . \ (D+1) \cdot \mathbf{Dist}_T(\mathcal{A}) \\
&\geq (\tfrac{1}{T} \textstyle\sum_{t=1}^T \mathbf{u}(\mathbf{x}_t, f_t))^\top \tilde{\mathbf{x}} \\
&\geq (\tfrac{1}{T} \textstyle\sum_{t=1}^T \mathbf{u}(\mathbf{x}_t, f_t))^\top (1 \oplus \mathbf{x}^\star) \\
&= \tfrac{1}{T} \textstyle\sum_{t=1}^T \nabla_t^\top \mathbf{x}_t - \tfrac{1}{T} \textstyle\sum_{t=1}^T \nabla_t^\top \mathbf{x}^\star \\
&\geq \tfrac{1}{T} \mathrm{Regret}_T(\mathcal{L}),
\end{aligned}$$

where the second inequality holds since the first inequality holds for every $\tilde{\mathbf{x}}$, in particular for the vector $1 \oplus \mathbf{x}^\star$.

□

13.3.3 Existence of a best response oracle

Notice that the reduction of this section from approachability to OCO does not require the best response oracle. However, Blackwell's approachability theorem does require this oracle as sufficient and necessary, and thus for the set S we constructed to be approachable at all, such an oracle needs to exist. This is what we show next.

Consider the vectors $\mathbf{u}_t$ constructed in the reduction. A best response oracle finds, for every vector $\mathbf{y}$, a vector $\mathbf{x}$ that guarantees $\mathbf{u}(\mathbf{x}, \mathbf{y}) \in S$. In our case, this translates to the condition

$$\forall f \in \mathcal{F} \ , \ \exists \mathbf{x} \in \mathcal{K} \ , \ \nabla f(\mathbf{x})^\top \mathbf{x} \oplus (-\nabla f(\mathbf{x})) \in (1 \oplus \mathcal{K})^0.$$

By definition of the polar set, this implies that for all $\tilde{\mathbf{x}} \in \mathcal{K}$, we have

$$\nabla f(\mathbf{x})^\top \mathbf{x} - \nabla f(\mathbf{x})^\top \tilde{\mathbf{x}} \leq 0.$$

In other words, the best response oracle corresponds to a procedure that given f, finds a vector $\mathbf{x}^\star$ such that

$$\forall \mathbf{x} \in \mathcal{K} \; . \; f(\mathbf{x}^\star) - f(\mathbf{x}) \leq \nabla f(\mathbf{x}^\star)^\top (\mathbf{x}^\star - \mathbf{x}) \leq 0.$$

This is an optimization oracle for the set $\mathcal{K}$!

13.4 Bibliographic Remarks

David Blackwell's celebrated Approachability Theorem was published in [Blackwell, 1956]. The first no-regret algorithm for a discrete action setting was given in a seminal paper by James Hannan in [Hannan, 1957] the next year. That same year, Blackwell pointed out [Blackwell, 1954] that his approachability result leads, as a special case, to an algorithm with essentially the same low-regret guarantee proven by Hannan. For Hannan's account of events see [Gilliland et al., 2010].

Over the years several other problems have been reduced to Blackwell approachability, including asymptotic calibration [Foster and Vohra, 1998], online learning with global cost functions [Even-Dar et al., 2009] and more [Mannor and Shimkin, 2008]. Indeed, it has been presumed that approachability, while establishing the existence of a no-regret algorithm, is strictly more powerful than regret-minimization; hence its utility in such a wide range of problems.

However, this was recently shown not to be the case. Abernethy et al. [2011] showed that approachability is in fact equivalent to OCO. This result is the basis of the material presented in this chapter. One side of their reduction was simplified and generalized in [Shimkin, 2016].

13.5 Exercises

1. Prove that Von Neumann's theorem does not hold for vector games, i.e., give an example of a vector game such that there is no single mixed strategy that ensures the vector payoff lies in a given set. Show that this is true even for approachable sets.

2. Prove that the polar set to a given convex set $\mathcal{K} \in \mathbb{R}^d$ is convex.

3. Prove that Blackwell's condition is necessary, i.e., prove that for a set S to be approachable in a given vector game, there must exist an oracle $\mathcal{O}$ such that

$$\forall \mathbf{y} \in \Delta_m \ , \ \mathbf{u}(\mathcal{O}(\mathbf{y}), \mathbf{y}) \in S.$$

4. Complete the proof of Blackwell's theorem using the first reduction of this chapter and the existence of OCO algorithms. That is, prove that for a given vector game, the existence of an oracle as per the previous question, is sufficient for a set $S \subseteq \mathbb{R}^d$ to be approachable.

Notes

[1]Alternatively and equivalently, any performance metric for this setting should depend on the magnitude of the largest loss. This is the viewpoint taken later in the rest of this book.

[2]Here and henceforth, we denote as $[n]$ the set of integers $\{1, ..., n\}$.

[3] We will discuss projections with respect to other distance notions in chapter 5.

[4]this notation stands for the arguments that minimize the expression inside the brakets, which are a subset in $\mathbb{R}^d$.

[5]That $a_0 \leq 1$ follows from Lemma 2.4. For $t = 1$, $a_1 \leq \frac{1}{2}$ since $a_1 \leq a_0(1 - a_0)$ and $0 \leq a_0 \leq 1$. For the induction step, $a_t \leq a_{t-1}(1 - a_{t-1}) \leq \frac{1}{t}(1 - \frac{1}{t}) = \frac{t-1}{t^2} = \frac{1}{t+1}(\frac{t^2-1}{t^2}) \leq \frac{1}{t+1}$.

[6]This assumes T is even. T odd leads to the same constants.

[7]Such a representation may seem naïve at first as it completely ignores the words' order of appearance and their context. Extensions to capture these features are indeed studied in the Natural Language Processing literature.

[8]see bibliography for references to these results.

[9]see exercises.

[10]Historically, this lemma is known as the "FTL-BTL," which stands for follow-the-leader vs. be-the-leader. BTL is a hypothetical algorithm that predicts $\mathbf{x}_{t+1}$ at iteration t, where $\mathbf{x}_t$ is the prediction made by FTL. These terms were coined by Kalai and Vempala [Kalai and Vempala, 2005].

[11]In harmonic analysis of Boolean functions, a similar quantity is called "average sensitivity".

[12]One such example is the self-concordant barrier regularization which we shall explore in the next chapter.

[13]Taken from [Gass, 2006]

[14]In certain cases this can be $k \times \text{AdaptiveRegret}_T$, depending on the particular algorithm used.

[15]The results henceforth hold with a different constant if we replace one by a different scaling.

Bibliography

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. ISBN 0521833787.

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Applied Optimization. Springer, 2004.

Arkadi S. Nemirovski and David B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley UK/USA, 1983.

A.S. Nemirovskii. Interior point polynomial time methods in convex programming, 2004. Lecture Notes.

J.M. Borwein and A.S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. CMS Books in Mathematics. Springer, 2006. ISBN 9780387295701. URL `http://books.google.fr/books?id=TXWzqEkAa7IC`.

R.T. Rockafellar. *Convex Analysis*. Convex Analysis. Princeton University Press, 1997. ISBN 9780691015866. URL `http://books.google.co.il/books?id=1TiOka9bx3sC`.

Elad Hazan. A survey: The convex optimization approach to regret minimization. In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for Machine Learning*, pages 287–302. MIT Press, 2011.

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

Alexander Rakhlin. Lecture notes on online learning. Lecture Notes, 2009.

Alexander Rakhlin and Karthik Sridharan. Theory of statistical learning and sequential prediction. Lecture Notes, 2014.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936, 2003.

Elad Hazan. *Efficient Algorithms for Online Convex Optimization and Their Applications*. PhD thesis, Princeton University, Princeton, NJ, USA, 2006. AAI3223851.

N. Littlestone and M. Warmuth. The weighted majority algorithm. In *Proceedings of the 30th Annual Symposium on the Foundations of Computer Science*, pages 256–261, 1989.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012. doi: 10.4086/toc.2012.v008a006. URL `http://www.theoryofcomputing.org/articles/v008a006`.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55 (1):119–139, August 1997.

Thomas Cover. Universal portfolios. *Math. Finance*, 1(1):1–19, 1991.

Adam Kalai and Santosh Vempala. Efficient algorithms for universal portfolios. *J. Mach. Learn. Res.*, 3:423–440, March 2003. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=944919.944942`.

Amit Agarwal, Elad Hazan, Satyen Kale, and Robert E. Schapire. Algorithms for portfolio management based on the newton method. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 9–16, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. In *Machine Learning*, volume 69(2–3), pages 169–192, 2007.

Avrim Blum and Adam Kalai. Universal portfolios with and without transaction costs. *Mach. Learn.*, 35(3):193–205, June 1999. ISSN 0885-6125.

doi: 10.1023/A:1007530728748. URL `http://dx.doi.org/10.1023/A:1007530728748`.

Elad Hazan and Satyen Kale. On stochastic and worst-case models for investing. In *Advances in Neural Information Processing Systems 22*. MIT Press, 2009.

Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *J. Comput. Syst. Sci.*, 74(1):97–114, 2008. ISSN 0022-0000. doi: http://dx.doi.org/10.1016/j.jcss.2007.04.016.

Stephen Boyd. Lecture notes: Subgradient methods, January 2014.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3–4):231–357, 2015.

Elad Hazan. Lecture notes: Optimization for machine learning. *arXiv preprint arXiv:1909.03550*, 2019.

Zeyuan Allen-Zhu and Elad Hazan. Optimal black-box reductions between optimization objectives. *CoRR*, abs/1603.05642, 2016.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

Boris T. Polyak. *Introduction to optimization*. Optimization Software, Inc., New York, 1987.

Elad Hazan and Sham Kakade. Revisiting the polyak step size. *arXiv preprint arXiv:1905.00313*, 2019.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, 1992.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.

Amit Daniely. Complexity theoretic limitations on learning halfspaces. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 105–117, 2016.

Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009. ISBN 0521424267.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Inf. Comput.*, 132(1):1–63, 1997.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 09 1951. doi: 10.1214/aoms/1177729586. URL `http://dx.doi.org/10.1214/aoms/1177729586`.

Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1-2):365–397, 2012.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: primal estimated sub-gradient solver for svm. *Math. Program.*, 127(1):3–30, 2011a.

Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. *Journal of Machine Learning Research - Proceedings Track*, pages 421–436, 2011.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, 2012.

Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML*, 2013.

Louis Bachelier. Théorie de la spéculation. *Annales Scientifiques de l'École Normale Supérieure*, 3(17):21–86, 1900.

M. F. M. Osborne. Brownian motion in the stock market. *Operations Research*, 2:145–173, 1959.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In Gábor Lugosi and Hans-Ulrich Simon, editors, *COLT*, volume 4005 of *Lecture Notes in Computer Science*, pages 499–513. Springer, 2006. ISBN 3-540-35294-5.

Jacob Abernethy, Rafael M. Frongillo, and Andre Wibisono. Minimax option pricing meets black-scholes in the limit. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1029–1040, New York, NY, USA, 2012. ACM. URL `http://doi.acm.org/10.1145/2213977.2214070`.

Peter DeMarzo, Ilan Kremer, and Yishay Mansour. Online trading algorithms and robust option pricing. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 477–486, 2006. ISBN 1-59593-134-1.

Jyrki Kivinen and ManfredK. Warmuth. Averaging expert predictions. In Paul Fischer and HansUlrich Simon, editors, *Computational Learning Theory*, volume 1572 of *Lecture Notes in Computer Science*, pages 153–167. Springer Berlin Heidelberg, 1999.

Volodimir G Vovk. Aggregating strategies. *Proc. of Computational Learning Theory, 1990*, 1990.

Dylan J Foster, Satyen Kale, Haipeng Luo, Mehryar Mohri, and Karthik Sridharan. Logistic regression: The importance of being improper. In *Conference On Learning Theory*, pages 167–208. PMLR, 2018.

Tim van Erven, Peter D Grünwald, Nishant A Mehta, Mark D Reid, and Robert C Williamson. Fast rates in statistical and online learning. *Journal of Machine Learning Research*, 16(54):1793–1861, 2015.

Katy S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Mach. Learn.*, 43 (3):211–246, June 2001. ISSN 0885-6125.

Kurt Riedel. A sherman-morrison-woodbury identity for rank augmenting matrices with application to centering. *SIAM J. Mat. Anal.*, 12(1):80–95, January 1991.

A. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(3):173–210, 2001.

Jyrki Kivinen and Manfred K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, 2001.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

James Hannan. Approximation to bayes risk in repeated play. *In M. Dresher, A. W. Tucker, and P. Wolfe, editors, Contributions to the Theory of Games, volume 3*, pages 97–139, 1957.

Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):115–142, 2007.

Shai Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University of Jerusalem, 2007.

Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Proceedings of the 21st Annual Conference on Learning Theory*, pages 263–274, 2008.

Elad Hazan and Satyen Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. In *The 21st Annual Conference on Learning Theory (COLT)*, pages 57–68, 2008.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 257–269, 2010.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

H. Brendan McMahan and Matthew J. Streeter. Adaptive bound optimization for online convex optimization. In *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 244–256, 2010.

Vineet Gupta, Tomer Koren, and Yoram Singer. A unified approach to adaptive regularization in online and stochastic optimization. *arXiv preprint arXiv:1706.06569*, 2017.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Jacob Abernethy, Chansoo Lee, Abhinav Sinha, and Ambuj Tewari. Online linear optimization via smoothing. In *Proceedings of The 27th Conference on Learning Theory*, pages 807–823, 2014.

Jacob Abernethy, Chansoo Lee, and Ambuj Tewari. Perturbation techniques in online learning and optimization. In Tamir Hazan, George Papandreou, and Daniel Tarlow, editors, *Perturbations, Optimization, and Statistics*, Neural Information Processing Series, chapter 8. MIT Press, 2016. to appear.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58(5):527–535, 1952.

Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1): 48–77, 2003. doi: http://dx.doi.org/10.1137/S0097539701398375.

Jean-Yves Audibert and Sébastien Bubeck. Minimax policies for adversarial and stochastic bandits. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009.

Abraham Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 385–394, 2005.

Sébastien Bubeck, Ofer Dekel, Tomer Koren, and Yuval Peres. Bandit convex optimization:\sqrtt regret in one dimension. In *Conference on Learning Theory*, pages 266–278. PMLR, 2015.

Elad Hazan and Yuanzhi Li. An optimal algorithm for bandit convex optimization. *arXiv preprint arXiv:1603.04350*, 2016.

Sébastien Bubeck, Yin Tat Lee, and Ronen Eldan. Kernel-based methods for bandit convex optimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 72–85, 2017.

Varsha Dani, Thomas Hayes, and Sham Kakade. The price of bandit information for online optimization. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

Y. E. Nesterov and A. S. Nemirovskii. *Interior Point Polynomial Algorithms in Convex Programming*. SIAM, Philadelphia, 1994.

Ohad Shamir. On the complexity of bandit linear optimization. In *Conference on Learning Theory*, pages 1523–1551. PMLR, 2015.

Jacob Abernethy and Alexander Rakhlin. Beating the adaptive bandit with high probability. In *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.

Ofer Dekel, Ambuj Tewari, and Raman Arora. Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

Gergely Neu, András György, Csaba Szepesvári, and András Antos. Online markov decision processes under bandit feedback. *IEEE Trans. Automat. Contr.*, 59(3):676–691, 2014. doi: 10.1109/TAC.2013.2292137. URL `http://dx.doi.org/10.1109/TAC.2013.2292137`.

Jia Yuan Yu and Shie Mannor. Arbitrarily modulated markov decision processes. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pages 2946–2953, 2009.

E. Even-Dar, S. Kakade, and Y. Mansour. Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

S. Mannor and N. Shimkin. The empirical bayes envelope and regret minimization in competitive markov decision processes. *Mathematics of Operations Research*, 28(2):327–345, 2003.

J. Y. Yu, S. Mannor, and N. Shimkin. Markov decision processes with arbitrary reward processes. *Mathematics of Operations Research*, 34(3): 737–757, 2009.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Nathan Srebro. *Learning with Matrix Factorizations*. PhD thesis, Massachusetts Institute of Technology, 2004.

Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 713–719, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102441. URL `http://doi.acm.org/10.1145/1102351.1102441`.

R. Salakhutdinov and N. Srebro. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. In *NIPS*, pages 2056–2064, 2010.

J. Lee, B. Recht, R. Salakhutdinov, N. Srebro, and J. A. Tropp. Practical large-scale optimization for max-norm regularization. In *NIPS*, pages 1297–1305, 2010.

E. Candes and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772, 2009.

O. Shamir and S. Shalev-Shwartz. Collaborative filtering with the trace norm: Learning, bounding, and transducing. *JMLR - Proceedings Track*, 19:661–678, 2011.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:149–154, 1956.

Elad Hazan. Sparse approximate solutions to semidefinite programs. In *LATIN*, pages 306–316, 2008.

Martin Jaggi and Marek Sulovský. A simple algorithm for nuclear norm regularized problems. In *ICML*, pages 471–478, 2010.

Simon Lacoste-Julien, Martin Jaggi, Mark W. Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 53–61, 2013.

Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML*, 2013.

Miroslav Dudík, Zaïd Harchaoui, and Jérôme Malick. Lifted coordinate descent for learning with trace-norm regularization. *Journal of Machine Learning Research - Proceedings Track*, 22:327–336, 2012.

Zaïd Harchaoui, Matthijs Douze, Mattis Paulin, Miroslav Dudík, and Jérôme Malick. Large-scale image classification with trace-norm regularization. In *CVPR*, pages 3386–3393, 2012.

Elad Hazan and Satyen Kale. Projection-free online learning. In *ICML*, 2012.

Shai Shalev-Shwartz, Alon Gonen, and Ohad Shamir. Large-scale convex minimization with a low-rank constraint. In *ICML*, pages 329–336, 2011b.

Francis Bach, Simon Lacoste-Julien, and Guillaume Obozinski. On the equivalence between herding and conditional gradient algorithms. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1359–1366, New York, NY, USA, July 2012. Omnipress. ISBN 978-1-4503-1285-1.

Ambuj Tewari, Pradeep D. Ravikumar, and Inderjit S. Dhillon. Greedy algorithms for structurally constrained high dimensional problems. In *NIPS*, pages 882–890, 2011.

Dan Garber and Elad Hazan. Approximating semidefinite programs in sublinear time. In *NIPS*, pages 1080–1088, 2011.

Dan Garber and Elad Hazan. Playing non-linear games with linear oracles. In *FOCS*, pages 420–428, 2013.

Aurélien Bellet, Yingyu Liang, Alireza Bagheri Garakani, Maria-Florina Balcan, and Fei Sha. Distributed frank-wolfe algorithm: A unified framework for communication-efficient sparse learning. *CoRR*, abs/1404.2644, 2014.

Dan Garber. Faster projection-free convex optimization over the spectrahedron. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 874–882, 2016.

Zeyuan Allen-Zhu, Elad Hazan, Wei Hu, and Yuanzhi Li. Linear convergence of a frank-wolfe type algorithm over trace-norm balls. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6192–6201, 2017.

Guanghui Lan and Yi Zhou. Conditional gradient sliding for convex optimization. *SIAM Journal on Optimization*, 26(2):1379–1409, 2016.

Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pages 1263–1271. PMLR, 2016.

Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *arXiv preprint arXiv:1804.09554*, 2018.

Lin Chen, Christopher Harshaw, Hamed Hassani, and Amin Karbasi. Projection-free online optimization with stochastic gradient: From convexity to submodularity. In *International Conference on Machine Learning*, pages 814–823, 2018.

Jiahao Xie, Zebang Shen, Chao Zhang, Hui Qian, and Boyu Wang. Stochastic recursive gradient-based methods for projection-free online learning. *arXiv preprint arXiv:1910.09396*, 2019.

Jacek Kuczyński and Henryk Woźniakowski. Estimating the largest eigenvalue by the power and lanczos algorithms with a random start. *SIAM journal on matrix analysis and applications*, 13(4):1094–1122, 1992.

Zeyuan Allen-Zhu and Yuanzhi Li. Lazysvd: even faster svd decomposition yet without agonizing pain. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 982–990, 2016.

Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1396–1404, 2015.

John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. ISBN 0691119937.

L.V. Kantorovich. A new method of solving some classes of extremal problems. *Doklady Akad Sci USSR*, 28:211–214, 1940.

G. B. Dantzig. *Maximization of a Linear Function of Variables Subject to Linear Inequalities, in Activity Analysis of Production and Allocation*, chapter XXI. Wiley, New York, 1951.

Donald J. Albers, Constance Reid, and George B. Dantzig. An interview with george b. dantzig: The father of linear programming. *The College Mathematics Journal*, 17(4):292–314, 1986.

Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. ISBN 1886529191.

Jiri Matousek and Bernd Gärtner. *Understanding and using linear programming*. Springer Science & Business Media, 2007.

Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1–2):79 – 103, 1999.

Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

Eyal Even-dar, Yishay Mansour, and Uri Nadav. On the convergence of regret minimization dynamics in concave games. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 523–532, 2009.

Tim Roughgarden. Intrinsic robustness of the price of anarchy. *Journal of the ACM*, 62(5):32:1–32:42, November 2015.

Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995.

Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff. Sublinear optimization for machine learning. *J. ACM*, 59(5):23:1–23:49, November 2012. ISSN 0004-5411.

Elad Hazan, Tomer Koren, and Nati Srebro. Beating sgd: Learning svms in sublinear time. In *Advances in Neural Information Processing Systems*, pages 1233–1241, 2011.

Ilan Adler. The equivalence of linear programs and zero-sum games. *International Journal of Game Theory*, 42(1):165–177, 2013.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994. ISBN 0-262-11193-4.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. doi: 10.1017/CBO9781107298019.

Nick Littlestone. From on-line to batch learning. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, COLT '89, pages 269–284, 1989.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. Inf. Theor.*, 50(9):2050–2057, September 2006. ISSN 0018-9448.

N. Cesa-Bianchi and C. Gentile. Improved risk tail bounds for on-line algorithms. *Information Theory, IEEE Transactions on*, 54(1):386–390, Jan 2008.

Tong Zhang. Data dependent concentration bounds for sequential prediction algorithms. In *Proceedings of the 18th Annual Conference on Learning Theory*, COLT'05, pages 173–187, 2005.

Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, 1992.

Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. 1986.

Avi Wigderson. *Mathematics and Computation: A Theory Revolutionizing Technology and Science*. Princeton University Press, 2019. URL `http://www.jstor.org/stable/j.ctvckq7xb`.

Shay Moran and Amir Yehudayoff. Sample compression schemes for vc classes. *Journal of the ACM (JACM)*, 63(3):1–10, 2016.

Ofir David, Shay Moran, and Amir Yehudayoff. On statistical learning via the lens of compression. *arXiv preprint arXiv:1610.03592*, 2016.

Steve Hanneke, Aryeh Kontorovich, and Menachem Sadigurschi. Sample compression for real-valued learners. In *Algorithmic Learning Theory*, pages 466–488. PMLR, 2019.

Lee-Ad Gottlieb, Aryeh Kontorovich, and Pinhas Nisnevitch. Near-optimal sample compression for nearest neighbors. *IEEE Transactions on Information Theory*, 64(6):4120–4128, 2018.

Aryeh Kontorovich, Sivan Sabato, and Roi Weiss. Nearest-neighbor sample compression: efficiency, consistency, infinite dimensions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1572–1582, 2017.

Olivier Bousquet, Steve Hanneke, Shay Moran, and Nikita Zhivotovskiy. Proper learning, helly number, and an optimal svm bound. In *Conference on Learning Theory*, pages 582–609. PMLR, 2020.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Non-stationary stochastic optimization. *Operations research*, 63(5):1227–1244, 2015.

Lijun Zhang, Tianbao Yang, Zhi-Hua Zhou, et al. Dynamic regret of strongly adaptive methods. In *International conference on machine learning*, pages 5882–5891. PMLR, 2018.

Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Mach. Learn.*, 32(2):151–178, 1998. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1007424614876.

Olivier Bousquet and Manfred K. Warmuth. Tracking a small set of experts by mixing past posteriors. *J. Mach. Learn. Res.*, 3:363–396, 2003. ISSN 1533-7928.

Yoram Singer. Switching portfolios. In *UAI*, pages 488–495, 1998.

S.S. Kozat and A.C. Singer. Universal constant rebalanced portfolios with switching. In *IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP)*, volume 3, pages 1129–1132, 2007.

András György, Tamás Linder, and Gábor Lugosi. Tracking the best of many experts. In *in Proceedings of the 18th Annual Conference on Learning Theory, COLT 2005*, pages 204–216. Springer, 2005.

Elad Hazan and Comandur Seshadhri. Adaptive algorithms for online decision problems. In *Electronic colloquium on computational complexity (ECCC)*, volume 14, 2007.

Dmitry Adamskiy, Wouter M Koolen, Alexey Chernov, and Vladimir Vovk. A closer look at adaptive regret. *The Journal of Machine Learning Research*, 17(1):706–726, 2016.

Lijun Zhang, Tie-Yan Liu, and Zhi-Hua Zhou. Adaptive regret of convex and smooth functions. *arXiv preprint arXiv:1904.11681*, 2019.

Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *International Conference on Machine Learning*, pages 1405–1411. PMLR, 2015.

F.M.J. Willems and M. Krom. Live-and-die coding for binary piecewise i.i.d. sources. In *Proc. 1997 IEEE International Symposium on Information Theory*, page 68, 1997.

G. I. Shamir and N. Merhav. Low-complexity sequential lossless coding for piecewise-stationary memoryless sources. *IEEE Trans. Inf. Theor.*, 45(5): 1498–1519, September 2006. ISSN 0018-9448. doi: 10.1109/18.771150. URL `http://dx.doi.org/10.1109/18.771150`.

Parikshit Gopalan, TS Jayram, Robert Krauthgamer, and Ravi Kumar. Estimating the sortedness of a data stream. In *SODA*, volume 7, pages 318–327. Citeseer, 2007.

Oren Anava, Elad Hazan, Shie Mannor, and Ohad Shamir. Online learning for time series prediction. In *Conference on learning theory*, pages 172–184. PMLR, 2013.

Paula Gradu, Elad Hazan, and Edgar Minasyan. Adaptive regret for control of time-varying dynamics. *arXiv preprint arXiv:2007.04393*, 2020.

Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2): 197–227, July 1990.

Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.

Richard Olshen. A conversation with Leo Breiman. *Statistical Science*, 16 (2):184–198, 2001.

R.E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. Adaptive computation and machine learning. MIT Press, 2012. ISBN 9780262017183. URL http://books.google.co.uk/books?id=blSReLACtToC.

Adam Tauman Kalai, Yishay Mansour, and Elad Verbin. On agnostic boosting and parity learning. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 629–638, 2008.

Adam Tauman Kalai and Rocco A. Servedio. Boosting in the presence of noise. *J. Comput. Syst. Sci.*, 71(3):266–290, 2005. doi: 10.1016/j.jcss.2004.10.015. URL https://doi.org/10.1016/j.jcss.2004.10.015.

Varun Kanade and Adam Kalai. Potential-based agnostic boosting. In *Advances in neural information processing systems*, 2009.

Vitaly Feldman. Distribution-specific agnostic boosting. *arXiv preprint arXiv:0909.2927*, 2009.

Shai Ben-David, Philip M Long, and Yishay Mansour. Agnostic boosting. In *International Conference on Computational Learning Theory*, pages 507–516. Springer, 2001.

Nataly Brukhim, Xinyi Chen, Elad Hazan, and Shay Moran. Online agnostic boosting via regret minimization, 2020.

Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

Christian Leistner, Amir Saffari, Peter M Roth, and Horst Bischof. On robustness of on-line boosting-a competitive study. In *IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1362–1369. IEEE, 2009.

Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu. An online boosting algorithm with theoretical justifications. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1873–1880, 2012.

Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu. Boosting with online binary learners for the multiclass bandit problem. In *International Conference on Machine Learning*, pages 342–350, 2014.

Alina Beygelzimer, Satyen Kale, and Haipeng Luo. Optimal and adaptive algorithms for online boosting. In *International Conference on Machine Learning*, pages 2323–2331, 2015a.

Alina Beygelzimer, Elad Hazan, Satyen Kale, and Haipeng Luo. Online gradient boosting. In *Advances in neural information processing systems*, pages 2458–2466, 2015b.

Naman Agarwal, Nataly Brukhim, Elad Hazan, and Zhou Lu. Boosting for dynamical systems. *arXiv preprint arXiv:1906.08720*, 2019.

Young Hun Jung, Jack Goetz, and Ambuj Tewari. Online multiclass boosting. In *Advances in neural information processing systems*, pages 919–928, 2017.

Young Hun Jung and Ambuj Tewari. Online boosting algorithms for multilabel ranking. In *International Conference on Artificial Intelligence and Statistics*, pages 279–287, 2018.

Nataly Brukhim and Elad Hazan. Online boosting with bandit feedback. *arXiv preprint arXiv:2007.11975*, 2020.

Robert M. Freund, Paul Grigas, and Rahul Mazumder. A new perspective on boosting in linear regression via subgradient optimization and relatives. *The Annals of Statistics*, 45(6):2328 – 2364, 2017.

Chu Wang, Yingfei Wang, Robert Schapire, et al. Functional frank-wolfe boosting for general loss functions. *arXiv preprint arXiv:1510.02558*, 2015.

Elad Hazan and Karan Singh. Boosting for online convex optimization, 2021.

Amir Beck. *First-order methods in optimization*. SIAM, 2017.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824, 2008.

Alexander Rakhlin and Karthik Sridharan. Bistro: An efficient relaxation-based method for contextual bandits. In *ICML*, pages 1977–1985, 2016.

Vasilis Syrgkanis, Haipeng Luo, Akshay Krishnamurthy, and Robert E Schapire. Improved regret bounds for oracle-based adversarial contextual

bandits. In *Advances in Neural Information Processing Systems*, pages 3135–3143, 2016a.

Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert Schapire. Efficient algorithms for adversarial contextual learning. In *International Conference on Machine Learning*, pages 2159–2168, 2016b.

Li Zhou. A survey on contextual multi-armed bandits. *arXiv preprint arXiv:1508.03326*, 2015.

Djallel Bouneffouf and Irina Rish. A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*, 2019.

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

D. Blackwell. Controlled random walks. In *Proceedings of the International Congress of Mathematicians*, volume 3, pages 336–338, 1954.

Dennis Gilliland, RV Ramamoorthi, and James Hannan. A conversation with james hannan. *Statistical Science*, pages 126–144, 2010.

D. P Foster and R. V Vohra. Asymptotic calibration. *Biometrika*, 85(2): 379, 1998.

E. Even-Dar, R. Kleinberg, S. Mannor, and Y. Mansour. Online learning for global cost functions. In *22nd Annual Conference on Learning Theory (COLT)*, 2009.

S. Mannor and N. Shimkin. Regret minimization in repeated matrix games with variable stage duration. *Games and Economic Behavior*, 63(1):227–258, 2008. ISSN 0899-8256.

Jacob Abernethy, Peter L Bartlett, and Elad Hazan. Blackwell approachability and no-regret learning are equivalent. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 27–46, 2011.

Nahum Shimkin. An online convex optimization approach to blackwell's approachability. *The Journal of Machine Learning Research*, 17(1):4434–4456, 2016.

Saul I. Gass. Ifors' operational research hall of fame: John von neumann. *International Transactions in Operational Research*, 13(1):85–90, 2006.