Εθνικο Μετσοβιο Πολυτεχνειο
Τμημα Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Λογικης και Επιστημης Υπολογιστων

Ανταγωνιστικοί Αλγόριθμοι για προβλήματα Άμεσης Κυρτής Βελτιστοποίησης με κόστη μετάβασης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μηνάς Χάτζος

Επιβλέπων: Δημήτρης Φωτάκης
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ανταγωνιστικοί Αλγόριθμοι για προβλήματα Άμεσης Κυρτής Βελτιστοποίησης με κόστη μετάβασης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μηνάς Χάτζος

Επιβλέπων: Δημήτρης Φωτάκης
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 16 Ιουλίου 2018.

.....	.....	.....
Δημήτρης Φωτάκης	Νικόλαος Παπασπύρου	Αριστείδης Παγουρτζής
Επίκουρος Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π	Αναπληρωτής Καθηγητής Ε.Μ.Π .

Αθήνα, Ιούλιος 2018

.....

Μηνάς Χάτζος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Η παρούσα διπλωματική εργασία αποτελεί το σημείο λήξης των σπουδών μου, έπειτα από 6 χρόνια, στην σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του ΕΜΠ.

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα μου, καθ. Δημήτρη Φωτάκη, για την εμπιστοσύνη που μου έδειξε, την υποστήριξη που μου παρείχε κατά την διάρκεια εκπόνησης της διπλωματικής εργασίας καθώς και τις πολύτιμες συμβουλές του σχετικά με τους μελλοντικούς ακαδημαϊκούς μου στόχους.

Επιπλέον τους φίλους που απέκτησα κατά την διάρκεια των σπουδών μου. Για τις ευχάριστες στιγμές που περάσαμε μαζί καθώς και τις συζητήσεις μας περί Μαθηματικών και Επιστήμης Υπολογιστών οι οποίες με ώθησαν να αγαπήσω και να ασχοληθώ περισσότερο με αυτά τα δύο αντικείμενα.

Τέλος, τους γονείς μου, Κωνσταντίνο και Γεωργία, για την ανιδιοτελή τους αγάπη όλα αυτά τα χρόνια. Χωρίς την αδιάκοπη υποστηριξή τους, το παρών κείμενο δεν θα είχε γραφτεί ποτέ.

Περίληψη

Στην παρούσα εργασία, μελετάμε ένα πρόβλημα το οποίο ανήκει στην τομή της περιοχών τών άμεσων αλγορίθμων και της άμεσης μάθησης με το όνομα Άμεση Κυρτή Βελτιστοποίηση (ΑΚΒ) με κόστη μετάβασης (ή αλλιώς, Ομαλή Άμεση Κυρτή Βελτιστοποίηση). Αυτή είναι μια εκδοχή ενός προβλήματος Άμεσης Κυρτής Βελτιστοποίησης οπού ο παίκτης, εκτός απο το αντικειμενικό κόστος, πληρώνει ένα κόστος λόγω αλλαγής των αποφασεών του μεταξύ συνεχόμενων γύρων, όπως και στο πρόβλημα των Μετρικών Συστημάτων Εργασίας. Αρχικά, αφού εισάγουμε τον αναγνώστη σε βασικά στοιχεία των άμεσων αλγορίθμων και της κυρτής βελτιστοποίησης, παρουσιάζουμε τα πεδία της άμεσης μάθησης και της άμεσης κυρτής βελτιστοποίησης που αποτελούν τις βάσεις του κύριου προβλήματος. Για το πρόβλημα της ΑΚΒ με κόστη μετάβασης, στοχεύουμε στον σχεδιασμό ανταγωνιστικών αλγορίθμων. Πιο συγκεκριμένα, εστιάζουμε στην περίπτωση όπου το κόστος μετάβασης είναι νόρμα. Για γενικές κυρτές συναρτήσεις, η βέλτιστη τιμή του ανταγωνιστικού λόγου που μπορούμε να επιτύχουμε εξαρτάται απο την διάσταση την εισόδου. Ως εκ τούτου, για να επιτύχουμε ένα σταθερό ανταγωνιστικό λόγο θα πρέπει να εξετάσουμε ειδικές περιπτώσεις κυρτών συναρτήσεων. Για αυτόν τον σκοπό, αναλύουμε έναν άμεσο αλγόριθμο που παίρνει μια απόφαση η οποία ισορροπεί το αντικειμενικό κόστος και το κόστος μετάβασης σε κάθε γύρο. Επιπλέον, εξετάζουμε την ειδική περίπτωση όταν η αντικειμενική συνάρτηση είναι γραμμική, το σύνολο αποφάσεων περιγράφεται απο περιορισμούς κάλυψης και η συνάρτηση μετάβασης είναι η l_1 νόρμα. Τέλος, αναφέρουμε ανοικτά προβλήματα στην εν λόγω ερευνητική περιοχή.

Λέξεις-Κλειδιά: Άμεσοι αλγόριθμοι, Κυρτή βελτιστοποίηση, Άμεση μάθηση, Ανταγωνιστική ανάλυση, Συναρτήσεις μετάβασης

Abstract

We study a problem which is of interest to both the algorithmic and the online learning community, named Online Convex Optimization with Switching Cost (or Smoothed Online Convex Optimization). This is a version of an online convex optimization (OCO) problem where the learner, apart from the objective cost, suffers a loss for changing her decisions between successive rounds, similarly to Metrical Task Systems. Initially, after introducing the reader to basic concepts in online algorithms and convex optimization, we perform a review on the field of online learning and online convex optimization which serves as the basis of our main problem. For OCO with switching cost, we are mainly interested in the design of competitive algorithms. In particular, we are focused on the case where the switching cost is a norm. For general convex functions, the optimal competitive ratio depends on the dimension of the instance, thus we need to constrain ourselves in certain types of convex objectives in order to achieve a constant competitive ratio. For this purpose, we analyze an algorithm that makes a decision which balances the objective and the switching cost. Moreover, we discuss the special case of linear objectives when the decision set is composed of linear covering constraints and the switching cost function is the l_1 norm. Finally, we mention open problems in this research area.

Keywords: Online algorithms, Convex optimization, Online learning, Competitive analysis, Switching cost functions

Contents

Introduction **1**

1 Online Computation **5**

1.1 Competitive Analysis 5

1.2 The Ski Rental Problem 6

1.2.1 A Deterministic Algorithm 7

1.2.2 A Randomized Algorithm 8

1.3 The Potential Function Method 9

1.4 Metrical Task Systems 11

2 Convex Optimization **13**

2.1 Basic concepts 14

2.2 Duality and Optimality Conditions 17

2.3 Gradient Descent 20

3 Online learning & Online Convex Optimization **26**

3.1 The Experts problem 26

3.1.1 The Weighted Majority Algorithm 27

3.2 A Unifying model 32

3.2.1 Online Gradient Descent 33

3.3 Competing against a dynamic comparator 37

4 Online Convex Optimization with Switching Cost **42**

4.1 Definition 42

4.2 The unidimensional case 45

4.2.1 A memoryless algorithm 46

4.2.2 An algorithm with memory 49
4.3 Higher dimensions 50
4.3.1 Minimizing Competitive Ratio 52
4.4 Covering constraints with l_1 switching cost 57
4.5 Open problems 62

List of Figures

1.1 Configurations for the online algorithm and OPT, Here we have $v = 3$ (the asterisks) 10

2.1 Examples of a convex and a non-convex set 14

2.2 Examples of a non-convex and a convex function 15

2.3 Illustration of theorem 2.5 16

2.4 Illustration of theorem 2.8. Observe the inner product of $-\nabla f(x)$ and $y - x$ for some $y \in X$ 17

2.5 Illustration of the Gradient Descent algorithm 21

2.6 projection onto a convex set 22

4.1 Illustration of Algorithm 1 for $f_t(u_t) = 0$ and $H_t > H_t^*$ 48

4.2 Geometric explanation of the above statement. Here we have $H_t(x_{t_1}) = H_t(x_{t_2})$ but $M_t(x_{t_1}) > M_t(x_{t_2})$ 53

4.3 Relation between x_t, x_{t-1}, x_t^* when $H_t > H_t^*$ 57

Introduction

In this thesis, we study the problem of Online Convex Optimization (OCO) with Switching Cost. This is a problem that is highly correlated with a typical problem in the area of online convex optimization but also shares elements with problems that arise from the community of online algorithms, such as the Metrical Task Systems (MTS). In this online problem, given a convex subset of $\mathbb{R}^n$, a player receives in each round a convex function $f_t(x)$ and makes a decision, a point $x_t \in X$. Now, apart from the cost of this objective function $f_t(x_t)$ the player suffers a cost for changing her decision between successive rounds $\|x_t - x_{t-1}\|$. Of course, the goal of the player is to minimize the total sum of her objective and switching costs. The setting resembles an online convex optimization problem where also in each round a convex function is revealed but the problem also is a not too restrictive special case of the very general online problem of MTS. In MTS, the euclidean space $\mathbb{R}^n$ is replaced by any metrical space, and the functions arriving online are arbitrary (possibly non-convex) functions.

Concerning MTS, a lot of research is pursued over the past decades for finite metric spaces. In [14] the authors showed that the lower bound on the competitive ratio for any deterministic algorithm over any n-point metric is $2n - 1$. They designed an algorithm, the Work-function algorithm which achieves exactly this competitive ratio. Randomization significantly improves the results. The best known lower bound for any algorithm is $\Omega(\frac{\log n}{\log^2 \log n})$ and the algorithm presented in [25] achieves a competitive ratio of $O(\log^2 n \log \log n)$. For the uniform metric (all the states are the same distance from each other), in [14] it was proven that the lower bound for any algorithm is H_n, the nth harmonic number. Subsequently, in [1] the authors provide a $\log n + O(\log \log n)$ algorithm based on entropic regularization, which shares the same idea with the one we'll discuss for the continuous case in later chapters.

Moreover, MTS is shown to be connected to the experts problem in online learning, which we'll discuss in chapter 3. Several efforts are made to provide algorithms for both problems simultaneously. The authors in [12] initiated this direction by providing an analytic framework that connects online learning and metrical task systems. More recently, in [19] the authors employed a primal-dual technique to develop an algorithm that, for

the first time, provided a unified approach for algorithm design across competitive ratio and regret in the MTS setting, over a discrete action space.

Although the MTS problem is heavily studied for many years and researchers continue to do so, the special case of OCO with switching cost has received attention only in recent years. Until very recently, there were results only for the unidimensional case of the problem. It was the first time in [35] that a 3-competitive algorithm was developed for the 1d case. Subsequently, in [8] a simpler 3-competitive algorithm was given along with a 2-competitive algorithm that utilizes past information, in contrast with the other ones. The first step towards the development of algorithms for higher dimensions can be found in [4] where the authors study the problem of Convex Body Chasing (first appeared in [26]), which is connected to OCO with switching costs. Using this connection, they provide a constant competitive ratio algorithm for the problem in 2 dimensions and show that any algorithm attains a competitive ratio of $\sqrt{d}$ when the switching cost is the l_2 norm. Recently, in [24] the authors restrict the class of objective functions the adversary may bring in order to break through the $\sqrt{d}$ lower bound and they provide a constant competitive ratio algorithm for l_2 switching cost in any dimension. This work also is the first one to generalize this result to other switching costs, such as l_p norms and more generalized norms (such as the Mahalanobis Distance) utilizing ideas from the Mirror Descent algorithm.

Furthermore, there is substantial work for the problem of OCO with Switching Cost when in time t the algorithm does not know only the function f_t as we mentioned earlier but also the functions $f_t, f_{t+1}, ..., f_{t+W}$ and hence, intuitively, the problem becomes substantially easier. In [23] the dependence of competitive ratio on W is studied and the conclusion that a constant competitive ratio can be achieved when future cost functions are known is drawn. Moreover, in [38] an algorithm with a competitive ratio of $1+O(\frac{1}{W})$ is developed.

The problem has find numerous applications during the past years that are modeled as online convex optimization problems in areas such as learning, control and networks where the aim is to minimize a convex function in each round and maintain a stable solution, a solution that doesn't change much throughout time. For instance, in right-sizing power-proportional data centers [36], [39]. In these applications, the data center consists of a homogenous collection of servers that are speed scalable and that they may powered down. In a data center, there are typically sufficiently many servers so that the problem can be reasonably be modeled a continuous one. The load on the data center changes through time and in each round there is a specific number of servers that should be operational and the operation cost is given by a convex function. However, there is a fixed cost (energy, for instance) for powering a server either on or off which is modeled as a switching cost. Of course, the load in each round arrives in an online fashion and the

designer aims at constructing an algorithm that performs well against the best offline strategy. Other fields of application include management of electrical vehicle charging [33], video streaming [32] and power generation planning [7].

Other related work, apart from the work on finite metric spaces for the problem of MTS which we discuss briefly in the next chapter and which can be considered the "discrete" counterpart on the research of online convex optimization with switching cost there is other work at other related problems of discrete nature which are special cases of MTS. However, utilizing algorithms for MTS for these problems does not lead to good bounds and thus an approach that takes into consideration the exact nature of the problem is employed. In [27] the authors analyze the online multistage matroid maintanance problem. A well known example of this is when one aims to maintain a minimum spanning tree (MST) in a graph where the weights of the edges are chosen arbitrary in each round and there is a switching cost for changing an edge. To encounter the online problem, the authors firstly solve an Online Linear Optimization problem with a specific type of constraints in order to maintain feasible fractional MST solution and then they provide an integer solution through rounding.

Moreover, there are other problems that lie in the intersection of convex optimization and online algorithms. Independently in [6] and [20], the respective authors provide a competitive algorithm for a convex optimization problem where the covering costraints arrive in an online fashion (i.e the rows of A of the equation $(Ax = b)$ arrive online).

In Chapter 1, we introduce the reader to basic elements of Online Computation that are related to our main problem. We explain how things work in an online setting through the ski-rental problem which is a special case of OCO with switching cost, we discuss briefly the MTS problem and we introduce through an example the reader to the potential function method which is heavily employed in proofs that are related to OCO with switching cost.

In Chapter 2, we discuss elements of Convex optimization including basic facts about convex sets and functions. Moreover, we discuss Duality and KKT conditions, which are powerful tools in the analysis of online algorithms. Finally, we rigorously analyze the Gradient Descent algorithm which is the basis for many algorithms that are found in the context of Online Convex Optimization.

In Chapter 3, we start by introducing the reader to the area of online learning through the well-known experts problem which serves as the motivation of developing the framework of online convex optimization. The OCO model can be used to model many online learning problems. Through the regret metric, we firstly discuss how the algorithm of online gradient descent performs in the case where the player competes against a static strategy. In the final paragraph, we analyze again an online gradient descent algorithm

which performs well against a dynamic strategy, the type of strategy which we'll encounter in the following chapter.

In Chapter 4, we delve into the problem of Online Convex Optimization with Switching Cost. After defining the problem and discussing some of its aspects, we start with the study of the problem in 1 dimension where we analyze an optimal 3-competitive algorithm which has a simple idea: Balance between the function's cost and the switching cost in time t. For higher dimensions, we prove a lower bound which shows that every algorithm when the switching cost is the l_2 norm is $\Omega(\sqrt{d})$ competitive. For this reason, we perform a beyond worst case analysis utilizing functions that grow at least linearly away from the minimizer to prove a constant competitive ratio using the same idea of "Balance". Moreover, we discuss a special case of OCO with switching cost, the case of linear objectives with covering constraints and l_1 switching cost which has found numerous applications in online combinatorial optimization in recent years. Finally, we state some open problems related to OCO with switching cost that we believe are interesting research directions for the future.

Chapter 1

Online Computation

The problems which are of interest to us in this thesis need to be solved in an *online* fashion. Its counterpart, the *offline* setting, the traditional setting in the analysis and design of algorithms, assumes that complete information of the problem is known to the designer from the beginning. Unlike this setting, in online computation, one aims at designing algorithms where the input is revealed piece-by-piece. The algorithm has to respond immediately when new information arrives without the knowledge of future information. Furthermore, when a decision is taken, it cannot be revoked.

In the *offline* setting, the goal is to design algorithms that provide an optimal solution, for instance, one that minimizes (or maximizes) a certain objective. We evaluate an algorithm based on how close its solution is to an optimal solution. For example, an α-approximation algorithm, with $\alpha > 1$, for a **NP**-hard minimization problem is defined as a polynomial-time algorithm that for every instance of the problem, provides a solution that its value is within a factor of α of the value of an optimal solution.

Now, a natural question arises: How to evaluate the performance of an online algorithm? A standard measure is the *competitive ratio* which compares the performance of an online algorithm to an optimal offline solution. The field of *competitive analysis* aims at designing this kind of algorithms. For an extensive analysis of online computation, algorithms and competitive analysis we refer the reader to [13] which is from where this chapter is based.

1.1 Competitive Analysis

We proceed with the definition of an c-competitive algorithm which is similar to the one for approximation algorithms.

Definition 1.1. Let $c \geq 1$ be a real number. An online algorithm is said to be c-competitive if for each instance of a minimization problem I, for any sequence of inputs,

it outputs a solution of cost at most $c \cdot \mathrm{OPT}(I) + \alpha$ where $\mathrm{OPT}(I)$ is the value of an optimal offline solution and a is constant. For $a \leq 0$ we say that the algorithm is strictly c-competitive.

The infinum over all c such that the algorithm is c-competitive is called the *competitive ratio* of the algorithm. The definition of competitiveness is similar for maximization problems. We should note at this point that the competitive ratio isn't the only metric we will use to evaluate the performance of an online algorithm. Unlike the competitive ratio which originates from the algorithmic community, in the learning community, the *Regret* metric is employed. In subsequent chapters, we will comment on the differences between the two metrics.

An important concept in the field of competitive analysis is the *adversary*. The online problem can be seen as a game between the player, the one that designs the online algorithm (In the future, we'll constantly use the term *player* to describe the online algorithm), and an adversary which aims at constructing the worst possible input for the algorithm, in order to maximize its cost. A c-competitive algorithm produces a solution with cost no more that c times the optimal offline cost for every sequence of inputs the adversary provides. An adversary that knows the online algorithm, the probability distribution used by the online algorithm to make its random decisions but does not know the random choices, is called an *oblivious* adversary. This kind of adversary is the one we'll consider throughout this thesis.

1.2 The Ski Rental Problem

One of the most elementary problems in the field of online algorithms is the Ski Rental Problem. In this paragraph, we'll discuss the problem and analyze an optimal deterministic and an optimal randomized algorithm. The reason we choose this particular problem is twofold; First, we gently introduce the reader to the nature of online computation. Second, the ski rental problem is strongly connected to the problem of Online Convex Optimization with switching cost. In fact, the ski rental problem is a special case of the 1-dimensional case of the latter problem. In Chapter 4, we'll prove this reduction. This fact will be proven useful in order to provide lower bounds for our main problem.

The ski rental problem is as follows: Suppose you're going skiing in Mount Parnassus but you haven't yet decided how many days you'll stay. In fact, you will decide the number of skiing days at the very last day. You're faced with a question of whether to buy skis for B€ ($B \geq 1$) or to rent skis at the cost of 1€ per day. Of course, your goal

is to minimize the amount of money you'll spend. Let's define the number of days you'll eventually stay as n.

1.2.1 A Deterministic Algorithm

Due to the simplicity of the problem, all possible deterministic online algorithms can be described just by a integer j which is the day that you'll buy skis. If $j > n$, you never buy skis. Let's define as ALG_j the deterministic algorithm that buys skis at the start of day j.

Theorem 1.2. *ALG_B has a competitive ratio of $(2 - \frac{1}{B})$ for the ski rental problem.*

Proof. We consider two cases:

- If $n \leq B - 1$, the algorithm has cost n while the optimal offline cost is n
- If $n \geq B$, the algorithm pays $(B - 1) + B = 2B - 1$ while the optimal offline cost is B

Thus, in either case, the competitive ratio of is ALG_B is $\max\{\frac{n}{n}, \frac{2B-1}{B}\} = 2 - \frac{1}{B}$ □

Is this the better we can do? The answer is negative. In fact,

Theorem 1.3. *Every deterministic online algorithm for the ski rental problem cannot attain a competitive ratio less than $2 - \frac{1}{B}$.*

Proof. The intuition behind the proof is that the algorithm has to buy skis at some point in time. If it doesn't, the skiing days may go on indefinitely, and the competitive ratio is unbounded. In case the algorithm buys at day j, then the adversary chooses to terminate the skiing days at day j. The cost of the optimal offline solution is $min\{j, B\}$.

- If $j \leq B$, then the competitive ratio is $\frac{j-1+B}{j} = 1 + \frac{B-1}{j} \geq 1 + \frac{B-1}{B} = 2 - \frac{1}{B}$
- If $j \geq B$ then the competitive ratio is $\frac{j-1+B}{B} \geq \frac{2B-1}{B} = 2 - \frac{1}{B}$

Thus, the competitive ratio is at least $2 - \frac{1}{B}$. □

The analysis for deterministic algorithms is complete in the classical sense. But what about a randomized algorithm? Can randomization be proven useful against an oblivious adversary?

1.2.2 A Randomized Algorithm

The ski rental is an example of an online problem where randomization improves the competitive ratio that an online algorithm can attain. Intuitively, since the worst case input for the online algorithm is based on the round the player buys and there is not a case that is bad for every possible deterministic algorithm, with the adversary not knowing for sure the time the player buys, the competitive ratio should be improved with a randomized algorithm. The definition of competitiveness easily extends to online randomized algorithms. Instead of using the cost of the online algorithm, we simply consider the expected value of its cost.

The proposed algorithm works as follows: Before the skiing days begin, the player chooses to buy skis after skiing for $0 \leq j \leq B-1$ days where j is sampled from a probability distribution. Intuitively, the probability density function has support the set $\{0, ..., B-1\}$ otherwise for large values of j the algorithm would have an unbounded competitive ratio. We describe the algorithm below:

Algorithm 2: **RandSki**

1: **Input:** $B > 1$, the cost of buying skis
2: Let $\rho = \frac{B}{B-1}$.
3: Sample according to the distribution $P[j = x] = \alpha\rho^x$, $x \in \{0, 1, ..., B-1\}$
4: **Output:** $j + 1$, the day you'll buy skis

where $\alpha = \frac{\rho-1}{\rho^B-1}$ is a scaling factor in order to have a valid probability density function.

Theorem 1.4. *The competitive ratio of **RandSki** is* $\frac{e}{e-1}$.

Proof. Because $j \leq B-1$, meaning that at the start of day B you will have bought skis, notice that the case $(n > B)$ is equivalent to $(n = B)$ (The optimal offline cost is B in both cases). For a specific day j and number of skiing days n, the cost of the algorithm is $j + B$ if $0 \leq j \leq n-1$ and n if $n \leq j \leq B$. Thus, the expected value of the cost is:

$$E[cost] = \sum_{j=0}^{n-1} \alpha\rho^j(j+B) + \sum_{j=n}^{B-1} \alpha\rho^j n =$$

$$\alpha \sum_{j=0}^{n-1} j\rho^j + \alpha B \sum_{j=0}^{n-1} \rho^j + \alpha n \sum_{j=n}^{B-1} \alpha\rho^j =$$

$$\alpha \frac{(n-1)\rho^{n+1} - n\rho^n + \rho}{(\rho-1)^2} + \alpha B \frac{\rho^B - 1}{\rho - 1} + \alpha n \frac{\rho^B - \rho^n}{\rho - 1} =$$

$$\frac{\alpha}{(\rho-1)^2}((n-1)\rho^{n+1} - n\rho^n + \rho + \rho^{n+1} - \rho + n(\rho-1)(\rho^B - \rho^n)) =$$

$$\frac{\alpha}{(\rho-1)^2}((\rho-1)n\rho^B) = \frac{\alpha}{\rho-1}n\rho^B = \frac{\rho^B}{\rho^B-1}n$$

Since $n \leq B$, the optimal offline cost is n. Hence the competitive ratio is bounded above by the value $\frac{\rho^B}{\rho^B-1} = 1+\frac{1}{\rho^B-1}$. Notice that $\rho^B = (\frac{B}{B-1})^B$ is an increasing sequence which converges to e. Thus, the competitive ratio of the algorithm is $\frac{e}{e-1} \approx 1.58$ and is attained as $B \to \infty$. □

The randomized algorithm performs better than the deterministic we analyzed earlier for $B \geq 3$. Furthermore, $\frac{e}{e-1}$ is the optimal competitive ratio that any online algorithm can attain for the problem. However, we'll omit the proof of the lower bound.

1.3 The Potential Function Method

We'll now discuss one of the most powerful tools for analyzing online algorithms and proving competitiveness results. The Potential Function Method will be extensively used in the analysis of the OCO problem with switching cost in Chapter 4. The method is based on a function, the potential, Φ which maps the current configuration (the state) of the player and the adversary in round i to a non-negative number and it shows how close are the two configurations. The bigger the potential, the bigger the difference. Let Φ_i be the potential on round i. We define the amortized cost a_i as:

$$\alpha_i = c_i + \Phi_i - \Phi_{i-1}$$

Where c_i is the cost of the online algorithm in round i. To prove that an online algorithm is c-competitive, we could bound the cost in round i by $c \cdot \text{OPT}(i)$ where $\text{OPT}(i)$ is the cost of the optimal offline algorithm in round i, for every round. However, that does not always work since the cost of the player in round i may be higher or lower than $c \cdot \text{OPT}(i)$, but the total cost is bounded above $c \cdot \text{OPT}(i)$. That's where the amortized cost comes into play; Notice that:

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} a_i + \Phi_0 - \Phi_n \leq \sum_{i=1}^{n} a_i + \Phi_0$$

Usually, $\Phi_0 = 0$ since the player and the adversary start with the same configuration. Hence, if we were able to show that for every round i that $a_i \leq c \cdot \text{OPT}(i)$ then the online algorithm is c-competitive.

To make things more clear, we'll analyze an online algorithm for the *list accessing problem* using the potential function method. The problem is defined as follows: Suppose you are given a list L with m elements. Requests for the list elements arrive online and the cost of answering the request is the position of the element in the list. Hence for an element is position k, the cost of the algorithm is k. In order to minimize the cost, there are two rules for reorganizing the list:

- After the request of element x, the element can be moved to any chosen position closer to the front at no cost.
- We can also change the position of adjacent elements at unit cost.

Now consider the following algorithm (MTF - move to the front): After the request of element i, the element is move to the front of the list at no cost.

Theorem 1.5. *The competitive ratio of MTF is* $2 - \frac{1}{m}$.

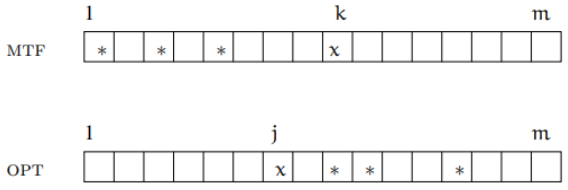


FIGURE 1.1: Configurations for the online algorithm and OPT, Here we have $v = 3$ (the asterisks)

Proof. We'll define the potential function as the number of pairs (a, b) such that element a is before b in the state of the online algorithm while b is before a in the state of the optimal offline algorithm (OPT). In round i, a request for element x arrives. Let v be the number of elements which are in front of x in the online state and after x in OPT and let the position of x to be k, j in the online state and in OPT respectively. The online algorithm moves the element to the front of the list. Thus, the increase in the potential is at most $(j-1) - v$ because $(j-1)$ new pairs can be created at most and v pairs are removed. Moreover, OPT, can increase the potential at most by $p - f$ where p is the paid exchanges and f the free exchanges to the front. That's because every paid exchange can increase the potential by 1 and any free decreases the number of inversions since the element x is already in front in the online state. Thus:

$$\begin{aligned} a_i &= c_i + \Phi_i + \Phi_i - 1 \\ &\leq k + j - 1 - v + p - f \\ &\leq j + p + k - v - 1 \end{aligned}$$

Now notice that $k - v \leq j$ and thus:

$$a_i \leq j + p + (j - 1) \leq 2(j + p) - 1 = 2OPT(i) - 1$$

Summing up for all rounds $i = 1, ..., T$ and noticing that $OPT \leq mT$ (The highest cost possible in a round is m and is attained when the element is placed in the end of the list) we get that:

$$\sum_{i=1}^{T} c_i \leq \sum_{i=1}^{T} a_i \leq 2OPT - T \leq 2OPT - \frac{OPT}{m} = (2 - \frac{1}{m})OPT$$

□

1.4 Metrical Task Systems

The problem of Metrical Task Systems (MTS) is one of the most well known problems in the field of online algorithms and it generalizes many problems like paging and OCO with switching cost. Consider a server which has to process a sequence of tasks that arrive in an online fashion. The server can be in one of a finite states and the cost of processing a task depends on the state of the server. When there is a state change, a cost is incurred. The problem of MTS is develop algorithms that choose to which state the server should migrate in order to minimize the total cost. We refer the reader to [14] for a more elaborate discussion on metrical task systems.

In particular, a metrical task system is defined as a pair (M, R) where $M = (X, d)$ is a metric space and R is a set of allowable tasks. Recall the definition of a metric space:

Definition 1.6. A metric space is an ordered pair $M = (X, d)$ where X is a set and $d : X \times X \to \mathbb{R}_{\geq 0}$ is a metric on X. Thus for every $x, y, z \in X$ it satisfies:

$$\begin{aligned} d(x, y) = 0 &\iff x = y \\ d(x, y) &= d(y, x) \\ d(x, z) &\leq d(x, y) + d(y, z) \end{aligned}$$

A well known example of an infinite metric space is the m-dimensional Euclidean space $(\mathbb{R}^m, \|\cdot\|)$ where $\|\cdot\|$ is the euclidean norm. For an example of a finite metric space consider a weighted undirected graph $G(V, E)$ with a weight function $w : E \to \mathbb{R}_{>0}$. The set of points in the metric are the vertices of the graph. Now notice that we can use as distance between two points, the shortest path between the two vertices which satisfies all the requirements in order to be a metric.

A task is a function $\tau : X \to \mathbb{R} \cup \{\infty\}$. The value $\tau(x), x \in X$ is the cost of processing the task τ in state x. Whenever the state is changed from x to y a cost of $d(x, y)$ is incurred. Hence, the goal is to minimize the total cost:

$$\sum_{i=1}^{T} \tau_i(x_i) + \sum_{i=1}^{T} d(x_{i-1}, x_i)$$

where the first sum represents the total processing (or service) cost and the second represents the total transaction (or switching) costs.

A simple example of a MTS problem is the following: Consider you have an ice cream shop where you sell two ice cream flavours: chocolate and vanilla. Your ice cream machine has two modes (C, V) for producing the two flavors. Changing between the two modes has a cost of 1. The machine produces the ice cream which corresponds at its state at low cost but you can also manually make one for a higher cost. In particular, in mode V, the cost of vanilla is 1 and the cost of chocolate is 2 while at mode C the cost of chocolate is 1 and the cost of vanilla is 2. Requests for ice cream come in an online fashion. How should you choose when to change the state of the machine in order to minimize the cost? This problem can be formulated as a MTS of two states C, V with distance of 1 and possible task functions $\tau_C = (1, 2)$ for states C, V respectively and $\tau_V = (2, 1)$ for states C, V respectively.

Chapter 2

Convex Optimization

Mathematical Optimization (or Mathematical programming) is the problem of selecting the best (optimal) element from a set S (usually $\mathbb{R}^n$) that minimizes (or maximizes) a certain function. More formally, a mathematical minimization problem can be formulated as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & g_i(x) \leq b_i,\ i = 1, \ldots, m. \end{aligned}$$

The set $X = \{x \in \mathbb{R}^n :\ g_i(x) \leq b_i,\ \forall i \in [m]\}$ is called the feasible set of solutions (or constraints) and a point $x \in X$ is called a feasible point. A point $x \notin X$ is called infeasible. For a minimization problem, a vector x^* for which $f(x^*) \leq f(x)$, $\forall x \in X$ is the global minimum of f. Now, a natural question arises: Is there always an optimal solution to such a problem ? The answer is no. The following theorem of Weirstrass states the sufficient conditions in order for an optimal solution to exist and throughout this thesis we'll only encounter problems where these conditions hold.

Theorem 2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function and $X \subset \mathbb{R}^n$ be a nonempty, bounded and closed set. Then, the optimization problem $\min f(x) :\ x \in X$ has an optimal solution*

The purpose of the field of mathematical programming is to design algorithms that find the optimal solution to such a problem efficiently. In the general case, this is extremely hard whereas in special cases like linear programming (f, g affine), an optimal solution can be found efficiently. In this chapter, we introduce the reader to a special case of mathematical programming when the objective is a convex function and the feasible set is convex named *convex optimization (or programming)*. As we'll see, these restrictions allow us to design efficient algorithms. One of the reasons the area of convex optimization has received great attention the past decades is because a very large

number of problems, from many distinct areas, can be formulated as convex programs. For instance in Machine Learning, fundamental problems like SVM, regression, logistic regression, LASSO and matrix completion can be formulated like convex programs. For an extensive analysis on theory, applications and algorithms of convex optimization we refer the reader to the excellent book of S. Boyd and L. Vandenberghe [15]. Moreover, a great survey that focuses on algorithms is [16]. The ideas we discuss in this chapter are drawn from these two books.

2.1 Basic concepts

We'll now proceed with the definitions of a convex set and a convex function.

Definition 2.2. A set $X \subset \mathbb{R}^n$ is called a **convex set** if for any two points in X it contains their line. This means for every $x, y \in X$ and any $\lambda \in [0, 1]$ we have that $\lambda x + (1 - \lambda)y \in X$.

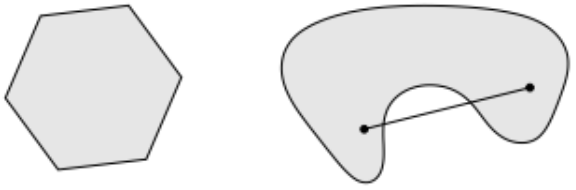


FIGURE 2.1: Examples of a convex and a non-convex set

Examples of convex sets are:

- The complete space $\mathbb{R}^n$
- The solution set of a system of linear equations $x : Ax = b$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$
- Hyperplanes $\{x : c^\top x = b\}$ and halfspaces $\{x : c^\top x \leq b\}$
- The p-norm ball ($p \geq 1$) $\{x : \|x\| \leq a\}$ for any $a \geq 0$.
- The sublevel set of a convex function f: $\{x : f(x) \leq g\}$ for some $g \in \mathbb{R}$.

An important feature of convex sets is the following:

Theorem 2.3. *The intersection of two convex sets is a convex set.*

Proof. Let A, B be convex sets. Take $x, y \in A \cap B$ and let z lie on the line segment between x, y. Then $z \in A$ since A is convex and similarly, $z \in B$ because B is convex. Therefore, $z \in A \cap B$. □

By induction, we can generalize this theorem for any finite number of convex sets. But why is this important? Because when formulating a convex optimization problem, if we desire to add one more constraint (a convex set) to the feasible set then the feasible set remains convex.

Definition 2.4. For a convex set $X \subseteq \mathbb{R}^n$, we say that a function $f : X \to \mathbb{R}^n$ is a **convex function** on X if for any two points $x, y \in X$ and any $\lambda \in [0, 1]$ we have that:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

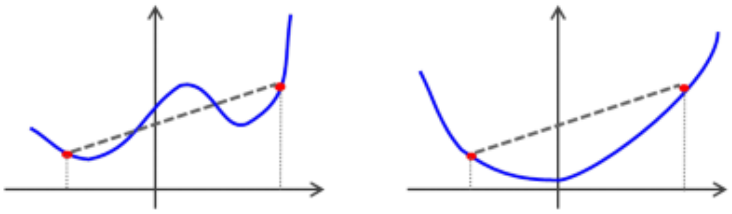


FIGURE 2.2: Examples of a non-convex and a convex function

Geometrically speaking, the line segment between $(x, f(x))$ to $(y, f(y))$ must be above the graph of f in order for f to be convex.

Examples of convex functions are:

- The affine function $f(x) = c^\top x + b$
- Any p-norm $(p \geq 1)$ $f(x) = \|x\|$
- For a positive definite matrix A, the quadratic function $f(x) = x^\top Ax + c^\top x + d$
- $\sum_{i=1}^{n} f_i(x)$ and $\max_i\{f_i(x)\}$ where $f_i(x)$ $\forall i$ are convex functions. Thus, convexity is preserved under sum and pointwise maximum.

An extremely important property that convex functions demonstrate and actually that is the main reason of algorithmic success of convex optimization is that they exhibit a local to global phenomenon. This means that if one has local information on a convex function then can arrive at conclusions about global properties of the function. This is demonstrated by the two following theorems:

Theorem 2.5. *Let $f : X \to \mathbb{R}^n$ be a differentiable function and X be a convex subset of $\mathbb{R}^n$. Then, f is convex if and only $\forall x, y \in X$ it holds:*

$$f(y) \geq f(x) + \nabla f(x)^T(y - x)$$

What does this theorem mean? It means that if we have the value of f at a point x and the respective gradient (both are local information), then we can come up with a global lower bound for f: The affine function $g(y) = f(x) + \nabla f(x)^T(y - x)$ which is also the first order approximation of f around x.

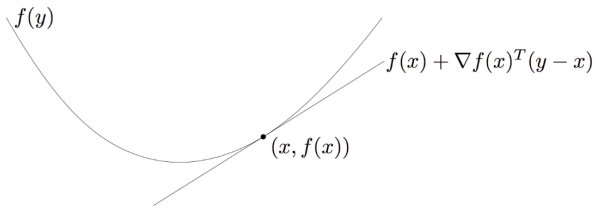


FIGURE 2.3: Illustration of theorem 2.5

Theorem 2.6. *If x is a local minimum of a convex function f, then x is a global minimum.*

Proof. Assume x isn't a global minimum of f and let y be a point for which $f(y) < f(x)$. Now consider the vector $z = \lambda x + (1 - \lambda)y$ for some $\lambda \in (0, 1)$. From the definition of convexity for f we have:

$$f(z) = f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) < \lambda f(x) + (1 - \lambda)f(x) = f(x)$$

Since z is an arbitrary convex combination of x, y, it can be as close to x as possible. Thus x isn't be a local minimum, a contradiction. □

This again means that if one has local information, has found a point x which is a local minimum is ensured that in fact is a global minimum. This does not hold in general for nonconvex optimization problems for which the usual methods converge only to local minima, and not global (for instance, the training of neural networks).

Theorem 2.7. *Consider a convex and diffentiable function f defined in a set X. Then any point $x \in X$ that satisfies $\nabla f(x) = 0$ is a global minimum of f.*

The proof comes straightforward from theorem 2.5. The reverse does not always hold (consider for example $f(x) = x^2,\ X = [1, 2]$), but it holds when $X = \mathbb{R}^n$. Hence, for unconstrained optimization of differentiable convex functions, a necessary and sufficient condition for global optimality is $\nabla f(x) = 0$. For constrained optimization, the following theorem states the necessary and sufficient condition for optimality:

Theorem 2.8. *Let $f : X \to \mathbb{R}$ be a convex function and X a closed, convex set on which f is differentiable. Then x is optimal if and only if $\nabla f(x)^T(y - x) \geq 0\ \forall y \in X$*

Proof. If x satisfies the inequality, the from theorem 2.5 we get that is indeed an optimal solution. For the other way round, consider that the inequality does not hold, thus there exists a vector y such that $\nabla f(x)^T(y-x) < 0$. Now consider $h(t) = f(x + t(y-x))$ for which $h'(0) = \nabla f(x)^T(y-x) < 0$, thus in the direction of y there exist some vector z for which $f(z) < f(x)$, a contradiction since x is optimal. □

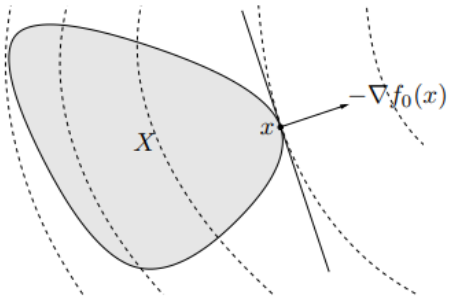


FIGURE 2.4: Illustration of theorem 2.8. Observe the inner product of $-\nabla f(x)$ and $y - x$ for some $y \in X$

2.2 Duality and Optimality Conditions

Now, we'll introduce the reader to an important concept in convex optimization (and in general, mathematical optimization): *Duality*, and in particular *Lagrange Duality*. In the algorithmic community, duality is extensively employed in order to design approximation algorithms. In particular, methods like Primal-Dual analysis or Dual fitting which mainly utilize linear programming duality can be used to design algorithm for combinatorial problems including the Set Cover, Uncapacitated Facility Location and the k-median problem. Concerning online algorithms, duality also is employed to design competitive algorithms. For instance, the $\frac{e}{e-1}$-competitive ratio for the ski-rental problem we discussed in Chapter 1 can be achieved with an approach based on duality. An interesting read on the use of duality for competitive analysis is [21]. The algorithmic success of duality is partially because through the dual problem we'll define below, it provides lower bounds for the optimization problem in hand. As we'll see in chapter 4, we'll analyze an algorithm for a linear covering problem with switching costs where the proof will be heavily based on duality and the lower bound it provides us. In addition, in this paragraph, we'll discuss the optimality conditions for an optimization problem (Karush-Kuhn-Tucker conditions) which are closely related to duality and we'll find them useful in later topics.

To understand the concept of duality first we have to start with the definition of the *Lagrangian* of an optimization problem:

Definition 2.9. Given the optimization problem, which from now on we'll refer to as the *primal* optimization problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0,\ i = 1, \ldots, m. \\ & h_j(x) = 0,\ j = 1, \ldots, p. \end{aligned}$$

The **Lagrangian** associated with the optimization problem is:

$$L(x, \lambda, v) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} v_j h_j(x)$$

The variables λ_i, $i \in [m]$ and u_j, $j \in [p]$ are called Lagrange multipliers.

Now we are ready to define the dual problem. The objective of the dual problem is the *Lagrangian Dual* defined as:

Definition 2.10. Given the Lagrangian $L(x, \lambda, u)$ of some optimization problem over domain X, the **Lagrangian Dual** is the function:

$$F(\lambda, v) = \inf_{x \in X} L(x, \lambda, u) = \inf_{x \in X} (f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} v_j h_j(x))$$

Notice that the Lagrangian dual is always a concave function (a function f is concave when $-f$ is convex). The Lagrangian dual function is the pointwise minimum of affine functions, since for every x we get an affine function in variables λ, u. It is easy to verify that the pointwise minimum of affine functions(and in general concave functions) is a concave function. The maximization of a concave function over a convex set is equivalent to a convex optimization problem since for a concave function g it holds: $arg\max_x\{g(x)\} = arg\min_x\{-g(x)\}$ (if such an x exists). Hence, even if the primal problem is not convex, the dual problem that we'll define below is always convex.

Definition 2.11. The Dual optimization problem of the primal is:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & F(\lambda, v) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned}$$

Now, we'll use the above to derive the dual of a linear program. A linear program is a problem that may have the form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax \geq b \end{aligned}$$

Where $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$. The Lagrangian function is: $L(x, \lambda) = c^T x + \lambda^T (b - Ax) = (c^T - \lambda^T A)x + \lambda^T b$. Hence, the Lagrangian dual is:

$$F(\lambda) = \inf_x L(x, \lambda) = \begin{cases} b^T \lambda, \text{ if } -A^T \lambda + c = 0 \\ -\infty, \text{ otherwise} \end{cases}$$

That's because an affine function $(a^T x, \ a \neq 0)$ is unbounded in $\mathbb{R}^n$. Thus the dual problem becomes:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & b^T \lambda \\ \text{subject to} \quad & A^T \lambda = c \\ & \lambda \geq 0 \end{aligned}$$

Which is also a linear program. As we previously said, the dual problem provides a lower bound to the optimal value of the primal problem. That is named *weak duality.*

Theorem 2.12 (Weak Duality)**.** *Let p^* be the value of the optimal solution for the primal problem, and d^* be the value of the optimal solution for the dual problem. Then it holds: $d^* \leq p^*$.*

Proof. Let X denote the feasible region of the primal problem. Now recall the definition of the primal problem. We had $g_i(x) \leq 0, \ i \in [m]$ and $h_i(x) = 0, \ j \in [p]$. Since λ_i are nonnegative, we have that:

$$f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} u_i h_i(x) \leq f(x), \ \forall x \in X$$

Since this holds for all λ, u, it holds also for the optimal values λ^*, u^*. Thus:

$$d^* = \inf_{x \in X} L(x, \lambda^*, u^*) \leq \inf_{x \in X} f(x) \leq p^*$$

□

A natural question arises: When does $p^* = d^*$ (which we refer to as strong duality), the optimal value of the primal solution is equal to the optimal value of the dual solution? This is determined by the *Slater's Condition* which is a sufficient condition to ensure strong duality. Slater's condition always holds for linear programming, except when both primal and dual problems are unfeasible (i.e $p^* = +\infty$ and $d^* = -\infty$). In general, this is not true for an optimization problem (even a convex one).

Definition 2.13 (Slater's condition)**.** For a primal optimization problem, we say that it respects Slater's condition if the objective function f is convex, the constraint functions

$g_i,\ i \in [m]$ are convex, the constraint functions $h_j, j \in [p]$ are affine, and there exists a point $\bar{x}$ in the interior of the region, i.e $g_i(\bar{x}) < 0,\ \forall i \in [m]$, and $h_j(\bar{x}) = 0,\ \forall j \in [p]$

A careful reader should have observed by now that the equality constraints h_i should always be affine functions in order to have a convex problem. Otherwise the constraint $h(x) = 0$ is equivalent to $h(x) \geq 0 \wedge h(x) \leq 0$. If h is not an affine function, at least one these constraints is described by a nonconvex set.

Now we'll state the optimality conditions (Karush-Kuhn-Tucker) we mentioned earlier, which we'll find extremely useful is later chapters:

Definition 2.14. The Karush-Kuhn-Tucker conditions for the pair of the primal and dual problem are:

$$\begin{aligned} 0 = \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x) + \sum_{j=1}^{p} u_j \nabla h_j(x) \\ u_i h_i(x) = 0,\ \forall i \in [m] \\ g_i(x) \leq 0,\ \forall i \in [m],\ h_j(x) = 0,\ \forall j \in [p] \\ \lambda_i \geq 0,\ \forall i \in [m] \end{aligned}$$

Which are, in order, the stationarity condition, complementary slackness conditions, primal feasibility and dual feasibility. We should note that for any optimization problem, these conditions are sufficient for optimality for the primal and the dual problem. If x^*, λ^*, u^* satisfy the KKT conditions then these are in fact the optimal solutions for the primal-dual pair. The necessity does not always hold. Only if strong duality holds, then the optimal solutions satisfy the KKT conditions. Thus, under strong duality, KKT conditions are necessary and sufficient for optimality of the primal-dual pair.

2.3 Gradient Descent

We'll now proceed and describe the most simple algorithm for convex optimization (and in general, optimization). Gradient descent is a first-order algorithm which means that uses only gradient information in order to find an optimal solution. In contrast, second-order methods (like Newton's Method) exploit the curvature of the objective function by using the Hessian (the square matrix of second-order derivatives of a scalar-valued function). Moreover, Gradient descent is an iterative method meaning that the optimization procedure proceeds in iterations, each one improving the objective value. The rule for updating is the following:

$$x_t = x_{t-1} - \eta_t \nabla f(x_{t-1})$$

Where η is the step size (or learning rate) at time t which is chosen based properties of the objective function and perhaps on which is iteration the algorithm is on. The idea is to take a step from the previous point designated by the negative gradient of the function in that point. In that way, we make a step towards a direction that the function has a smaller value. A question at this point is how to evaluate such an algorithm? Usually, an input to the gradient descent algorithm is a desired accuracy of ϵ in the solution it provides. Suppose x^* is the value that minimizes the objective and x_t is the algorithm's value in iteration t. Then we'd like to minimize the number of iterations (t) in order to have (or having a high convergence ratio):

$$f(x_t) - f(x^*) \leq \epsilon$$

An algorithm is evaluated based on the number of iterations needed in order to provide an accuracy of ϵ. $T = O(g(\epsilon))$ (ignoring other constants). We'll see shortly that if the objective function has some useful properties the convergence ratio is significantly improved. Now let's state the algorithm more formally for the case of constrained optimization:

Algorithm 1: Gradient Descent

1: **Input:** function f, number of iterations T, Feasible set X, initial point $x_1 \in X$, sequence of step sizes $\{\eta_t\}$
2: **for** $t = 1 : T$ do
3: Let $y_{t+1} = x_t - \eta_t \nabla f(x_t),\ x_{t+1} = \Pi_X(y_{t+1})$
4: **end for**
5: **return** x_{T+1}

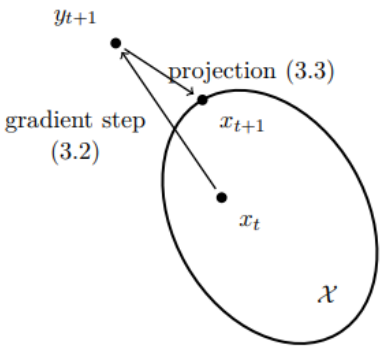


FIGURE 2.5: Illustration of the Gradient Descent algorithm

Since the point y_{t+1} may lie outside of the feasible set we need to project back to the feasible set. The step $x_{t+1} = \Pi_X(y_{t+1})$ does exactly that. The algorithms we'll discuss throughout this thesis will contain as a step a projection onto a convex set. First, we

have to say that the projection of a point x onto a convex set X is given by:

$$\Pi_X(y) = arg\min_{x \in X} \|x - y\|$$

and it can be shown that the above problem has a unique solution. Using theorem 2.8 we get that for $x \in X$ and $y \in \mathbb{R}^n$.

$$(\Pi_X(y) - x)^T(\Pi_X(y) - y) \leq 0$$

and by using law of cosines we get:

$$\|\Pi_X(y) - x\|^2 + \|y - \Pi_X(y)\|^2 \leq \|y - x\|^2$$

Which implies,

$$\|\Pi_X(y) - x\| \leq \|y - x\|$$

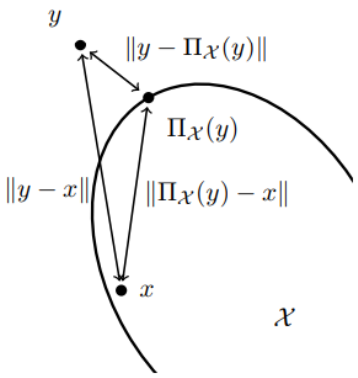


FIGURE 2.6: projection onto a convex set

Now, let's discuss some properties of the objective function and the constraint set that are useful in the analysis of Gradient Descent. Also, we'll find them extremely useful in later chapters.

Definition 2.15. The diameter of a convex set X is given by:

$$max_{x,y \in X} \|x - y\|$$

In general, the diameter will affect negatively the convergence ratio, since for largest sets, it will take longer to find the minimizer.

Definition 2.16. We say that a function is Lipschitz continuous with parameter G if:

$$|f(x) - f(y)| \leq G\|x - y\|, \quad \forall x, y \in X$$

which equivalent to a bounded gradient $\|\nabla f(x)\| \leq G$. Low values of G means that the function isn't changing at a high rate. Thus the values of points near to the minimizer will be closer to the optimal value and the convergence ratio will be higher.

Definition 2.17. We say that a function is α**-strongly convex** if:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\alpha}{2} \|y - x\|^2, \quad \forall x, y \in X$$

which means that the function f grows at least quadratically. Similarly,

Definition 2.18. We say that a function is β**-smooth** if:

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{\beta}{2} \|y - x\|^2, \quad \forall x, y \in X$$

which means that the function f grows at most quadratically. This condition is equivalent to a Lipschitz condition of the gradient:

$$\|\nabla f(x) - \nabla f(y)\| \leq \|x - y\|, \quad \forall x, y \in X$$

Finally, we'll define the *condition number* of a function f which is a decisive paramenter in the convergence ratio of gradient descent: First we have to say that for an α-strongly convex and β- smooth twice differentiable function the follow holds:

$$\alpha I \preceq \nabla^2 f(x) \preceq \beta I$$

where we denote $A \preceq B$ if $B - A$ is a positive semidefinite matrix. Moreover, this is equivalent to all the eigenvalues of the hessian matrix to lie in the interval $[\alpha, \beta]$. The condition number of f is now:

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

The condition number shows how "spherical" are the sublevel sets of the convex function. If $\gamma = 1$ we get spherical subsets and thus at any point, the gradient will point towards the minimizer and thus the convergence ratio of gradient descent is higher. This does not happen for small values of γ where the sublevel sets are ellipsoidal and the gradient points to a location far from the minimizer.

First, let's see how gradient descent performs for a general case:

Theorem 2.19. *For G-Lipschitz convex functions and diameter of the feasible set equal to D, it holds:*

$$f(\frac{1}{T} \sum_{t=1}^{T} x_t) - f(x^*) \leq \frac{DG}{\sqrt{T}}$$

There is no need to provide the proof for this theorem. We'll derive it as a special case of Online Gradient Descent in the next chapter. Thus one needs $O(\frac{1}{\epsilon^2})$ iterations in order to achieve an error of ϵ. We'll see now that the number of iterations is exponentially lower in the case we have a γ-well conditioned function.

Theorem 2.20. *For constrained minimization of γ-well conditioned functions and $\eta_t = \frac{1}{\beta}$, it holds:*

$$f(x_{t+1}) - f(x^*) \leq ((f(x_1) - f(x^*))e^{-\frac{\gamma}{4}t}$$

Proof. By strong convexity, we have for any pair $x, x_t \in X$:

$$\nabla f(x_t)^T(x - x_t) \leq f(x) - f(x_t) - \frac{\alpha}{2}\|x - x_t\|^2$$

Because $\|x\|^2 = x^T x$, observe that:

$$\begin{aligned} x_{t+1} = \Pi_X(x_t - \eta_t \nabla f(x_t)) = arg\min_{x \in X}\{\|x - (x_t - \eta_t \nabla f(x_t))\|^2\} = \\ arg\min_{x \in X}\{\|x - x_t\|^2 + 2\eta_t(x - x_t)^T \nabla f(x_t) + \eta_t^2 \nabla^2 f(x_t)\} = \\ arg\min_{x \in X}\{\nabla f(x_t)^T(x - x_t) + \frac{\beta}{2}\|x - x_t\|^2\} \end{aligned}$$

Hence, by smoothness of f, we have:

$$\begin{aligned} f(x_{t+1}) - f(x_t) \leq \nabla f(x_{t+1})^T(x - x_t) + \frac{\beta}{2}\|x_{t+1} - x_t\|^2 = \\ = \min_{x \in X}\{\nabla f(x)^T(x - x_t) + \frac{\beta}{2}\|x - x_t\|^2\} \end{aligned}$$

Let $h_t = f(x_t) - f(x^*)$, the error at time t. By using strong convexity we get:

$$h_{t+1} - h_t \leq \min_{x \in X}\{f(x) - f(x_t) + \frac{\beta - \alpha}{2}\|x - x_t\|^2\}$$

Let's take a look intuitively what we've achieved at this point just by using the powerful assumptions of strong convexity and smoothness. For α close to β we get easily that $h_{t+1} \leq 0$, and thus convergence is achieved. This means that for $\gamma = 1$ even 1 iteration suffices for optimality (Now the reader can also understand why we chose this specific learning rate). Observe now that for γ close to one the minimization leads to a point that is close to x^*. That agrees with the intuitively explanation we gave for γ above. As we previously said, f is bounded above and below from quadratic functions. Closer the quadratic functions are to each other (a.k.a closer a is to b), the above minimum will provide us as close as to $-h_t$, which is our aim for fast convergence.

Obviously, by minimizing over a subset of X, the minimum cannot get any bigger. Let's consider as a subset, the set of convex combinations of x_t and x^* which is a line.

Intuitively, the minimizer should be a point close to the set of convex combinations of x^* and x_t since those two points minimize each of the functions in the sum (In one dimension, the minimizer will always lie in this subset). Even though with this relaxation the analysis will not be optimal, it suffices to show that an exponential convergence ratio is achieved. Therefore for any $\lambda \in [0, 1]$

$$\begin{aligned} h_{t+1} - h_t \leq \min_{[x_t, x^*]} \{f(x) - f(x_t) + \frac{\beta - \alpha}{2} \|x - x_t\|^2\} \leq \\ f((1-\lambda)x_t + \lambda x^*) - f(x_t) + \frac{\beta - \alpha}{2} \lambda^2 \|x - x_t\|^2 \leq \\ \leq -\lambda h_t + \frac{\beta - \alpha}{2} \lambda^2 \|x - x_t\|^2 \end{aligned}$$

Now, we have by strong convexity and Theorem 2.8:

$$h_t = f(x_t) - f(x^*) \geq \nabla f(x^*)^T (x_t - x^*) + \frac{\alpha}{2} \|x^* - x_t\|^2 \geq \frac{\alpha}{2} \|x^* - x_t\|^2$$

Finally, we get:

$$h_{t+1} - h_t \leq (-\lambda + \frac{\beta - \alpha}{\alpha} \lambda^2) h_t$$

We ask for a value of k such that $-\lambda^2 \frac{\beta - \alpha}{a} + \lambda + k > 0, \forall \lambda$. We easily choose $k = -\frac{\alpha}{4(\beta - \alpha)}$. Thus,

$$h_{t+1} \leq h_t (1 - \frac{\alpha}{4(\beta - \alpha)}) \leq h_t (1 - \frac{\alpha}{4\beta}) \leq h_1 (1 - \frac{\alpha}{4\beta})^t \leq h_1 e^{-\frac{\gamma}{4} t}$$

And the proof is complete. □

Thus the γ-well condition on f improves substantially the number of iterations to $O(\log(\frac{1}{\epsilon}))$. A careful reader at this point has observed that even though the projected gradient descent is an algorithm for optimization, it contains a step of projection onto a convex set, which in the general case is an optimization problem!. We didn't built from scratch an algorithm that solves an optimization problem. An interested reader can see [16] for non black-box optimization algorithms like the Ellipsoid Method or Interior Point Methods. We discussed gradient descent to see how the algorithm manages to solve an optimization problem and in order to introduce the reader more gently to online gradient descent in the next chapter where its natural to assume that the designer has a non black-box optimization solver at her disposition. However, we should note that there are specific cases, like the euclidean ball, where the calculation is straightforward as well as other sets such as the probability simplex which we see in many applications in online learning: $\{x \in \mathbb{R}^n : x \geq 0, \sum_{i=1}^n x_i = 1\}$ where the projection can be calculated analytically in time complexity $O(n \log n)$ without the use of an optimization procedure.

Chapter 3

Online learning & Online Convex Optimization

In the previous chapter, we discussed convex optimization in an *offline* setting. Now, we'll discuss in an *online* setting which defines the area of Online Convex Optimization (OCO). OCO is strongly related to the area of online learning as well as the main problem of this thesis, OCO with switching cost. In fact, many online learning problems can be formulated as OCO problems. We start by introducing the reader to the area of online learning through the experts problem, one of the fundamental problems of this area. For an extensive survey on online learning and online convex optimization we refer the reader to [22],[28],[41]. This Chapter is mostly based on the first Chapters of [28].

3.1 The Experts problem

Let's consider the following online scenario: Each day for T days you you receive a certain question which has two possible answers(A and B), without knowing the questions of the future days. However, you have to give your answer before the question if revealed. After your answer in each day, if you chose the right answer then you don't get penalized. Otherwise you suffer a loss. Fortunately, to help you make your decisions there is a team of N *experts* who each of them recommends one of these answers. The question is how to choose a good strategy for such a problem. To be more precise, how to exploit the information the experts give in each round. This is a typical online decision-making scenario where there are external sources that help the player to make choices. For instance, the question may be the choice of buying or not a certain stock and the experts to be a team of brokers.

Recall now the notion of the adversary from chapter 1: The adversary will construct the worst possible input sequence for the player including the decisions of the experts

as well as which of the questions will be asked in each round. At first, it may seem that the adversary has too much power and the player has no possible good strategy, but as we'll see shortly if we define an appropriate and fair benchmark we can design efficient online algorithms. But let's start with a benchmark that it's unfair to the player which is to minimize the number of mistakes the player does (relatively to T). Actually, there is a trivial strategy to obtain $\frac{T}{2}$ mistakes which is also the optimal for any randomized algorithm.

To obtain $\frac{T}{2}$ mistakes, we simply choose one of these choices based on a fair coin flip. Recall that the adversary we consider does not know the random number generator the player uses. Such a benchmark is not fair in a sense that it doesn't take into consideration how poignant is the advice of the experts. That's because, in two different instances, if in the first, the experts make random predictions based on a coin flip and in another the experts are right almost every time it doesn't make sense to evaluate both players based on the number of their respective mistakes.

A meaningful approach is to compete against the best expert, the one that makes the fewest mistakes. Let's denote as $m_t(i)$ a binary function which indicates if the expert i makes a wrong prediction in day t and as m_t similarly if the player makes a mistake at day t. Our goal is to minimize the average between the mistakes the player makes and the number of mistakes of the best expert. That's equivalent to minimizing the average *Regret* of the player which we'll constantly use as a performance metric throughout this chapter. Intuitively, it means that at the end, looking back at the experts predictions, you don't want to regret not having picked to follow one of the experts all the time. The purpose of the algorithms in online learning is to find as fast as possible such an expert.

$$\frac{1}{T}(\sum_{t=1}^{T} m_t - \min_{i \in N} \sum_{t=1}^{T} m_t(i))$$

Now we'll discuss an efficient algorithm in order to minimize the average Regret, named the weighted majority algorithm.

3.1.1 The Weighted Majority Algorithm

Before stating the algorithm in a general setting, let's consider a simple case in order to gain intuition on the problem. Suppose there is an expert that makes no mistakes. Your strategy is simple: Take a majority vote, which means to make the choice the majority of the experts predict. Of course, after you realize an expert made a mistake, you don't take his opinion in consideration anymore. Based on this strategy, how many mistakes will you do you find the unerring one? Notice, that each time the player makes a mistake, half of the experts are discarded because otherwise the player would be right.

Therefore, the player makes $O(\log n)$ mistakes until she finds the best expert and the average regret goes to zero as $T \to \infty$.

Now, in the general case, there won't be an unerring expert and thus the above algorithm won't work. In the above case, the trust of the player to an expert is binary. Either she listens to her or not. Let's generalize this concept and make the trust (or weight) to be a function w_t in the interval $[0, 1]$. The idea stays the same: sum up the trust of the experts that prediction an action and take the action which has the highest trust. When an expert makes a mistake now, the player will reduce her trust by a factor of $1 - \epsilon$. Let's describe the algorithm formally. Let the two choices be A, B and $S_t(A)(S_t(B))$ the set of the experts whose opinion is $A(B)$ in round t.

Algorithm 1: Deterministic Weighted Majority Algorithm

```
Initialize: w_1(i) = 1, ∀i ∈ [n]
for t = 1 : T do
    if Σ_{i∈S_t(A)} w_t(i) ≥ Σ_{i∈S_t(B)} w_t(i) then
        choose A, otherwise choose B
    end if
    for expert i which made a mistake in round t
        w_{t+1}(i) = (1 − ε) w_t(i)
    end for
end for
```

Theorem 3.1. *Let M_T and $M_T(i)$ be the total number of mistakes the algorithm and the i expert make until step T, respectively. Then, for any $i \in [n]$:*

$$M_T < 2(1+\epsilon)M_T(i) + 2\frac{\log n}{\epsilon}$$

Proof. We'll analyze the algorithm using a potential $\phi_t = \sum_{i=1}^{n} w_t(i)$ which simply sums up the total weight of all the experts in round t. Since the weights can only be decreased it holds: $\phi_t \leq \phi_{t+1}$. At every round t that we made a mistake (say we chose A and B appeared), at least half of the weight corresponded to experts that chose A and is equal to $\sum_{i \in S_t(A)} w_t(i) \geq \frac{\phi_t}{2}$. Hence,

$$\phi_{t+1} \leq \sum_{i \in S_t(A)} (1-\epsilon) w_t(i) + \sum_{i \in S_t(b)} w_t(i) \leq \frac{1}{2}\phi_t(i)(1-\epsilon) + \frac{1}{2}\phi_t(i) = (1 - \frac{\epsilon}{2})\phi_t$$

By induction we get:

$$\phi_T = \phi_1(1 - \frac{\epsilon}{2})^{M_T} = n(1 - \frac{\epsilon}{2})^{M_T}$$

Moreover, for every expert i it holds (recall that the weight $w_t(i)$ is decreased by a factor of $(1-\epsilon)$ every time the expert i makes a mistake):

$$\phi_T = \sum_{i=1}^{n} w_T(i) > w_T(i) = (1-\epsilon)^{M_T(i)}$$

The above two inequalities give us:

$$(1-\epsilon)^{M_T(i)} < n(1-\frac{\epsilon}{2})^{M_T} \Rightarrow M_T(i)\log(1-\epsilon) < \log n + log(1-\frac{\epsilon}{2})M_T$$

Finally, applying the inequalities $-x-x^2 < log(1-x) < -x$ for $x \in (0, \frac{1}{2})$ which derive from the Taylor expansion of $log(1-x)$ we get:

$$-M_T(i)(\epsilon+\epsilon^2) \leq \log n - M_T\frac{\epsilon}{2} \Rightarrow M_T < 2(1+\epsilon)M_T(i) + 2\frac{\log n}{\epsilon}$$

□

This result implies that when $T >> n$, the number of mistakes the algorithm makes is approximately twice the number of mistakes of the best expert. Actually, if $L \leq \frac{T}{2}$, no algorithm can improve substantially these results.

Theorem 3.2. *Suppose the number of mistakes the best expert makes is $L \leq \frac{T}{2}$. Then, in the worst case, no deterministic algorithm can make fewer than $2L$ mistakes.*

Proof. Assume an instance with two experts: A, B which always predict A, B respectively. For any given decision of the algorithm, the adversary chooses the opposite decision (that's possible, since the algorithm is deterministic). Hence, the total number of mistakes the algorithm makes is T. The best of the two experts makes $\leq \frac{T}{2}$ mistakes since in each round at least one of them is right. □

Now, a natural questions arises: Can randomization be proven useful in such a setting? The answer is yes. We'll now discuss a randomized version of the WM algorithm. The only difference from the deterministic is instead of adding the weights of experts when comparing the sums in order to make our decision, we now choose to follow one expert basec on the weight. The probability of choosing expert i in round t is now: $p_t(i) = \frac{w_t(i)}{\sum_{j=1}^{n} w_t(j)}$. The algorithm is described below:

Algorithm 2: Randomized Weighted Majority Algorithm

Initialize: $w_1(i) = 1,\ \forall i \in [n]$
for $t = 1 : T$ do
    Select advice of expert i with probability $p_t(i) = \frac{w_t(i)}{\sum_{j=1}^{n} w_t(j)}$
    for for each expert i that made a mistake in round t do:
        $w_{t+1}(i) = (1 - \epsilon)w_t(i)$
    end for
end for

Theorem 3.3. *Let $m_T, m_T(i)$ denote the total number of total mistakes of the player and the expert i. Then for every expert i:*

$$\mathbb{E}[m_t] < (1 + \epsilon)m_T(i) + \frac{\log n}{\epsilon}$$

Thus randomization improves the bound by a factor of two. The proof of the above theorem shares the same idea with the one of the deterministic case and we won't include it at this point.

So far, we've assumed that the loss of the experts is binary. Either expert i makes a mistake or not. Let's generalize this concept and assume that the loss of the experts is a number $c \in [0, 1]$. Now consider the same strategy as the above randomized algorithm. Choosing the expert i with probability $p_t(i)$. Then, the expected loss of the algorithm in round t is $\sum_{i=1}^{n} c_t(i)p_t(i) = c_t^T p_t$. This leads us to the *hedge* algorithm which we analyze below:

Algorithm 3: Hedge Algorithm

Initialize: $w_1(i) = 1,\ \forall i \in [n]$
for $t = 1 : T$ do
    Select advice of expert i with probability $x_t(i) = \frac{w_t(i)}{\sum_{j=1}^{n} w_t(j)}$
    Suffer loss $c_t^T x_t$
    Update weights $w_{t+1}(i) = w_t(i)e^{-\epsilon c_t(i)}$
end for

Theorem 3.4. *Let c_t^2 denote the n-dimensional vector of pointwise square losses (i.e $c^2(i) = c(i)^2$). Then, the hedge algorithm satisfies for every expert i:*

$$\sum_{t=1}^{T} c_t^T x_t \leq \sum_{t=1}^{T} c_t(i) + \epsilon \sum_{t=1}^{T} c_t^2 x_t + \frac{\log n}{\epsilon}$$

Proof. Set $\Phi_t = \sum_{t=1}^{T} w_t(i)$, then:

$$\Phi_{t+1} = \sum_{i=1}^{n} w_t(i) e^{-\epsilon c_t(i)} = \Phi_t \sum_{i=1}^{n} x_t e^{-\epsilon c_t(i)}$$

Using the taylor approximation of e^{-x}: $\forall x \geq 0,\ e^{-x} \leq 1 - x + x^2$:

$$\Phi_{t+1} \leq \Phi_t \sum_{i=1}^{n} x_t(1 - \epsilon c_t(i) + \epsilon^2 c_t(i)) = \Phi_t(1 - \epsilon c_t^T x_t + \epsilon^2 (c_t^2)^T x_t)$$

Finally, by using $1 + x \leq e^x$

$$\Phi_{t+1} \leq \Phi_t e^{-\epsilon c_t^T x_t + \epsilon^2 (c_t^2)^T x_t}$$

Given that $w_t(i)$ is less than Φ_t we get that:

$$w_t(i) \leq \Phi_t \leq n e^{-\epsilon c_t^T x_t + \epsilon^2 (c_t^2)^T x_t}$$

Now, observe that:

$$w_t(i) = e^{-\epsilon \sum_{t=1}^{T} c_t(i)}$$

By taking logarithm of both sides:

$$-\epsilon \sum_{i=1}^{T} c_t(i) \leq \log n - \epsilon \sum_{i=1}^{T} c_t^T x_t + \epsilon^2 (c_t^2)^T x_t$$

and the theorem follows. □

Now we have to choose an appropriate value of ϵ in order to minimize the regret.

Theorem 3.5. *The hedge algorithm for $\epsilon = \sqrt{\frac{\log n}{T}}$ has the following regret:*

$$\sum_{i=1}^{T} c_t^T x_t - \min_{j \in [n]} \sum_{j=1}^{T} c_t(j) \leq 2\sqrt{T \log n}$$

Proof. First observe that $c_t^2 \leq 1$ and thus $(c_t^2)^T x_t \leq 1$. Hence, using the result of the previous theorem for the best expert in hindsight i^*:

$$\sum_{i=1}^{T} c_t^T x_t - \sum_{t=1}^{T} c_t(i^*) \leq T\epsilon + \frac{\log n}{\epsilon} = 2\sqrt{T \log n}$$

□

In the above scenario, in each round, the player receives a function $f_t(x_t) = c_t^T x_t$. She has to choose a point from the probability simplex $\{x \in \mathbb{R}^n : x \geq 0, \sum_{i=1}^{n} x_i = 1\}$ in order

to minimize the regret. What about if the player received an arbitrary convex function $f_t(x)$ and has to choose a point from an arbitrary convex set X ? What algorithm should she follow then? The area of online convex optimization aims at answering this question.

3.2 A Unifying model

We now proceed and define more formally an online convex optimization problem which generalizes the previous expert setting. The model is very general and can be used to model problems from a broad range of areas. The OCO model can be expressed by the following elements:

- At each round $t = 1, 2, ..., T$, the decision maker makes a decision, choosing a point from a convex set $x_t \in X \subseteq \mathbb{R}^n$
- After the decision, the adversary reveals a convex function $f_t(x),\ t \in [T]$ and the player suffers a loss of $f_t(x_t)$.
- The goal of the decision maker is to minimize the regret. That is, in every round, to choose a strategy x_t in order to minimize:

$$\text{Regret} = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in X} \sum_{t=1}^{T} f_t(x)$$

An algorithm that guarantees a sublinear in T ($o(T)$) regret, is called a no-regret algorithm. OCO algorithms aim at finding the best fixed decision in hindsight x^* which minimizes the sum of the functions $f_t,\ t \in [T]$. The model accepts also the following interpretation: Initially, the adversary chooses a function f and breaks it in parts f_t so that the follow holds: $\sum_{t=1}^{T} f_t(x) = f(x)$. It breaks it in the worst way possible in order for the player to make the most iterations (rounds) in order to converge to the best decision in hindsight.

Apart from the experts problem, there are other problems that can be modeled as OCO problems.

- **Online Spam Detection**: In online spam detection, we observe emails $w_1, w_2..., w_T$ and need to classify them as spam / not spam at every period $t \in [T]$. We can model each email w_t as a bag of words over dictionary of size d: there are d words in our language and each email is a vector over $\{0, 1\}^n$ s.t. each index j receives a value of 1 if the word $j \in [d]$ is in the email and 0 otherwise. At every step t, our goal is to create a classifier which is a vector $x_t \in \mathbb{R}^d$ s.t. $x_t^T w_t \geq 0$ if the email

is not spam and $x_t^T w_t$ if it is spam. At every stage, after classifying the email we observe whether our classifier x_t made a mistake. One natural cost function is to assign cost 1 on each iteration we make a mistake. Another natural notion is to have a cost function that measures the square loss: $(\hat{y} - y))^2$ where $\hat{y}$ is the prediction made with our classifier at time step t (1 if not spam, -1 if spam) and y is the true label.

- **Online Recommendation Systems**: Recommendation Systems are often modeled as matrix completion problems. We assume we have some sparse 0,1 matrix, where the rows are the people the columns are media items. For example, for a service like Netflix, the entry M_{ij} takes value 1 if person i enjoyed movie j. In the online setting, at each iteration, a matrix $M_t \in \{0,1\}^{n \times m}$ is revealed, and the adversary chooses a user/movie pair together with the real preference. The goal is to use existing matrix and make an educated guess that minimizes the square loss.

3.2.1 Online Gradient Descent

Although the OCO problem may seem general and hard in its nature, there exist a very simple algorithm that guarantees the best regret possible! The algorithm is Online Gradient Descent,a straightforward generalization of the offline version we discussed in the previous chapter. It appeared in [45] which laid the foundations for the area of Online Convex Optimization. Let's state the algorithm formally:

Algorithm 4: Online Gradient Descent

1: **Input:** number of iterations T, Feasible set X, initial point $x_1 \in X$, sequence of step sizes $\{\eta_t\}$
2: **for** $t = 1 : T$ do
3: Choose x_t and suffer loss $f_t(x_t)$
4: Perform gradient step: $y_{t+1} = x_t - \eta_t \nabla f(x_t)$
5: Project onto X: $x_{t+1} = \Pi_X(y_{t+1})$
6: **end for**

The algorithm may seem strange at first sight. In round t we make a gradient step based on the gradient of f_t at point x_t in order to suffer a lower loss at the next function f_{t+1}. But the functions f_t and f_{t+1} may be completely different! We try to minimize a function f_{t+1} based on the gradient of f_t which seems entirely irrational. The gradient step may make things even worse and we suffer an even bigger loss compared to the case we wouldn't move at all. However, it's not irrational, it makes perfect sense. Remember that we don't try to converge to the minimizer of the function f_t but at a point x^* which

minimizes the sum of the functions. The point x^* is correlated with the minimizers of each function separately and intuitively, we'd expect to lie somewhere between the minimizers (for instance consider $f_1(x) = (x-1)^2$ and $f_2(x) = (x-2)^2$, making a step based on the gradient of f_1 gets us closer to $x = \frac{3}{2}$). Therefore when we make the gradient step we get closer to the point x^* and as time progresses we converge to the point x^*. When we'are closer to the point, the regret is increasing at a low rate, which is the whole point! In fact gradient descent, is a no-regret algorithm and achieves a regret of $O(\sqrt{T})$. As time progresses, the increase of regret in each step decreases.

Theorem 3.6. *Assume that at every round, the step size is $\eta_t = \frac{D}{G\sqrt{T}}$ where D is the diameter of the convex set and G is a bound on the gradient $\|\nabla f(x)\| \leq G,\ \forall x \in X$.*

$$\sum_{t=1}^{T} f_t(x_t) - \min_{x \in X} \sum_{t=1}^{T} f_t(x) \leq \frac{3}{2} GD\sqrt{T}$$

Proof. Let $x^* = arg\min_{x \in X} \sum_{t=1}^{T} f_t(x)$. We start by bounding the regret at time t using the convexity of function f_t which as we discussed is found useful when we'd like to bound the values between two points of the functions.

$$f_t(x_t) - f_t(x^*) \leq \nabla f_t(x_t)^T (x_t - x^*)$$

At this point, if we'd aimed for a regret of DGT, our job would be over. Because of the Cauchy-Shwartz inequality $f_t(x_t) - f_t(x^*) \leq GD$. Summing up we'd get the result. But as we intuitively described in the above paragraph we'd expect of the algorithm to converge to x^*, and thus $\|x_t - x^*\|$ is expected to be decreased as T grows. For this reason, we apply the gradient descent rule using the projection:

$$\|x_{t+1} - x^*\|^2 = \|\Pi_{x \in X}(x_t - \eta_t \nabla f(x_t)) - x^*\|^2 \leq \|x_t - \eta_t \nabla f(x_t) - x^*\|^2$$

From the law of the parallelogram and the bound on the gradient of f we get that:

$$\begin{aligned} \|x_t - \eta_t \nabla f(x_t) - x^*\|^2 &= \|x_t - x^*\|^2 + \eta_t^2 \|\nabla f(x_t)\| - 2\eta_t \nabla f(x_t)^T (x_t - x^*) \\ &\leq \|x_t - x^*\|^2 + \eta_t^2 G^2 - 2\eta_t \nabla f(x_t)(x_t - x^*) \end{aligned}$$

Putting the above together, we get:

$$\begin{aligned} \|x_{t+1} - x^*\|^2 \leq \|x_t - x^*\|^2 + \eta_t^2 G^2 - 2\eta_t \nabla f(x_t)^T (x_t - x^*) \Rightarrow \\ 2\nabla f_t(x_t)^T (x_t - x^*) \leq \frac{\|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2}{\eta_t} + \eta_t G^2 \end{aligned}$$

At this point, recall the gradient step $\eta = \frac{D}{G\sqrt{t}}$. For such a value of η_t we can easily see that the above inequality bounds the total regret by the factor of $DG\sqrt{T}$ by simple

calculations. Summing up over all the rounds and applying the above inequalities we get:

$$\begin{aligned}
2\sum_{t=1}^{T}(f_t(x_t) - f_t(x^*)) &\\
\leq 2\sum_{t=1}^{T}\nabla f_t(x_t)^T(x_t - x^*) &\\
\leq \sum_{t=1}^{T}(\frac{\|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2}{\eta_t} + \eta_t G^2) &\\
= \sum_{t=1}^{T}(\frac{\|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2}{\eta_t}) + G^2\sum_{t=1}^{T}\eta_t &\\
\leq \sum_{t=1}^{T}(\|x_t - x^*\|^2(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}})) + G^2\sum_{t=1}^{T}\eta_t &\\
\leq D^2\sum_{t=1}^{T}(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}) + G^2\sum_{t=1}^{T}\eta_t &\\
\leq D^2\frac{1}{\eta_T} + G^2\sum_{t=1}^{T}\eta_t &\\
\leq 3DG\sqrt{T} &
\end{aligned}$$

In the last inequality we used the fact that: $\sum_{t=1}^{T}\frac{1}{\sqrt{t}} \leq 2\sqrt{T}$ □

Notice that, in the case that all the functions f_t are the same function f we get Theorem 2.19:

$$f(\frac{1}{T}\sum_{t=1}^{T}x_t) - f(x^*) \leq \frac{1}{T}(\sum_{t=1}^{T}f(x_t) - \sum_{t=1}^{T}f(x^*)) \leq \frac{DG}{\sqrt{T}}$$

where the first inequality is due to the convexity of f.

In the case that functions f_t is general convex functions and they don't have any particular property like strong convexity or smoothness, the above regret is the best that any algorithm can attain for an online convex optimization problem. For instance, having functions that are α-strongly convex, online gradient descent can attain a regret of $\frac{G^2}{2\alpha}(1 + \log T)$ which improves upon the $O(\sqrt{T})$. For more details, see [28], Theorem 3.3.

Theorem 3.7. *The bound $\Omega(DG\sqrt{T})$ is a tight bound for any algorithm for online convex optimization.*

Proof. We consider the following setting: The feasible set of solutions is the hypercube K with vertices $x = \{\pm 1\}^n$. The function f_t in each round comes from a family of

functions $f_u(x) = u^T x$ where u is a random vector and each of its coordinates is 1 with probability $\frac{1}{2}$ and -1 otherwise. Thus, there are 2^n possible linear cost functions. Now observe that the diameter of the feasible set:

$$D \leq \sqrt{\sum_{i=1}^{n} 2^2} = 2\sqrt{n}$$

And for the bound of the gradient of the functions it holds:

$$G \leq \sqrt{\sum_{i=1}^{n} 1^2} = \sqrt{n}$$

Thus $n = DG$. Now notice that it doesn't matter what choice the player makes in each round; Her expected loss in every round will always be zero since $\mathbb{E}_u[f_u(x)] = 0$. For the adversary we'll show that there exist an instance for which his cost is less that $-cn\sqrt{T}$ for some constant c.

$$\mathbb{E}[\min_{x \in K} \sum_{t=1}^{T} f_t(x)] = \mathbb{E}[\min_{x \in K} \sum_{i=1}^{n} \sum_{t=1}^{T} v_t(i)x(i)] = n\mathbb{E}[-|\sum_{t=1}^{T} v_t(1)|]$$

Although we won't get into details we can prove that $\mathbb{E}[|\sum_{t=1}^{T} v_t(1)|] = \Omega(\sqrt{T})$ and thus the expected loss of the adversary is bounded above by $-cn\sqrt{T}$. By the expectation argument, there will be an instance for which the adversary will attain a loss smaller than $-cn\sqrt{T}$. That implies a lower bound of $\Omega(DG\sqrt{T})$. □

At this point we believe is very important to enstablish a connection between the hedge algorithm and Online Gradient Descent. Although these algorithms may seem entirely different, they share the same idea. Recall that the hedge algorithm has a regret bound of $O(\sqrt{T \log n})$. If we apply the gradient descent rule to the same problem we'll achieve a regret of $O(\sqrt{nT})$ since the diameter of the probability simplex is $\sqrt{2}$ and the l_2-norm of the costs is bounded by $\sqrt{n}$. Hence, the hedge algorithm is better. Now, recall that the gradient step is given also by the formula:

$$x_{t+1} = arg\min_{x \in X} \{\nabla f_t(x_t)^T x + \frac{1}{2\eta_t} \left\| x - x_t^2 \right\|\}$$

Thus gradient descent can be equivalently be seen by minimizing the first-order Taylor approximation of f_t around x_t plus a regularization term $\left\| x - x_t^2 \right\|$ which keeps x_{t+1} close to x_t. If we are able to change the regularization and impose a different one, the method of mirror descent (online in our case) arises. In general, such a regularization term is called *Bregman Divergence*. We won't delve into the technicalities of mirror descent. An interested reader can see [16] for a mathematically rigorous analysis of mirror descent.

Actually, apart from Gradient Descent which is Mirror Descent when the regularizer is l_2^2, the hedge algorithm is a special case when the regularizer is the relative entropy function which is widely used in learning problems when one maintains a distribution over a set of elements, in this case, the experts. Therefore, when we use as a regularizer, the relative entropy, the update rule becomes (where Δ_n is the probability simplex):

$$x_{t+1} = arg \min_{x \in \Delta_n} \epsilon c^T x + \sum_{i=1}^{n} x(i) \log(\frac{x(i)}{x_t(i)}) - \sum_{i=1}^{n} x(i) + \sum_{i=1}^{n} x_t(i)$$

Now using the KKT conditions we get:

$$\epsilon c_i(t) + \log(x_{t+1}(i)) - log(x_t(i)) - \mu_i - \lambda = 0 \Rightarrow x_{t+1}(i) = x_t(i) e^{\mu_i + \lambda} e^{-\epsilon c_t(i)}$$

Since we start with the uniform distribution it will always hold that $x_t > 0$. This means that $\mu = 0$ from the complementary slackness condition. Now for the value of λ, observe:

$$\sum_{i=1}^{n} x_{t+1}(i) = 1 \Rightarrow \sum_{i=1}^{n} x_t(i) e^{\lambda} e^{-\epsilon c_t(i)} = 1 \Rightarrow$$

$$e^{\lambda} = \frac{1}{\sum_{i=1}^{n} x_t(i) e^{-\epsilon c_t(i)}}$$

Finally, the update rule is the same as of the Hedge Algorithm:

$$x_{t+1}(i) = \frac{x_t(i) e^{-\epsilon c_t(i)}}{\sum_{j=1}^{n} x_t(j) e^{-\epsilon c_t(j)}}$$

3.3 Competing against a dynamic comparator

So far, we've assumed that the player aims at finding the best fixed strategy in hindsight, x^*, which minimizes the sum $\sum_{t=1}^{T} f_t(x)$. That is, that the player is competing against a *static* strategy. Now, we'll discuss a much more difficult problem for the player. Suppose the player aims at minimizing:

$$\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x_t^*)$$

which from now on, we'll call it *Dynamic Regret*. In other words, the player has to compete the optimal *dynamic* strategy which consists of the series of minimizers $x_1^*, x_2^*, ..., x_T^*$, $x_t^* = arg \min_{x \in X} f_t(x)$ of each function the adversary presents in each round. Now we have a tracking problem which is to follow the path of these minimizers instead of converging to a single point. It is straightforward, that the regret we acquired in the static case cannot be attained in this scenario. Recall that the gradient step we

made in the static case brought us closer to the optimal static decision x^*, but now, if the functions f_t and f_{t+1} are uncorrelated the gradient step makes no sense at all.

In order to provide a meaningful benchmark for this kind of problem we have to make strong assumptions on the functions f_t. Without such assumptions, which enstablish a correlation between functions in successive rounds, there is no strategy by the player that it makes sense and the dynamic regret cannot be bounded (The player simply plays blindly). Several measures of variation between the functions have been considered so far. For instance,

$$V_T = \sum_{t=1}^{T} \sup_{x \in X} |f_t(x) - f_{t-1}(x)|$$

measures the maximum variation of consecutive functions f_{t-1}, f_t. Such an assumption was considered in [11] to provide an expected dynamic regret of $O(T^{\frac{2}{3}}(1+V_T)^{\frac{1}{3}})$ for convex functions and $O(\sqrt{T(1+V_T)})$ when strong convexity is considered. In this paragraph, we'll design an algorithm that provides a dynamic regret bound according to the following measure:

$$L_T = \sum_{t=1}^{T} \left\| x_t^* - x_{t-1}^* \right\|$$

which measures the distance of the minimizers of successive functions. A relatively small value of L_T means that the minimizers are close from each other. Thus, a gradient step makes sense in this scenario. In round t, knowing that the minimizer of f_t is close to f_{t-1} is invaluable knowledge to the player. We'll show that online gradient descent with an extra step, provides a dynamic regret of $O(1+L_T)$ (ignoring other constants) when the functions f_t are strongly convex and smooth. The algorithm is described below and it was first analyzed in [40].

Algorithm 5: Online Gradient Descent (dynamic case)

1: **Input:** number of iterations T, feasible set X, initial point $x_1 \in X$, stepsize γ, constant $h \in (0, 1]$
2: **for** $t = 1 : T$ do
3: Choose x_t and suffer loss $f_t(x_t)$
4: Perform gradient step: $y_{t+1} = x_t - \frac{1}{\gamma}\nabla f(x_t)$
5: Project onto X: $\hat{x}_{t+1} = \Pi_X(y_{t+1})$
6: $x_{t+1} = (1-h)x_t + h\hat{x}_t$
7: **end for**

Notice that the step size is constant now, in contrast with the static case, and does not decrease as a function of t since now we don't have a problem of convergence but rather a problem of tracking. The player now wants its play x_t to be close to the minimizer x_t^*. We'll make the following assumptions for every $t = 1, 2, ..., T$:

- f_t are μ-strongly convex
- f_t are L-smooth. Recall that is equivalent to the Lipschitz assumption on the gradient: $\|\nabla f_t(x) - \nabla f_t(y)\|, \ x, y \in X$
- f_t are G-Lipschitz. This is equivalent to the norm of the gradient be bounded by G: $\|\nabla f_t(x)\| \leq G$.

The proof, although highly technical, has a very simple idea: If we'd able to bound the value $\|x_t - x_t^*\|$ by $O(1 + L_t)$ our job is over since the functions are Lipschitz. We start by showing that $\|x_{t+1} - x_t^*\| \leq \rho \|x_t - x_t^*\|$ for some $\rho > 0$, which we'd expect to hold since in time t the algorithm makes a step towards the minimizer of f_t and thus can be seen as part of the proof for the offline version of gradient descent we discussed in the previous chapter. From there the desired bound is straightforward through the triangle inequality. Hence, in order to prove the main theorem, we start with the follow lemma which utilizes the powerful assumption that the function is well-conditioned.

Lemma 3.8. *If the step size $\gamma \geq L$, for the decisions $x_t, \ t \in [T]$ of the player (Algorithm 5) and the minimizers of the functions $x_t^*, \ t \in [T]$ it holds:*

$$\|x_{t+1} - x_t^*\| \leq \rho \|x_t - x_t^*\|$$

where $0 \leq \rho = \sqrt{1 - \frac{h\mu}{\gamma}} < 1$ is a non-negative constant strictly smaller than 1.

Proof. In round t, by strong convexity of f_t, $\forall x \in X$:

$$f_t(x) - \frac{\mu}{2}\|x - x_t\|^2 \geq f_t(x_t) + \nabla f_x(x_t)^T(x - x_t) \Rightarrow$$
$$f_t(x) - \frac{\mu}{2}\|x - x_t\|^2 \geq f_t(x_t) + \nabla f_t(x_t)^T(\hat{x}_t - x_t) + \nabla f_t(x_t)^T(x - \hat{x})$$

Now recall theorem 2.8 and that the gradient step is given also by:

$$\hat{x_t} = arg\min_{x \in X}\{\nabla f_t(x_t)^T(x - x_t) + \frac{\gamma}{2}\|x - x_t\|^2\}$$

Hence, we get:

$$f_t(x_t) - \frac{\mu}{2}\|x - x_t\|^2 \geq f_t(x_t) + \nabla f_t(x_t)^T(\hat{x}_t - x_t) + \gamma(x_t - \hat{x}_t)^T(x - \hat{x}_t)$$

By the smoothness of f_t and the fact that $\gamma \geq L$:

$$f_t(\hat{x}_t) \leq f_t(x_t) + \nabla f_t(x_t)^T(\hat{x}_t - x_t) + \frac{\gamma}{2}\|\hat{x}_t - x_t\|$$

Combining the two above inequalities:

$$f_t(x) - \frac{\mu}{2}\|x - x_t\|^2 \geq f_t(\hat{x}_t) - \frac{\gamma}{2}\|\hat{x}_t - x_t\| + \gamma(x_t - \hat{x}_t)^T(x - \hat{x}_t) \Rightarrow$$
$$f_t(x) - \frac{\mu}{2}\|x - x_t\|^2 \geq -\frac{\gamma}{2}\|\hat{x}_t - x_t\| + \gamma(x_t - \hat{x}_t)^T(x - x_t)$$

By setting $x = x_t^*$:

$$0 \geq f_t(x_t^*) - f_t(\hat{x}_t) \geq \frac{\mu}{2}\|x_t^* - x_t\|^2 + \frac{\gamma}{2}\|\hat{x}_t - x_t\|^2 + \gamma(x_t - \hat{x}_t)^T(x_t^* - x_t) \Rightarrow$$
$$(x_t - \hat{x}_t)^T(x_t - x_t^*) \geq \frac{1}{2}\|\hat{x}_t - x_t\|^2 + \frac{\mu}{2\gamma}\|x_t^* - x_t\|^2$$

Now recall algorithm 4:

$$\|x_{t+1} - x_t^*\|^2 = \|x_t - x_t^*\|^2 + h^2\|x_t - \hat{x}_t\|^2 - 2h(x_t - x_t^*)^T(x_t - \hat{x}_t)$$

By substitution of the inner product:

$$\|x_{t+1} - x_t^*\|^2 \leq (1 - \frac{h\mu}{\gamma})\|x_t - x_t^*\|^2 + h(h-1)\|x_t - \hat{x}_t\|^2 \Rightarrow$$
$$\|x_{t+1} - x_t^*\|^2 \leq (1 - \frac{h\mu}{\gamma})\|x_t - x_t^*\|^2$$

And the lemma follows. □

Now if we're able to bound $\sum_{t=1}^{T}\|x_t - x_t^*\|$, we're almost done:

Lemma 3.9. *Consider the diameter of the set X to be equal to D. If the step size $\gamma \geq L$ and for the decisions $x_t,\ t \in [T]$ of the player (Algorithm 4) and the minimizers of the functions $x_t^*,\ t \in [T]$ it holds:*

$$\sum_{t=1}^{T}\|x_t - x_t^*\| \leq \frac{D}{1-\rho}L_T + \frac{1}{1-\rho}$$

Proof. By using the triangle inequality:

$$\sum_{t=1}^{T}\|x_t - x_t^*\| \leq \|x_1 - x_1^*\| + \sum_{t=2}^{T}\left\|x_t - x_{t-1}^*\right\| + \sum_{t=2}^{T}\left\|x_t^* - x_{t-1}^*\right\|$$

Now, due to lemma 3.8:

$$\sum_{t=1}^{T}\|x_t - x_t^*\| \leq \|x_1 - x_1^*\| + \rho\sum_{t=1}^{T}\|x_t - x_t^*\| + \sum_{t=2}^{T}\left\|x_t^* - x_{t-1}^*\right\|$$

By regrouping, the lemma follows. □

Now, we're ready to state the main theorem:

Theorem 3.10. *Algorithm 4 achieves a dynamic regret of:*

$$\textit{Regret} \leq \frac{GD}{1-\rho} L_T + \frac{G}{1-\rho}$$

The theorem follows trivially from lemma 3.9 by using the Lipschitz condition of $f_t,\ t = 1, 2, ..., T$.

Chapter 4

Online Convex Optimization with Switching Cost

4.1 Definition

In this chapter, we focus on the main problem of this thesis: Online Convex Optimization (OCO) with switching cost. We'll see how the problem relates to both the OCO model and a typical problem studied the community of online algorithms, such as those we discussed in Chapter 1. In order to define the problem, we'll start by comparing it with the OCO model we discussed in the previous chapter. The differences between the two are the following:

- **Lookahead:** First, in the OCO model, the player in round t started by stating her decision x_t and then suffering a loss $f_t(x_t)$. This is typical for a learning (or prediction) problem. In this case we say that the player has 0-lookahead. However, in the case of OCO with switching cost the reverse happens. First the player observes a function $f_t(x_t)$ and then plays her decision x_t. We now say that the player has 1-lookahead. This is typical in the field of online algorithms.

- **Switching between actions:** It's clear that OCO under 1-lookahead is a trivial problem. Although OCO with switching cost is easier in a sense that the player knows the function f_t before taking any action, now there is also a switching cost between the decision x_{t-1}, x_t which makes immediately the problem non-trivial. A typical case of switching cost function we'll discuss is the norm (for instance, the euclidean norm, l_2) of the difference of x_{t-1} and x_t i.e $\|x_t - x_{t-1}\|$ which translates in how much different are the actions of successive rounds. Notice now, that unlike the OCO setting, the function the player receives in each round is not chosen entirely by the adversary. The player receives the function $f_t(x_t) + \|x_t - x_{t-1}\|$.

The first term is chosen by the adversary while the second term is based on the previous action of the player. Notice also that this function is convex as a sum of convex functions.

- **Benchmark:** In the previous chapter we discussed OCO according to two benchmarks: Static and Dynamic. The static case in OCO with switching cost is shown to be no more challenging than OCO [3] and algorithms such as Online Gradient Descent perform well in the case of OCO with switching cost. Our focus will be on the dynamic case which is an active area of research. Apart from the dynamic regret (or competitive difference), which bounds the difference between the cost of the player and the optimal offline cost there is also the competitive ratio, the benchmark used in the field of online algorithms, which bounds the ratio of these costs. As the authors comment in [24] the techniques in order to achieve good bounds in each of these benchmarks are different. We'll most aim at designing competitive algorithms. Observe that in order for an analysis based on competitive ratio to make sense, the convex cost functions the adversary reveals have to be non-negative. In addition, the authors of [3] perform an extensive study on whether is possible for a single algorithm to achieve good static regret and competitive ratio simultaneously. They give a negative answer which is demonstrated by the following theorem:

  Theorem 4.1. *There is no online algorithm (randomized or deterministic) which can achieve sublinear static regret and constant competitive ratio for an online convex optimization problem with switching cost even when the cost functions are linear.*

  and they design an algorithm for the unidimensional case (Randomly Biased Greedy) which achieves simultaneously a competitive ratio of $O(1+\gamma)$ while maintaining a $O(\max\{\frac{T}{\gamma}, \gamma\})$ static regret.

With the above in mind, we're ready to define the problem of OCO with switching cost when the switching cost function is a norm. This easily extends to the case where the switching cost function is a general convex function.

An instance of the problem consists of a fixed decision space, a convex set $X \in \mathbb{R}^n$ and a sequence of non-negative convex cost functions $f_t(x),\ t \in [T]$. In round t, the player observes the function $f_t(x)$ and chooses a point $x_t \in X$ incurring a hitting (or service) cost $f_t(x_t)$ and a switching cost $\|x_t - x_{t-1}\|$. We assume that both the player and the adversary start from the origin. The total cost of the player is:

$$\sum_{t=1}^{T} f_t(x_t) + \|x_t - x_{t-1}\|$$

while the optimal cost is:

$$\min_{\{x_t\}_{t=1}^T \in X} \sum_{t=1}^{T} f_t(x_t) + \|x_t - x_{t-1}\|$$

We'll denote the adversary's decisions as $\{x_t^*\}_{t=1}^T$. Notice that the offline problem is a convex optimization problem and can be solved efficiently. Also, the problem can be stated as an uncostrained problem when we consider the extension of function f_t in $\mathbb{R}^n$ which takes the value $f_t(x),\ x \in X$ and the value $+\infty,\ x \notin X$. The epigraph of the function is the same as of f_t and thus convexity is preserved. We'll find this fact useful in the analysis of the algorithms that will follow.

The algorithm that comes naturally to someone's mind for this problem is to pick the decision x_t that minimizes the sum of the hitting and switching cost at time t. In first sight, it may seem that is the best the player can do, but as we'll see soon it fails miserably and we'll explain why. More precisely:

$$x_t = arg\min_{x \in X} f_t(x) + \|x - x_{t-1}\|$$

Consider now the following scenario in one dimension. The adversary reveals a function of the form $f_t(x) = \frac{a}{2}(x-c)^2,\ a, c > 0$. The player starting from the origin will pick a point $x_1 \in [0, c]$ that minimizes the sum:

$$\frac{a}{2}(x-c)^2 + x$$

Which is minimized at $x = c - \frac{1}{a}$. Notice now that if $ca = 1$, the player stays at 0. If the adversary reveals functions of this form indefinitely, the player will still continue to stay at 0. That makes her extremely vulnerable to the adversary. After T rounds, if the player receives the indicator function of the c, which takes the value 0 at c and $+\infty$ otherwise the player has to move necessarily to c. Thus the player has a a total hitting cost of cT while the adversary simple moves to the first round, for a total cost of c. Therefore the competitive ratio of this algorithm is $\Omega(T)$ and is due to the fact that the use of such a criterion by the player may stuck her at one point.

Such a result shows us that we have to come up with an algorithm that always makes a small step towards the minimizer of function f_t, no matter what. An interesting idea, which will encounter in later paragraphs, is to consider a gradient based rule of the function f_t, $x_t = x_{t-1} - \frac{1}{\gamma_t}\nabla f(x_{t-1})$. For the 1-d case, without loss of generality if $x_{t-1} < x_m$ the player will have to pick a point in the interval $(x_{t-1}, x_m]$. Notice now, that every possible algorithm for the unidimensional problem can be equilavently described as a gradient descent rule with an appropriate choice of γ_t. However, there

is an algorithm that comes more intuitively for the problem than the gradient descent rule. We pick a point x_t in order to balance between the costs $f_t(x_t)$ and $\|x_t - x_{t-1}\|$. As we'll see such an idea provides constant competitive ratio for the unidimensional case as well as constant competitive ratio for higher dimensions when we add an assumption for the functions f_t.

4.2 The unidimensional case

We start to delve into the problem by studying the unidimensional case where is proved that we can achieve a constant competitive ratio. As we'll see, the algorithm for higher dimensions is based on the idea of the one we'll discuss in this paragraph. Moreover, note that any l_p norm in the real line reduces to the absolute value. Firstly, we'll prove that randomization provides no benefit for the design of a competitive algorithm for OCO with switching cost.

Proposition 4.2. *For an OCO problem with switching cost, if there is a c-competitive randomized algorithm R then there is also a c-competitive deterministic algorithm D.*

Proof. The proof of the theorem is based on *Jensen's Inequality* which states that for a random variable X and a convex function g it holds:

$$g[\mathbb{E}[X]] \leq \mathbb{E}[g[X]]$$

In the general case, a randomized algorithm R maintains at time t a probability distribution over the the real line. Let x_t be a random variable according to this distribution. Then the expected cost of the randomized algorithm in round t is:

$$\mathbb{E}[f_t(x_t)] + \mathbb{E}[|x_t - x_{t-1}|]$$

Now consider instead of sampling from the distribution we just take as a decision the expected value of x_t, $\{\mathbb{E}[x_t]\}$. In this case, the respective cost of the now deterministic algorithm is:

$$f_t(\mathbb{E}[x_t]) + |\mathbb{E}[x_t] - \mathbb{E}[x_{t-1}]|$$

Since the objective is a convex function, by Jensen's inequality, we conclude that in round t, the deterministic algorithm has a lower or equal cost than the expected cost of the randomized algorithm. Summing over all $t \in [T]$ completes the proof. □

An important concept in the design of algorithms for OCO with switching cost is the notion of *memory* and has to do with what information in mind the player chooses a

point x_t in round t. Generally speaking, if the player just uses her current position x_{t-1} and $f_t(x_t)$ in order to decide x_t then the algorithm is memoryless. If her decision is based on her previous states as well as the previous functions the adversary revealed then the algorithm is with memory. For the unidimensional case, we'll discuss 2 algorithms for the two kinds of memory we mentioned. An algorithm that keeps memory can achieve a tight competitive ratio of 2 while a memoryless algorithm can achieve a tight competitive ratio of 3.

4.2.1 A memoryless algorithm

The algorithm that we'll discuss now was developed in [8]. The idea of the memoryless algorithm is the following: Without loss of generality suppose the player is at state x_{t-1} and the adversary reveals a convex function f_t with a minimizer at $x_m > x_{t-1}$ (which we'll assume it always exists). Then the player moves to the direction of x_m (to the right) until $|x_t - x_{t-1}| = \frac{f_t(x)}{2}$ (which can be performed using binary search) and balances the hitting and the switching cost. Notice that if $\frac{f_t(x_m)}{2} < |x_m - x_{t-1}|$ such an x_t always exists in the interval (x_{t-1}, x_m). If $\frac{f_t(x_m)}{2} \geq |x_m - x_{t-1}|$, the algorithm simply moves to x_m. From now on we denote as $H_t = f_t(x_t)$ and $M_t = |x_t - x_{t-1}|$ the hitting and the moving cost of the algorithm respectively. Similarly for the adversary we denote as H_t^*, M_t^*.

Algorithm 1: Memoryless algorithm

1: **for** $t = 1 : T$ do
2: Let $x_m = arg\min f_t(x)$
3: Move in the direction of x_m until we reach either a point x such that $M_t = \frac{H_t}{2}$ or x_m
4: Set x_t as that point
5: **end for**

Theorem 4.3. ***Algorithm 1*** *is 3-competitive for 1-d OCO with switching cost*

Proof. The proof is based on the potential function method we discussed in chapter 1. We consider the potential function $\Phi_t(x_t, x_t^*) = 3|x_t - x_t^*|$ with $\Phi_0 = 0$ since the player and the adversary start from the origin. In order to prove that the algorithm is 3-competitive it suffices to show $\forall t \in [T]$:

$$H_t + M_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_{t-1}^*) \leq 3(H_t^* + M_t^*)$$

Since while moving towards x_m the hitting cost is decreased while the moving cost is increased it always holds that: $M_t \leq \frac{H_t}{2}$ and the equality holds when $x_t \neq x_m$. First

observe from the triangle inequality:

$$\Phi(x_{t-1}, x_t^*) - \Phi(x_{t-1}, x_{t-1}^*) \leq 3M_t^*$$

Therefore it suffices to prove:

$$H_t + M_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) \leq 3H_t^*$$

In order to prove this, we consider two cases:

- $H_t \leq H_t^*$. In this case we get:

$$\begin{aligned} H_t + M_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) &\overset{M_t \leq \frac{H_t}{2}}{\leq} H_t + \frac{H_t}{2} + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) \\ &\overset{\text{TI}}{\leq} \frac{3}{2}H_t^* + \Phi(x_t, x_{t-1}) \leq \frac{3}{2}H_t^* + \frac{3}{2}H_t^* \leq 3H_t^* \end{aligned}$$

- $H_t > H_t^*$. Consider $x_{t-1} < x_m$ (the reverse case is similar). Since $H_t > H_t^*$ we can't have $H_t = f_t(x_m)$ and thus $M_t = \frac{H_t}{2}$. Observe that we must have $x_{t-1} < x_t$ since the algorithm moves to the right and $x_t < x_t^*$ since $H_t > H_t^*$. Thus:

$$\begin{aligned} &\Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) = 3(x_t^* - x_t - x_t^* + x_{t-1}) = -3M_t \Rightarrow \\ &H_t + M_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) = \frac{3}{2}M_t - 3M_t < 0 \leq 3H_t^* \end{aligned}$$

And the proof is complete. □

As we previously commented, a competitive ratio of 3 is actually tight. We'll now discuss the proof of the lower bound. But first let's discuss memorylessness again. Memorylessness as we discussed previously is ill-defined. Because the state of the algorithm is a real number with any accuracy (any number of bits), it's possible to encode the memory of the previous rounds in the low order bits of the state. As the authors claimed in [8] memorylessness is based on the following:

- **Scale**: Algorithm's responses don't depend on the scale of the line. For instance, suppose for the case of a 2-piecewise linear function $a|x - b|$ the algorithm that operates in the interval $[x_{t-1}, b]$ moves to $x_{t-1} + \gamma(a)(b - x_{t-1})$. If now the algorithm had received the function $a|x - c|$, $c \neq b$ under memorylessness the algorithm has to move also to $x_{t-1} + \gamma(a)(c - x_{t-1})$. Thus the decision depends only on the slope of the function, a. As we'll see the instance that gives the lower bound is composed entirely of such functions.

- **Symmetry**: Algorithm's responses are bilaterally symmetric. If the player receives a function to her right will make the same decision (mirrored) as in the case she receives it to her left.

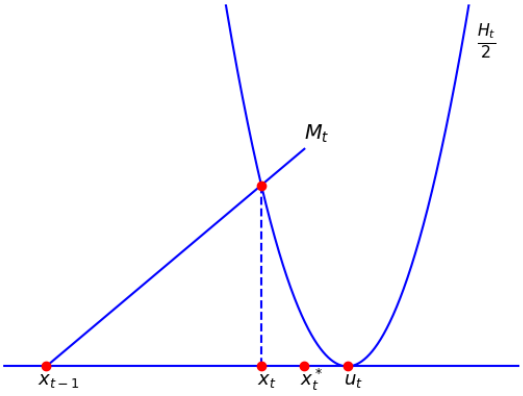


FIGURE 4.1: Illustration of Algorithm 1 for $f_t(u_t) = 0$ and $H_t > H_t^*$

Theorem 4.4. *Any memoryless algorithm for OCO with switching cost attains a competitive ratio of at least 3.*

Proof. Recall that the algorithm starts from the origin. The intuition of the proof is the following: Given a function $\epsilon|1 - x|$ at the first round. We are interested in the case as $\epsilon \to 0$. If the player makes a small step towards $x = 1$, the adversary continues bringing copies of the same function indefinitely. Due to the invariant of memorylessness we mentioned for 2-piecewise linear functions, the steps in the next interval $[x_1, 1]$ will be small as well. The player will make a long time to reach $x = 1$, incurring a high total hitting cost in the process, whereas the adversary simply moves to $x = 1$ at the first round. On the other hand, if the player made a relatively big step at the first round, then the adversary reveals indefinitely functions of the form $\epsilon|x|$ making the optimal solution to stay at 0 early from the beginning. The player not only paid a big distance step, but continues to incurr hitting cost, while the adversary does not. Thus, we'll consider two cases which due to memorylessness will determine the behavior of the algorithm throughout the analysis after the adversary reveals the function $\epsilon|1 - x|$

- The player moves to a point $x \leq \frac{\epsilon}{2}$. Then the adversary continues bringing copies of the function $\epsilon|1 - x|$ indefinitely. Notice now, due to the first property of memorylessness if the algorithm in the future is at a point y then moves necessarily to the point $y + x(1 - y)$. Thus if at round t the hitting cost is $H_t = \epsilon(1 - y)$ then

$H_{t+1} = \epsilon(1 - y - x(1 - y)) = (1 - x)H_t$. Thus the hitting cost is asymptotically:

$$\sum_{t=1}^{+\infty} H_t \geq \sum_{t=1}^{+\infty}(1 - \frac{\epsilon}{2})^t\epsilon = \frac{\epsilon}{x} - \epsilon \geq 2 - \epsilon$$

For $\epsilon \to 0$ the hitting cost is 2 while the moving cost is asymptotically 1. That makes the total cost equal to 3. The adversary simple moves at the first round at $x = 1$ and his total cost is 1.

- The player moves to a point $x \geq \frac{\epsilon}{2}$. Then the adversary reveals the function $\epsilon|x|$ indefinetely and the algorithm returns to the origin. Hence, the moving cost is $2x$. For the hitting cost, recall the second property of memorylessness. For $t \geq 2$, if the algorithm's position at $t - 1$ is y then at t will be $y(1 - x)$ and thus $H_t = (1 - x)H_{t-1}$. Thus the total hitting cost is:

$$\sum_{t=1}^{+\infty} H_t \geq \epsilon(1 - x) + \sum_{t=2}^{+\infty} \epsilon x(1 - x)^{t-1} = 2\epsilon(1 - x)$$

That makes the total cost of the player equal to $2x + 2\epsilon(1 - x) \geq 2\epsilon + (2 - 2\epsilon)\frac{\epsilon}{2} = 3\epsilon - 3\epsilon^2$. The optimal offline cost is ϵ and is attained when one never leaves the origin. Thus the competitive ratio is at least $3 - 2\epsilon$. As $\epsilon \to 0$ a competitive ratio of 3 is attained asymptotically.

□

4.2.2 An algorithm with memory

We'll proceed with the description of an algorithm for the unidimensional case that in contrast with the one we mentioned, it utilizes memory. In fact, that's the only algorithm so far that has appeared for OCO with switching cost that utilizes information from the previous rounds and it was presented in [8]. Algorithms that utilize memory have appeared extensively in the online learning and OCO community, named *follow the leader*. An interested reader can see [28], [41] for an elaborate discussion of these algorithms.

The algorithm in each round maintains a probability distribution over the real line. As we mentioned earlier, randomization provides no benefit at our setting. So the response of the algorithm can be the expected value of the distribution at time t. The algorithm utilizes memory through the distribution. The decision of the algorithm at round t is not based only on its place at time $t - 1$ and the function f_t but also at the distribution the algorithm maintains at time $t - 1$. The idea of the algorithm is to pick carefully designated points $[x_l, x_r]$ where $x_l \leq x_m \leq x_r$ and to create a probability

distribution with support the interval $[x_l, x_r]$. We describe more formally the algorithm below:

Algorithm 2: Algorithm with memory

1: **for** $t = 1 : T$ do
2: Let $x_m = arg\min f_t(x)$
3: Pick point $x_r \geq x_m$ such that $\frac{1}{2}\int_{x_m}^{x_r} f''(y)dy = \int_{x_r}^{+\infty} p_{t-1}(y)dy$
4: Pick similarly $x_l \leq x_m$ such that $\frac{1}{2}\int_{x_l}^{x_m} f''(y)dy = \int_{-\infty}^{x_r} p_{t-1}(y)$
5: Update the pdf $p_t(x) = p_{t-1}(x) + \frac{1}{2}f''(x),\ x \in [x_k, x_r]$ and $p_t(x) = 0$, otherwise
6: Choose $\mathbb{E}_x[p_t(x)]$
7: **end for**

The authors in [8] show that the update rule indeed maintains a valid probability distribution over the real line, and they prove that the algorithm is 2-competitive which is tight for any algorithm for OCO with switching cost. The idea of the lower bound comes from the ski-rental problem which is a special case of OCO with switching cost. Consider a ski-rental problem where the cost of buying skis is 1 and the cost of renting skis for 1 day is ϵ. The ski-rental problem over T days can be seen as an OCO with switching cost problem where the player starts from the origin and the adversary reveals the function $\epsilon|1 - x|$ for T days. For a determintic ski-rental algorithm if the player chooses to rent, is equivalent to stay at the origin incurring a cost of ϵ and if the player chooses to buy incurrs a cost of 1. Notice now, that any randomized algorithm for the ski-rental problem where the player at time t has bought skis with a probability p_t is equivalent to go to the point $x_t = p_t, \in [0, 1]$. Therefore, we immediately conclude that $\frac{e}{e-1}$ is a lower bound for OCO with switching cost. However, the problem is strictly harder. The authors in [8] provide a lower bound of 1.86 which is based on the idea we discussed for the lower bound of the memoryless algorithm which is after some time, based on the player's position in $[0, 1]$ to bring an infinite number of functions $\epsilon|x|$. However, in this case, the invariant the memorylessness force to the players doesn't appear and the authors prove that any algorithm is 1.86 competitive. They conjecture that 2 is not also the optimal competitive ratio. Finally, in [5], that claim was disproved. The authors provide a lower bound of 2 where also only functions of the form $\epsilon|x - 1|$ and $\epsilon|x|$ are considered in the instance.

4.3 Higher dimensions

So far, we've discussed the problem in 1 dimensions and we analyzed a 3-competitive memoryless algorithm. In this paragraph we'll prove that the same idea, which is to balance between the hitting and the switching cost at time t provides a constant competitive

ratio when the functions f_t have a basic property.

But first we'll prove a fundamental lower bound for OCO with switching cost which shows that the competitive ratio for certain type of switching costs depends necessarily on the dimension of the instance. That leads us to perform beyond worst case analysis for the case of the competitive ratio and to consider a subset of convex functions. The bound is based on another problem, Convex Body Chasing, one is given in an online fashion a sequence of convex sets $X_1, X_2, ..., X_T$. When a set X_t arrives the player must move to a point $x \in X$. The total cost of the algorithm is the distance that the player has traveled $\sum_{t=1}^{T} d(x_{t-1}, x_t)$. In OCO with switching cost when the function f_t is the indicator function of some convex set X_t then OCO with switching cost reduces to Convex Body Chasing.

Theorem 4.5. *Any algorithm for OCO with switching cost in d dimensions attains a competitive ratio of at least* $\Omega(\sqrt{d})$ *when the switching cost is the* l_2 *norm.*

Proof. The intuition behind the proof is that the adversary can bring a function f_t which is minimized over a set X_t and has a very large value outside X_t rather than being minimized at single point and creates an instance of series of functions $\{f_t\}_{t=1}^{T}$ that all of them have different sets that are minimized, although they have a common minimizer, the intersection of the sets. The adversary simply moves to that minimizer in the first round whereas the player has to move to a point in X_t where the function is minimized. When the next function arrives the player has to move again. More formally, for an instance of OCO with switching cost in d dimensions the adversary brings d functions. The adversary creates the function f_t based on the t-coordinate of the place of the player in time t.

- If $x_t < 0$, then the adversary brings the indicator function of the hyperplane $x_t = 1$. Thus the player has to pay a moving cost of at least 1.
- If $x_t \geq 0$, then the adversary brings the indicator function of the hyperplane $x_t = -1$. Again, the player pays at least a cost of 1.

Thus, the cost of the player is at least d. However, these functions have a common minimizer which the player cannot know beforehand. The adversary simply moves to the intersection of these hyperplanes incurring a cost of $\sqrt{d}$. That implies a lower bound of $\sqrt{d}$ on the competitive ratio. □

The indicator function may seem like a non practical instance for OCO with switching cost but even quadratic functions can be used to derive this lower bound. For instance consider the function $f_t(x) = \frac{1}{\epsilon}(x_i - 1)^2$ for $\epsilon \to 0$ which for $x_1 = 1$ takes the value 0

whereas for $x_1 \neq 1$ takes a very large value. Moreover, notice that the lower bound is different for different switching costs. For instance consider the l_1 norm or the squared euclidean distance. In these cases the proof leads to nowhere. Both the adversary and the player pay a cost of d. On the other hand, for cases like $l_\infty = max_{i=1}^{n}|x_i|$ the lower bound is $\Omega(d)$.

Concerning the l_1 norm, it is easy to notice that under specific circumstances there is an algorithm that is 3-competitive. For the uncostrained case, if the functions $f_t(x)$ have the property that $f_t(x) = \sum_{i=1}^{d} f_{it}(x_i)$ then the problem reduces to the one dimensional problem since the l_1 norm also obeys this property.

4.3.1 Minimizing Competitive Ratio

The lower bound we discussed implies a competitive ratio which depends on the dimension of the instance. In order to break this barrier we'll consider convex functions that obey a specific property. Recall that the lower bound was based on the fact that the function may be minimized in a set rather than a single point. Due to this fact, we could start our analysis by considering functions that have a unique minimizer. Such functions are norms. Hence, we start by the definition of α-polyhedral functions.

Definition 4.6. A function f_t defined in a set X with minimizer u_t is α-polyhedral with respect to a norm $\|\cdot\|$ if $\forall x \in X \ f_t(x) - f_t(u_t) \geq \alpha \|x - u_t\|$.

That means that the functions we consider are bounded below by a norm function, or in other words, grow at least linearly away from the minimizer. It's important to notice that functions which obey this property for even x that are ϵ close to the minimizer $\|x - u_t\| \leq \epsilon$ are still α-polyhedral. Without loss of generality consider $u_t = 0$ and $f_t(u_t) = 0$. From the definition of convexity we have that for any $\lambda \in [0, 1]$ it holds: $f(\lambda x) \leq \lambda f(x)$. For every $x \in X$ there exists a λ such that $\|\lambda x\| \leq \epsilon$. Thus $f(x) \geq \frac{\alpha\|\lambda x\|}{\lambda} = \alpha \|x\|$.

Now we'll discuss how we can generalize using the memoryless algorithm we analyzed for the 1 dimension in the previous paragraph. Recall that the algorithm begins a movement from x_{t-1} towards u_t (the minimizer of f_t) until there is balance between the hitting and the switching cost. However, we could interpret this procedure in another way. Consider the set $f_t(x) \leq l, \ l \geq 0$ which is convex. Initially, we start with $l = f_t(u_t)$ and we project the point x_{t-1} to this set. Thus $x_t = u_t$. We start now by increasing l until the balance is achieved. This is equivalent to the procedure in 1-dimension but it is more formally stated and this idea can be applied to higher dimensions. The projection step in 1 dimension is of course trivial but in higher dimensions, as we have commented in past chapters, in the general case, is an optimization problem. Notice now, that in

order to achieve a hitting cost of l we could choose any point in the respective l level set of f. However, the one that minimizes the switching cost, is the projection of point x_{t-1} to the sublevel set. That's why a procedure which would find a point in order to balance the costs in the convex hull of x_{t-1}, u_t or a simple gradient step won't work well for a general convex function. These methods don't take into consideration the geometry of the level sets of the function f_t.

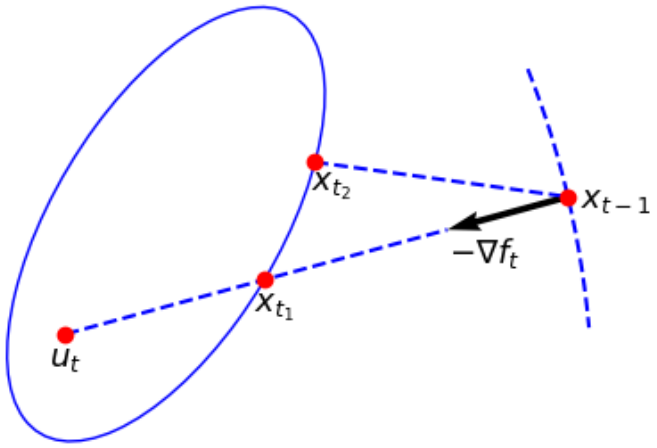


FIGURE 4.2: Geometric explanation of the above statement. Here we have $H_t(x_{t_1}) = H_t(x_{t_2})$ but $M_t(x_{t_1}) > M_t(x_{t_2})$

The algorithm for higher dimensions will project the point x_{t-1} onto a subset level set of f_t, $\{x : f_t(x) \leq l\}$ for a designated choice of l in order for the balance to be achieved. Of course, the choice of l is not given by a closed formula. In general, we have to find l such that $\|x_t - x_{t-1}\| = \beta f_t(x_t)$. Or, if $x(l)$ is the projection of x_{t-1} onto the l-sublevel set of f_t then $\|x(l) - x_{t-1}\| = \beta l$. Equivalently, to find a value of λ such that $g(l) = \beta l$ where $g(l) = \|x(l) - x_{t-1}\|$ is a function in 1 dimension and we'll prove it's continuous. Thus, l can be found using bisection. We describe the algorithm below, which was appeared in [24]. Moreover, in this paper, they propose an algorithm which balances between the norm of the gradient (instead of the objective cost) and the switching cost that provides a dynamic regret proportional to $\sqrt{L_T}$. Recall from the previous chapter that L_T is the length of the trajectory of the points of the optimal offline solution.

Algorithm 3: Online Balanced Descent

for $t = 1 : T$ do
    Set $u_t = arg\min_x f_t(x)$
    if $\|u_t - x_{t-1}\| < \beta f_t(u_t)$
        Set $x_t = u_t$
    else
        Let $x(l) = \Pi_{X_t^l}(x_{t-1})$, increase l from $f_t(u_t)$ until $\|x(l) - x_{t-1}\| = \beta l$
        where X_t^l is the l-sublevel set of f_t
        Set $x_t = x(l)$
end for

In general, β will be less that 1 and its exact value will be determined from the analysis and will depend on the α of the α-polyhedral function. Notice also, that if $\|u_t - x_{t-1}\| < \beta f_t(u_t)$ there is not l such that $\|x(l) - x_{t-1}\| = \beta l$. In this case, the player simply moves to the minimizer of f_t. Recall that this case was taken into consideration on the algorithm for the unidimensional case. In the case that $f_t(u_t) = 0$ such an l will always exists. Moreover, notice that although this algorithm seems like solving an uncostrained problem, the feasible set X can be incorporated into the sublevel set of f_t if we consider $f_t = f_t, x \in X$ and $+\infty$, otherwise.

Although we mentioned that we can increase l from 0 until $\|x(l) - x_{t-1}\| = \beta l$, a value of l can be find more efficiently using bisection. The function $g(l)$ takes the value 0 for an l such that $g(l) = 0$ (consider simply a sublevel set for $l = f_t(x_{t-1})$) and a value greater or equal than βl for $l = f_t(u_t)$. According to the following lemma, the function $g(l)$ is continuous and thus bisection can be employed.

Lemma 4.7. *The function $g(l) = \|x(l) - x_{t-1}\|$ is continuous in l.*

Proof. To show the above lemma, we start by showing that $h(l) = \frac{1}{2}\|x(l) - x_{t-1}\|^2$ is continuous in l. Recall that $x(l)$ is the solution to the following optimization problem (projection):

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}\|x - x_{t-1}\|^2 \\ \text{subject to} \quad & f_t(x) \leq l \end{aligned}$$

A remark to be made at this point is that after taking KKT conditions for the above problem one sees that:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_t)$$

which is similar to a step of gradient descent but with the huge difference that instead of computing the gradient at x_{t-1} one has to compute it at x_t. That gives power to the algorithm compared to a simple gradient descent, since gradient descent only limits the

algorithm's actions only to one direction. Notice also that when we have OLO (Online linear optimization) with switching cost, for designated choice of gradient stepsize, the two methods are equivalent.

In the above form is somewhat hard to prove the claim. Using duality (notice that strong duality holds) we can equivalently write $h(l)$ as a function. In particular,

$$h(l) = \max_{\lambda \geq 0} \min_{x} \{ \frac{1}{2} \|x - x_{t-1}\|^2 + \lambda(f_t(x) - l) \} = \min_{x} \max_{\lambda \geq 0} \{ \frac{1}{2} \|x - x_{t-1}\|^2 + \lambda(f_t(x) - l) \}$$

Now, let $H(x, \lambda, l) = \min_x \max_{\lambda \geq 0} \{ \frac{1}{2} \|x - x_{t-1}\|^2 + \lambda(f_t(x) - l) \}$. The equation of the two forms follows from the minimax theorem since $H(x, \lambda)$ is convex in x for constant λ and affine (thus concave) in λ for constant x. Recall now that maximization preserves convexity (for every λ, we get a convex function in (x, l)). Moreover, $H(x, \lambda, l)$ is jointly convex in (x, l) and thus minimization over x preserves convexity. Finally since $h(l)$ is convex, it must be continuous. Now for the continuity of $g(l)$, give a $\epsilon > 0$ we can find a $\delta > 0$ such that $|h(l) - h(l + \delta)| < \epsilon^2$. Thus,

$$|g(l) - g(l + \delta)| \overset{TI}{\leq} \|x(l) - x(l + \delta)\| = \sqrt{\|x(l) - x(l + \delta)\|^2}$$

Now recall the cosine law for the projection step from Chapter 2:

$$|g(l) - g(l + \delta)| \leq \sqrt{\|x(l + \delta) - x_{t-1}\|^2 - \|x(l) - x_{t-1}\|^2} = \sqrt{|h(l) - h(l + \delta)|} < \epsilon$$

□

With the above in mind, we are ready to analyze Algorithm 3 for α-polyhedral functions. Essentially the fact that the hitting costs must be expressed by α-polyhedral functions restricts the use of the algorithm to functions $f_t(x) = g_t(x) + \|x - u_t\|$ where u_t is the minimizer of $g(x)$ is a convex function which must be nonnegative.

Theorem 4.8. *Algorithm 3 achieves a competitive ratio of* $3 + O(\frac{1}{\alpha})$ *for* α*-polyhedral function for the problem of OCO with switching cost.*

Proof. The analysis is very similar to the one for the unidimensional case. We again consider $\Phi(x_t, x_t^*) = C \|x_t - x_t^*\|$. As before, in order to enstablish a competitive ratio of C, it suffices to show that:

$$H_t + M_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_{t-1}^*) \leq C(H_t^* + M_t^*)$$

Taking into consideration the fact that:

$$\Phi(x_{t-1}, x_t^*) - \Phi(x_{t-1}, x_{t-1}^*) \leq CM_t^*$$

Therefore it suffices to prove:

$$H_t + M_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) \leq CH_t^*$$

Recall that always $M_t \leq \beta H_t$. Again consider two cases:

- $H_t \leq H_t^*$. This is the case that remains easy. We have that

$$\begin{aligned} H_t + M_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) &\overset{M_t \leq \beta H_t}{\leq} H_t + \beta H_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) \\ &\overset{\text{TI}}{\leq} (1+\beta)H_t + \Phi(x_t, x_{t-1}) \leq (1+\beta)H_t + C\beta H_t \leq (1+\beta(C+1))H_t \end{aligned}$$

  Thus we search for a value of β such that $1 + \beta(C+1) \leq C$

- $H_t > H_t^*$. That's the difficult case compared to the problem in one dimension. Let's recall what we have to prove and notice that in this case $M_t = \beta H_t$.

$$(1+\beta)H_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) \leq CH_t^*$$

  In the unidimensional case we easily show that $\Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) = -CM_t$ with an obvious observation. In higher dimensions this does not necassarily hold. Although, in general, intuitively the difference should be nonpositive. Remember that x_t is the projection of x_{t-1} to a l-sublevel set of $f_t(x)$. Because $f_t(x_t^*) < f_t(x_t)$, x_t^* must lie in the interior of this sublevel set. Thus the distance between x_t and x_t^* is less than the distance between x_{t-1}, x_t^* because x_t, x_t^*, x_{t-1} form an obtuse triangle. We formulate this idea below. We'll prove that for some $\gamma > 0$:

$$\|x_t - x_t^*\| - \|x_t^* - x_{t-1}\| \leq -\gamma \|x_t - x_{t-1}\|$$

  First, we bound the term $\|x_t - x_t^*\|$ using the triangle inequality:

$$\|x_t - x_t^*\| \leq \|x_t - u_t\| + \|x_t^* - u_t\| \overset{\alpha-polyhedral}{\leq} \frac{1}{\alpha}H_t + \frac{1}{\alpha}H_t^* \leq \frac{2}{\alpha}H_t \leq \frac{2}{\alpha\beta}M_t$$

  Let $\|x_t - x_t^*\| = rM_t,\ r \leq \frac{2}{\alpha\beta}$. Now for the projection, recall that it holds:

$$\begin{aligned} \|x_t - x_t^*\|^2 + \|x_t - x_{t-1}\|^2 \leq \|x_t^* - x_{t-1}\|^2 \Rightarrow \|x_t^* - x_{t-1}\| \geq \sqrt{1+r^2}M_t \\ \Rightarrow \|x_t^* - x_{t-1}\| - \|x_t - x_t^*\| \geq (\sqrt{1+r^2} - r)M_t \geq (\sqrt{1+(\frac{2}{\alpha\beta})^2} - \frac{2}{\alpha\beta})M_t \end{aligned}$$

because $h(r) = \sqrt{1+r^2} - r$ is a strictly decreasing function which is always positive. Hence we conclude $\gamma = \sqrt{1 + (\frac{2}{\alpha\beta})^2} - \frac{2}{\alpha\beta} > 0$. Finally, using the above:

$$H_t + M_t + C(\|x_t^* - x_{t-1}\| - \|x_t - x_t^*\|) \leq H_t + M_t - C\gamma M_t = (1 + \beta(1 - C\gamma))H_t$$

Thus we search for a value of β such that C is minimized and the two following hold:

$$1 + \beta(1 - C\gamma) \leq 0$$
$$1 + \beta(C + 1) \leq C$$

After careful calculations, we conclude that $\beta = \frac{1}{2} + \frac{1}{\alpha+2}$ and $C = 3 + \frac{8}{\alpha}$. □

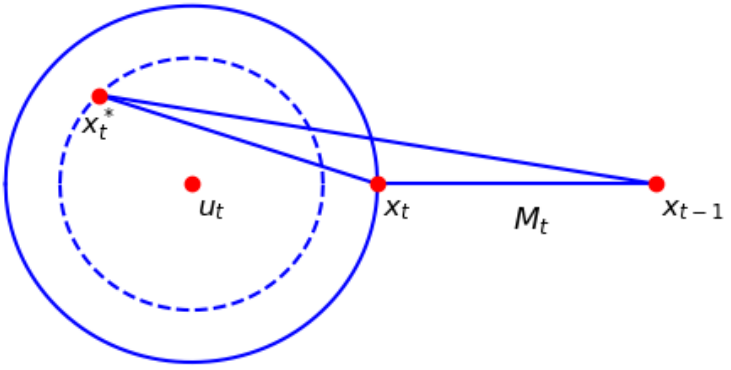


FIGURE 4.3: Relation between x_t, x_{t-1}, x_t^* when $H_t > H_t^*$

The above analysis was for the case that the switching cost was the l_2 norm. However, recall that there is a equivalence between the norms. Therefore we can extend the above result to other norms. For example, for l_1 norm it holds:

$$\frac{1}{\sqrt{d}} \|x\|_1 \leq \|x\|_2 \leq \|x\|_1$$

Therefore the cost of the algorithm for l_1 switching cost is at most $\sqrt{d}$ times the cost of the algorithm for the l_2 case. Similarly, the offline optimal cost for the l_1 norm is no less than the optimal offline cost for the l_2 norm. Therefore we conclude to a competitive ratio of $O(\sqrt{d}(3 + \frac{1}{\alpha}))$ when the switching cost is l_1.

4.4 Covering constraints with l_1 switching cost

In this paragraph, we discuss a special case of OCO with switching cost. In particular, when the objective is a linear function, the switching cost is the (weighted) l_1 norm and the feasible set of solutions is expressed as linear covering constraints. To explain

the motivation behind the development of such an algorithm we have to say first that in combinatorial optimization, a standard procedure to design approximation algorithms for NP-complete problems is to express them as integer programs, consider the linear programming relaxation, trivially acquire a solution, since linear programs can be efficiently solved and then round the solution to provide the result for the combinatorial problem. That's the case here as well for online combinatorial problems wthat involve a switching cost between successive rounds. However, obtaining a good fractional solution is not trivial in this case since it's an OCO with switching cost problem.

In general, a lot of fundamental combinatorial problems can be expressed with covering constraints including the Set Cover problem and the Shortest Path problem. In the Set Cover problem, one has a collection of sets $S_1, S_2, ..., S_n$, each one associated with a cost $c_i,\ i \in [n]$ and a number of elements $e_1, e_2, ..., e_m$. Each of the elements is covered only by a subset of these sets. The goal is to find the Sets with the minimum possible cost that cover all the elements. Subsequently, the online multistage Set Cover problem is when one has the same sets in each round and the cost of each one, $c_{i,t}$ changes between rounds. Moreover, different subset of elements are needed to covered in each round and the player pays a cost when acquiring or removing a Set from her disposal.

The algorithm which will analyze appeared in [18]. The authors provide a competitive algorithm to solve the Set Cover problem with switching cost. They solve the convex relaxation, which is our focus, and they provide a rounding algorithm for acquiring integer solutions which we'll omit from this paragraph. Let's start by considering the relaxation of the problem and its dual. The purpose of defining the dual is because our analysis will be based on the lower bound the dual provides to the optimal offline solution.

$$
\begin{aligned}
\underset{y\in[n]\times[T]}{\text{minimize}} \quad & \sum_{t=1}^{T}\sum_{i=1}^{n} c_{i,t}y_{i,t} + \sum_{t=1}^{T}\sum_{i=1}^{n} w_i z_{i,t} \\
\text{subject to} \quad & \sum_{i\in S_{j,t}} y_{i,t} \geq 1,\ \forall t \geq 1\ \&\ 1 \leq j \leq m_t \\
& z_{i,t} \geq y_{i,t} - y_{i,t-1},\ \forall t \geq 1\ \&\ 1 \leq i \leq n \\
& z_{i,t}, y_{i,t} \geq 0,\ \forall t \geq 1\ \&\ 1 \leq i \leq n
\end{aligned}
$$

Where $0 \leq y_{i,t} \leq 1$ determines what percentage of the Set i we acquire in time t. Notice also, that without loss of generality, it suffices to pay only for increasing variables, since in each round a distribution over the sets is maintained. We denote by $S_{j,t}$ the sets which can be used to cover element j in round t. m_t are the set of elements that must be covered in time t. The solution to the above linear program, is the optimal offline

solution for the problem. The dual problem is now:

$$\begin{aligned}
\text{maximize} \quad & \sum_{t=1}^{T}\sum_{j=1}^{m_t} a_{j,t} \\
\text{subject to} \quad & b_{i,t} \leq w_i, \ \forall t \geq 1 \ \& \ 1 \leq i \leq n \\
& b_{i,t+1} - b_{i,t} \leq c_{i,t} - \sum_{j|i \in S_{j,t}} a_{j,t}, \ \forall t \geq 1 \ \& \ 1 \leq i \leq n \\
& a_{j,t}, b_{i,t} \geq 0, \ \forall t \geq 1 \ \& \ i, j
\end{aligned}$$

The algorithm that we'll use to solve the problem in an online fashion shares ideas with aforementioned algorithms. But first, let's describe it.

Algorithm 4: Regularization Algorithm

1: parameters: $\epsilon > 0$, $\eta = ln(1 + \frac{n}{\epsilon})$
2: initialize: $y_0 = 0$,
3: **for** $t = 1 : T$ do
4: Observe the cost vector c_t and let P_t be the feasible set of solutions at time t, which are the covering constraints at time t.
5: Solve the following convex program to obtain y_t
6: $y_t = arg\min_{x \in P_t}\{c_t^T x_t + \frac{1}{\eta}\sum_{i=1}^{n} w_i((x_i + \frac{\epsilon}{n})\ln(\frac{x_i + \frac{\epsilon}{n}}{y_{i,t-1} + \frac{\epsilon}{n}}) - x_i)\}$
7: **end for**

Recall now the Hedge algorithm and Online Gradient Descent. We discussed their connection in Chapter 3. The algorithm is a gradient descent based rule, where we minimize the sum of the first-order Taylor approximation of the objective (here is the objective itself, since the objective is linear) penalized by the relative entropy function. The choice of this regularizer is not random. We saw that when we used relative entropy instead of l_2^2 as a regularizer in the hedge algorithm where one maintains a distribution over a set of elements we obtained a significantly better regret. This is the case here as well, since we maintain a distribution over the sets and the constraints in this case generalize the constraints in the experts setting. Moreover, recall that the balanced descent rule we discussed in the previous paragraph is essentially the same rule as gradient descent when the objective is a linear function as it is in this case. We now state the main theorem of this paragraph:

Theorem 4.9. *For every $\epsilon > 0$, Algorithm 4 provides a solution to the problem described by the primal program that is $O((1+\epsilon)\log(1+\frac{k}{\epsilon}))$ - competitive compared to the optimal offline solution, the optimal solution of the primal program.*

k is the maximal sparsity of the covering constraints: $k = \max\{|S_{j,t}| : 1 \leq t \leq T, 1 \leq j \leq m_t\}$. In the worst case, $k = n$ and for $\epsilon = 1$ we get an algorithm that is

$O(\log n)$-competitive where n is the number of sets and of course the dimension of the instance.

We'll now prove the main theorem. The idea of the proof is the following: Firstly, for each round t we express the solution of Algorithm 4 using the Karush-Kuhn-Tucker conditions. Using the optimal values given by these equations, we create a feasible dual solution. In particular, each variable of the dual of the linear program is assigned to a value from the optimal dual variables given by KKT. Then we show that the total hitting cost of the algorithm and the total moving (switching) cost of the algorithm which are expressed through the optimal primal variables given by the KKT can be bounded by the value of the feasible dual solution. Because of duality, we have bounded the total cost of the algorithm by the minimum value of the primal problem, which is the optimal offline cost and the competitive analysis is complete.

Proof. We'll start by stating the KKT conditions of the convex criterion of Algorithm 4 in time t. Let y_t be the optimal primal solution.

$$\sum_{i \in S_{j,t}} y_{i,t} - 1 \geq 0, \forall 1 \leq j \leq m_t \tag{4.1}$$

$$a_{j,t}(\sum_{i \in S_{j,t}} y_{i,t} - 1) = 0, \ \forall 1 \leq j \leq m_t \tag{4.2}$$

$$c_{i,t} + \frac{w_i}{\eta} \ln(\frac{y_{i,t} + \frac{\epsilon}{n}}{y_{i,t-1} + \frac{\epsilon}{n}}) - \sum_{j | i \in S_{j,t}} a_{j,t} \geq 0, \ \forall 1 \leq i \leq n, \tag{4.3}$$

$$y_{i,t}(c_{i,t} + \frac{w_i}{\eta} \ln(\frac{y_{i,t} + \frac{\epsilon}{n}}{y_{i,t-1} + \frac{\epsilon}{n}}) - \sum_{j | i \in S_{j,t}} a_{j,t}) = 0, \ \forall 1 \leq i \leq n, \tag{4.4}$$

Now, we assign values to the variables of the dual program. $a_{j,t}$ is assigned to the same of the KKT conditions, while $b_{i,t+1}$ is assigned the value $b_{i,t+1} = \frac{w_i}{\eta} \ln(\frac{1+\frac{\epsilon}{n}}{y_{i,t}+\frac{\epsilon}{n}})$. We now have to show that indeed these values are feasible. To this end we have $\forall t$ from inequality (4.3) that,

$$b_{i,t+1} - b_{i,t} = -\frac{w_i}{\eta} \ln(\frac{y_{i,t} + \frac{\epsilon}{n}}{y_{i,t-1} + \frac{\epsilon}{n}}) \leq c_{i,t} - \sum_{j | i \in S_{j,t}} a_{j,t}$$

Moreover, $0 \leq b_{i,t+1} = \frac{w_i}{\ln(1+\frac{n}{\epsilon})} \ln(\frac{1+\frac{\epsilon}{n}}{y_{i,t}+\frac{\epsilon}{n}}) \leq w_i$ which follows since $0 \leq y_{i,t} \leq 1$. Finally, of course we have $a_{j,t} \geq 0$ since $a_{j,t}$ is a Lagrangian dual variable. We proceed with bounding the hitting and the moving (switching) cost of the algorithm. Firstly, the

moving cost at time t is:

$$M_t = \eta \sum_{y_{i,t} > y_{i,t-1}} \frac{w_i}{\eta}(y_{i,t} - y_{i,t-1}) \tag{4.5}$$

$$\leq \eta \sum_{y_{i,t} > y_{i,t-1}} (y_{i,t} + \frac{\epsilon}{n})(\frac{w_i}{\eta} \ln(\frac{y_{i,t} + \frac{\epsilon}{n}}{y_{i,t-1} + \frac{\epsilon}{n}})) \tag{4.6}$$

$$= \eta \sum_{y_{i,t} > y_{i,t-1}} (y_{i,t} + \frac{\epsilon}{n})(\sum_{j | i \in S_{j,t}} a_{j,t} - c_{i,t}) \tag{4.7}$$

$$\leq \eta \sum_{i=1}^{n} (y_{i,t} + \frac{\epsilon}{n}) \sum_{j | i \in S_{j,t}} a_{j,t} = \eta \sum_{t=1}^{m_t} a_{j,t} (\sum_{i \in S_{j,t}} y_{i,t} + \frac{\epsilon}{n} |S_{j,t}|) \tag{4.8}$$

$$\leq \eta (1 + \frac{\epsilon k}{n}) \sum_{j=1}^{m_t} a_{j,t} = (1 + \frac{\epsilon k}{n}) D \tag{4.9}$$

Summing up we get that the total moving cost is at most $(1 + \frac{\epsilon k}{n})D$ where D is the value of the dual solution we created. Inequality (4.6) follows from the fact that $a - b \leq a \ln(\frac{a}{b}),\ \forall a, b > 0$. Equality (4.7) follows from condition (4.4) since $y_{i,t} \geq y_{i,t-1}$ implies $y_{i,t} > 0$. Inequality (4.8) follows since $c_{i,t}, y_{i,t}, a_{j,t}$ are nonnegative. Finally, inequality (4.9) follows from condition (4.2).

For the total service cost we have,

$$S = \sum_{t=1}^{T} \sum_{i=1}^{n} c_{i,t} y_{i,t} = \sum_{t=1}^{T} \sum_{j=1}^{m_t} a_{j,t} \sum_{i \in S_{j,t}} y_{i,t} - \frac{1}{\eta} \sum_{t=1}^{T} \sum_{i=1}^{n} w_i y_{i,t} \ln(\frac{y_{i,t} + \frac{\epsilon}{n}}{y_{i,t-1} + \frac{\epsilon}{n}}) \tag{4.10}$$

$$= \sum_{t=1}^{T} \sum_{j=1}^{m_t} a_{j,t} - \frac{1}{\eta} \sum_{i=1}^{n} \{\sum_{t=1}^{T} (y_{i,t} + \frac{\epsilon}{n}) \ln(\frac{y_{i,t} + \frac{\epsilon}{n}}{y_{i,t-1} + \frac{\epsilon}{n}}) - \frac{\epsilon}{n} \sum_{t=1}^{T} \ln(\frac{y_{i,t} + \frac{\epsilon}{n}}{y_{i,t-1} + \frac{\epsilon}{n}})\} \tag{4.11}$$

$$\leq D - \frac{1}{\eta} \sum_{i=1}^{n} w_i \{\sum_{t=1}^{T} (y_{i,t} + \frac{\epsilon}{n})) \ln(\frac{\sum_{t=1}^{T} (y_{i,t} + \frac{\epsilon}{n})}{\sum_{t=1}^{T} (y_{i,t-1} + \frac{\epsilon}{n})}) - \frac{\epsilon}{n} \ln(\frac{y_{i,T} + \frac{\epsilon}{n}}{y_{i,0} + \frac{\epsilon}{n}})\} \tag{4.12}$$

$$\leq D \tag{4.13}$$

Equality (4.10) follows from condition (4.4). Equality (4.11) follows from condition (4.2). Inequality (4.12) follows by telescopic sum and the log-sum inequality. Inequality (4.13) follows since $y_0 = 0$ and thus:

$$\frac{\epsilon}{n} \ln(\frac{y_{i,T} + \frac{\epsilon}{n}}{y_{i,0} + \frac{\epsilon}{n}}) = (y_{i,0} + \frac{\epsilon}{n}) \ln(\frac{y_{i,0} + \frac{\epsilon}{n}}{y_{i,T} + \frac{\epsilon}{n}}) \geq y_{i,0} - y_{i,T}$$

and:

$$\sum_{t=1}^{T} (y_{i,t} + \frac{\epsilon}{n}) \ln(\frac{\sum_{t=1}^{T} (y_{i,t} + \frac{\epsilon}{n})}{\sum_{t=1}^{T} (y_{i,t-1} + \frac{\epsilon}{n})}) \geq \sum_{t=1}^{T} (y_{i,t} + \frac{\epsilon}{n}) - \sum_{t=1}^{T} (y_{i,t-1} + \frac{\epsilon}{n}) = y_{i,T}$$

both because $a - b \leq a \ln(\frac{a}{b})$. Finally by choosing $\epsilon' = \frac{\epsilon n}{k}$ one concludes that the total cost of the algorithm is at most $1 + ((1 + \epsilon') \ln(1 + \frac{k}{\epsilon'}))$ the value of D and thus of the optimal offline solution. □

We provided the proof for the case of the covering constraints. If, in addition we are give a fixed set of precedence constraints of the form $x \leq y$ we can still provide the proof with some tweaks. These kind of constraints appear for example in facility location problems. The above idea was used to solve the problem of online version of the dynamic facility location problem ([2]) in [44] . Additionally, in [42] the authors provide an algorithm for the combinatorial problem of Online Shortest Path with Switching Cost using the regularization algorithm to provide a fractional solution.

4.5 Open problems

The study of OCO with switching cost is a relatively new research area and we believe that plenty of challenges remain to be addressed. We did a beyond worst case analysis using polyhedral functions to avoid the $O(\sqrt{d})$ lower bound when the switching cost is the l_2 norm. However, it would be interesting to find out if the same idea, to balance between the hitting and the switching cost can give us a $O(\sqrt{d})$ competitive ratio without further assumptions on the hitting cost functions. Moreover, an extremely interesting case is the one of the l_1 norm. For the l_1 norm we don't have a known lower bound and perhaps a constant competitive ratio can be achieved for a class of convex more general than α-polyhedral. Of course, notice that a constant competitive ratio for the case of the l_1 norm, due to norm equivalence, will lead to $O(\sqrt{d})$ competitive ratio for the l_2 norm.

An intuitively more difficult problem is the one where the switching cost function is not a norm and is a general convex function. Only very recently there has been work towards this direction. In [34] the authors study the case when the switching cost function is the squared euclidean norm. By making many assumptions including strong convexity, smoothness, Lipschitz continuity and a bound on the diameter of the feasible space (and of course a bound on the length of the optimal trajectory), they analyze and provide an algorithm that minimizes dynamic regret. Using in addition bounds of the form $f_t(x_t) \geq \epsilon$ for some $\epsilon > 0$ through the dynamic regret analysis they conclude to a competitive ratio of $1 + \frac{s}{\epsilon}$ where s depends on the parameters of the aforementioned assumptions. None of these assumptions were considered in the algorithms we previously analyzed and as the authors strongly suggest in [24] the techniques to acquire good competitive ratio bounds and dynamic regret bounds are different and that's why in their paper they provide two different algorithms for the two different metrics. We

believe that future work on this area should be to find out how the rule of balance performs in this case as well because of its success we saw in this chapter. We believe that this kind of approach will significantly improve the competitive ratio the authors provided in [34].

Bibliography

[1] Jacob Abernethy, Peter L. Bartlett, Niv Buchbinder, and Isabelle Stanton. A regularization approach to metrical task systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6331 LNAI, pages 270–284, 2010.

[2] Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. *ACM Trans. Algorithms*, 13(2):21:1–21:20, 2017.

[3] Lachlan L H Andrew, Siddharth Barman, Katrina Ligett, Minghong Lin, Adam Meyerson, Alan Roytman, and Adam Wierman. A Tale of Two Metrics: Simultaneous Bounds on Competitiveness and Regret. *The 26th Annual Conference on Learning Theory*, 30:741–763, 2013.

[4] Antonios Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, Kevin Schewior, and Michele Scquizzato. Chasing convex bodies and functions. In *LATIN 2016: Theoretical Informatics - 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings*, pages 68–81, 2016.

[5] Antonios Antoniadis and Kevin Schewior. A tight lower bound for online convex optimization with switching costs. *International Workshop on Approximation and Online Algorithms*, WAOA 2017: Approximation and Online Algorithms pp 164-175.

[6] Yossi Azar, Ilan Reuven Cohen, and Debmalya Panigrahi. Online covering with convex objectives and applications. *CoRR*, abs/1412.3507, 2014.

[7] Masoud Badiei, Na Li, and Adam Wierman. Online convex optimization with ramp constraints. In *54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 2015*, pages 6730–6736, 2015.

[8] N.a Bansal, A.b Gupta, R.c Krishnaswamy, K.d Pruhs, K.e Schewior, and C.f Stein. A 2-competitive algorithm for online convex optimization with switching costs. *Leibniz International Proceedings in Informatics, LIPIcs*, 40:96–109, 2015.

[9] Shai Ben-David and Shai Shalev-Shwartz. *Understanding Machine Learning: From Theory to Algorithms*. 2014.

[10] D. Bertsekas. *Nonlinear Programming*. 1999.

[11] O. Besbes, Y. Gur, and A. Zeevi. Non-stationary stochastic optimization. *Operations Research, no. 5, pp. 1227–1244*, vol. 63, 2015.

[12] Avrim Blum and Carl Burch. Online Learning and Metrical Task System problem.pdf. 1998.

[13] Allan Borodin and Ran El-Yaniv. Online Computation and Competitive Analysis, 1998.

[14] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM*, 39(4):745–763, 1992.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*, volume 25. 2010.

[16] Sébastien Bubeck. Convex Optimization: Algorithms and Complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[17] Sébastien Bubeck, Michael B. Cohen, James R. Lee, Yin Tat Lee, and Aleksander Madry. k-server via multiscale entropic regularization. *CoRR*, abs/1711.01085, 2017.

[18] N.a Buchbinder, S.b Chen, and J.b Naor. Competitive analysis via regularization. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 436–444, 2014.

[19] Niv Buchbinder. Unified Algorithms for Online Learning and Competitive Analysis. *Colt*, 23:1–18, 2012.

[20] Niv Buchbinder, Shahar Chen, Anupam Gupta, Viswanath Nagarajan, Joseph, and Naor. Online Packing and Covering Framework with Convex Objectives. pages 1–33, 2014.

[21] Niv Buchbinder and Joseph (Seffi) Naor. *The design of competitive online algorithms via a primal-dual approach*, volume 3. 2007.

[22] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. 2006.

[23] Niangjun Chen, Joshua Comden, Zhenhua Liu, Anshul Gandhi, and Adam Wierman. Using predictions in online optimization: Looking forward with an eye on the past. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science, Antibes Juan-Les-Pins, France, June 14-18, 2016*, pages 193–206, 2016.

[24] Niangjun Chen, Gautam Goel, and Adam Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. *CoRR*, abs/1803.10366, 2018.

[25] Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM J. Comput.*, 32(6):1403–1422, June 2003.

[26] Joel Friedman and Nathan Linial. On convex body chasing. *Discrete and Computational Geometry*, page 9(3):293–321, Mar 1993.

[27] Anupam Gupta, Kunal Talwar, and Udi Wieder. Changing bases: Multistage optimization for matroids and matchings. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8572 LNCS(PART 1):563–575, 2014.

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

[29] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. In *Machine Learning*, volume 69, pages 169–192, 2007.

[30] Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. Online optimization : Competing with dynamic comparators. *CoRR*, abs/1501.06225, 2015.

[31] Lei Jiao, Antonia Maria Tulino, Jaime Llorca, Yue Jin, and Alessandra Sala. Smoothed online resource allocation in multi-tier distributed cloud networks. *IEEE/ACM Trans. Netw.*, 25(4):2556–2570, August 2017.

[32] Vinay Joseph and Gustavo de Veciana. Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs. In *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, pages 567–575, 2012.

[33] Seung-Jun Kim and Georgios B. Giannakis. Real-time electricity pricing for demand response using online convex optimization. In *IEEE PES Innovative Smart Grid Technologies Conference, ISGT 2014, Washington, DC, USA, February 19-22, 2014*, pages 1–5, 2014.

[34] Y. Li, G. Qu, and N. Li. Online Optimization with Predictions and Switching Costs: Fast Algorithms and the Fundamental Limit. *ArXiv e-prints*, January 2018.

[35] Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional data centers. *2011 Proceedings IEEE INFOCOM*, pages 1098–1106, 2011.

[36] Minghong Lin, Adam Wierman, Lachlan L.H. Andrew, and Eno Thereska. Online dynamic capacity provisioning in data centers. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2011*, pages 1159–1163, 2011.

[37] Minghong Lin, Adam Wierman, Alan Roytman, Adam Meyerson, and Lachlan L.H. Andrew. Online optimization with switching cost. *ACM SIGMETRICS Performance Evaluation Review*, 40(3):98, 2012.

[38] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H. Low, and Lachlan L. H. Andrew. Greening geographical load balancing. *IEEE/ACM Trans. Netw.*, 23(2):657–671, 2015.

[39] Tan Lu, Minghua Chen, and Lachlan L. H. Andrew. Simple and effective dynamic provisioning for power-proportional data centers. *IEEE Trans. Parallel Distrib. Syst.*, 24(6):1161–1171, 2013.

[40] Aryan Mokhtari, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro. Online Optimization in Dynamic Environments: Improved Regret Rates for Strongly Convex Problems. 2016.

[41] Shai Shalev-Shwartz. Online Learning and Online Convex Optimization. *Found. Trends Mach. Learn.*, 4(2):107–194, 2012.

[42] Isidoros Tziotis. Online shortest path with switching cost. *UOA thesis*, 2017.

[43] Kai Wang, Minghong Lin, Florin Ciucu, Adam Wierman, and Chuang Lin. Characterizing the impact of the workload on the value of dynamic resizing in data centers. *CoRR*, abs/1207.6295, 2012.

[44] Lydia Zakinthinou. Online facility location with switching costs. *UOA thesis*, 2017.

[45] Martin Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. *Machine Learning*, 20(February):421–422, 2003.

Περιεχόμενα

Εισαγωγή **1**

0.1 Άμεσοι Αλγόριθμοι 1

0.2 Κυρτή Βελτιστοποίηση 2

1 Άμεση Κυρτή Βελτιστοποίηση **3**

2 Άμεση Κυρτή Βελτιστοποίηση με κόστη μετάβασης **5**

Εισαγωγή

Στήν παρούσα εργασία, μελετάμε το πρόβλημα της Άμεσης Κυρτής βελτιστοποίησης (ΑΚΒ) με κόστη μετάβασης. Αυτό είναι ένα πρόβλημα που είναι συσχετισμένο με ενα τυπικό πρόβλημα της περιοχής της ΑΚΒ αλλά επίσης είναι επηρεασμένο και απο ένα πρόβλημα που θα συναντούσαμε στην περιοχή των άμεσων αλγορίθμων, όπως τα Μετρικά συστήματα εργασίας. Σε αυτό το άμεσο πρόβλημα, δοσμένου ενός κυρτού υποσυνόλου του $\mathbb{R}^n$, ο παίκτης σε κάθε γύρο δέχεται μια κυρτή συνάρτηση $f_t(x)$ και παίρνει μια απόφαση, ένα σημείο $x_t \in X$. Τώρα, εκτός απο το αντικειμενικό κόστος $f_t(x_t)$, ο παίκτης πληρώνει και ένα κόστος λόγω της μεταβολής των αποφασεών του μεταξύ συνεχόμενων γύρων, το κόστος μετάβασης, $\|x_t - x_{t-1}\|$. Φυσικά, ο στόχος του παίκτη είναι να ελαχιστοποιήσει το συνολικό αντικειμενικό και κόστος μετάβασης για όλους τους γύρους. Αυτή η σύνθεση λοιπόν είναι κοντά στο πρόβλημα της ΑΚΒ όπου και εκεί σε κάθε γύρο εμφανίζεται μια κυρτή συνάρτηση αλλά επίσης το πρόβλημα είναι μια όχι και τόσο εξιδεικευμένη περίπτωση των Μετρικών συστημάων εργασίας (ΜΣΕ). Στα ΜΣΕ, ο χώρος $\mathbb{R}^n$ αντικαθιστάται απο έναν τυχαίο μετρικό χώρο (είτε συνεχή, είτε διακριτό) και οι συναρτήσεις που εμφανίζονται σε κάθε γύρο είναι αυθαίρετες και όχι αναγκαία κυρτές.

0.1 Άμεσοι Αλγόριθμοι

Τα προβλήματα που μας ενδιαφέρουν σε αυτή τη διατριβή πρέπει να λυθούν με άμεσο τρόπο. Ο μή άμεσος τρόπος, η παραδοσιακή ρύθμιση στην ανάλυση και τον σχεδιασμό αλγορίθμων, προϋποθέτει ότι η πλήρης πληροφορία του προβλήματος είναι γνωστή στον σχεδιαστή από την αρχή. Σε αντίθεση με αυτή τη ρύθμιση, στούς άμεσους αλγορίθμους, στοχεύουμε στο σχεδιασμό αλγορίθμων όπου η είσοδος αποκαλύπτεται κομμάτι-κομμάτι. Ο αλγόριθμος πρέπει να ανταποκρίνεται αμέσως όταν φθάνουν νέες πληροφορίες χωρίς τη γνώση των μελλοντικών πληροφοριών. Επιπλέον, όταν λαμβάνεται κάποια απόφαση, δεν μπορεί να ανακληθεί.

Ορισμός 0.1. Έστω $c \geq 1$ ένας πραγματικός αριθμός. Ένας άμεσος αλγόριθμος λέγεται ότι είναι c-ανταγωνιστικός αν για κάθε είσοδο ενός προβλήματος ελαχιστοποίησης I, για κάθε ακολουθία εισόδων, εξάγει μια λύση κόστους το πολύ $c \cdot OPT\ (I) + \alpha$ όπου OPT

(I) είναι το κόστος μιας βέλτιστης λύσης όταν όλη η πληροφορία είναι γνωστή απο την αρχή και το a είναι σταθερό. Για $a \leq 0$ λέμε ότι ο αλγόριθμος είναι αυστηρά c -ανταγωνιστικός.

0.2 Κυρτή Βελτιστοποίηση

Για τους αλγορίθμους που θα αναλύσουμε, σε κάθε γύρο θα χρειαστεί να λύσουμε ένα πρόβλημα κυρτής βελτιστοποίησης. Ένα παράδειγμα προβλήματος κυρτής βελτιστοποίησης είναι η προβολή ενός σημείου y στον χώρο σε ένα κυρτό υποσύνολο του χώρου X. Δηλαδή το σημείο του συνόλου που απέχει την μικρότερη απόσταση απο το δοσμένο σημείο y. Εν γένει, ένα πρόβλημα κυρτής βελτιστοποίησης, έχει την εξής μορφή:

$$\begin{aligned} &\text{Ελαχιστοποίησε} \quad && f_0(x) \\ &\underset{x\in\mathbb{R}^n}{} \\ &\text{Υπο περιορισμούς:} \quad && g_i(x) \leq b_i,\ i = 1, \ldots, m. \end{aligned}$$

Όπου η συνάρτησεις $f_0(x)$ και $g_i(x),\ x = 1, 2, ..., m$ είναι κυρτές. Ο πιο θεμελιώδης αλγόριθμος για να λύσουμε τέτοιου είδους προβλήματα είναι η μέθοδος κλίσης.

Αλγόριθμος 1: Μέθοδος Κλίσης

1: **Είσοδος:** Κυρτή συνάρτηση f, Αριθμός επαναλήψεων T, Σύνολο αποφάσεων X, Αρχικό σημείου $x_1 \in X$, ακολουθία $\{\eta_t\}$
2: **Για** $t = 1 : T$:
3: Λετ $y_{t+1} = x_t - \eta_t \nabla f(x_t),\ x_{t+1} = \Pi_X(y_{t+1})$
4: **Επέστρεψε** x_{T+1}

Αυτή η μέθοδος μας υπόσχεται ένα φράγμα της τάξης του $O(\frac{1}{\sqrt{T}})$ στο σφάλμα της λύσης μετά απο T γύρους.

Θεώρημα 0.2. *Για G-Lipschitz κυρτές συναρτήσεις και διάμετρο του συνόλου αποφάσεων ίση με D, για τις αποφάσεις του Αλγορίθμου 1 ισχύει:*

$$f(\frac{1}{T}\sum_{t=1}^{T} x_t) - f(x^*) \leq \frac{DG}{\sqrt{T}}$$

Άμα η συναρτήση f, πληροί επιπλέον προυποθέσεις (π.χ φράζεται απο κάτω και απο πάνω απο τετραγωνικές συναρτήσεις) μπορούμε να μειώσουμε πάρα πολύ τον αριθμό των επαναλήψεων ώστε να πάρουμε σφάλμα ϵ στην λύση. Συγκεκριμένα, απο $O(\frac{1}{\epsilon^2})$, όπως παραπάνω, μπορούμε να επιτύχουμε αριθμό επαναλήψεων $O(\log \frac{1}{\epsilon})$

Κεφάλαιο 1

Άμεση Κυρτή Βελτιστοποίηση

Σε αυτό το κεφάλαιο θα ορίσουμε το πρόβλημα της Άμεσης Κυρτής Βελτιστοποίσης (ΑΚΒ) το οποίο είναι άρρηκτα συνδεδεμένο τόσο με το πρόβλημα της ΑΚΒ με κόστη μετάβασης, καθώς και με το πεδίο της άμεσης μάθησης. Πολλά προβλήματα άμεσης μάθηση (π.χ άμεση ανίχνευση ανυπιθύμητης αλληλογραφίας) είναι ειδικές περιπτώσης ενός προβλήματος ΑΚΒ.

- Σε κάθε γύρο $t = 1, 2, ..., T$, ο υπεύθυνος λήψης αποφάσεων αποφασίζει, επιλέγοντας ένα σημείο από ένα κυρτό σύνολο $x_t \in X \subseteq \mathbb{R}^n$
- Μετά την απόφαση, ο αντίπαλος αποκαλύπτει μια κυρτή συνάρτηση $f_t(x),\ t \in [T]$ και ο παίκτης έχει απώλεια (η κόστος) $f_t(x_t)$.
- Ο στόχος του υπεύθυνου λήψης αποφάσεων είναι η ελαχιστοποίηση της λύπης ($Regret$). Δηλαδή, σε κάθε γύρο, να επιλέξει μια στρατηγική x_t για να ελαχιστοποιήσει:

$$\text{Λύπη} = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in X} \sum_{t=1}^{T} f_t(x)$$

Ένας αλγόριθμος που εγγυάται μια υπογραμμική ως προς το Τ ($o(T)$) λύπη, ονομάζεται αλγόριθμος μη-λύπης. Οι αλγόριθμοι ΑΚΒ στοχεύουν στην εξεύρεση της καλύτερης σταθερής απόφασης x^* η οποία ελαχιστοποιεί το άθροισμα των συναρτήσεων $f_t,\ t \in [T]$. Το μοντέλο δέχεται επίσης την ακόλουθη ερμηνεία: Αρχικά, ο αντίπαλος επιλέγει μία συνάρτηση f και την σπάει στα τμήματα f_t, έτσι ώστε να ισχύει το ακόλουθο: $\sum_{t=1}^{T} f_t(x) = f$. Το σπάει με τον χειρότερο δυνατό τρόπο ώστε ο παίκτης να κάνει τις περισσότερες επαναλήψεις (γύρους), προκειμένου να συγκλίνει στην καλύτερη απόφαση εκ των υστέρων x^*. Υπάρχει ένας απλός αλγόριθμος που μας δίνει το καλύτερο δυνατό φράγμα στην Λύπη του παίκτη όταν δεν υποθέτουμε κάτι περαιτέρω για την φύση των κυρτών συναρτήσεων σε κάθε γύρο.

Αλγόριθμος 2: Άμεση μέθοδος κλίσης

1: **Είσοδος:** Αριθμός επαναλήψεων T, Σύνολο αποφάσεων X, Αρχικό σημείο $x_1 \in X$, Ακολουθία ρυθμών εκμάθησης $\{\eta_t\}$
2: **Για** $t = 1 : T$
3: Αποφάσισε x_t και δέξου κόστος $f_t(x_t)$
4: Κάνε βήμα: $y_{t+1} = x_t - \eta_t \nabla f(x_t)$
5: Πρόβαλε στο X: $x_{t+1} = \Pi_X(y_{t+1})$

Θεώρημα 1.1. *Ας υποθέσουμε ότι σε κάθε γύρο, το βήμα μάθησης δίνεται απο τον τύπο:* $\eta_t = \frac{D}{G\sqrt{T}}$ *όπου* D *είναι η διάμετρος του κυρτού συνόλου και* G *είναι ένα φράγμα στην τιμή της κλίσης των συναρτήσεων* $\|\nabla f(x)\| \leq G,\ \forall x \in X$. *Τότε θα ισχύει:*

$$\sum_{t=1}^{T} f_t(x_t) - \min_{x \in X} \sum_{t=1}^{T} f_t(x) \leq \frac{3}{2} GD\sqrt{T}$$

Επιπλέον, ενδιαφέρον παρουσιάζουν τα προβλήματα όπου ο παίκτης έχει να ανταγωνιστεί μια δυναμική στρατηγική και όχι μια στατική x^* όπως στην παραπάνω περίπτωση. Τώρα δηλαδή, ο παίκτης πρέπει να ελαχιστοποιήσει:

$$\text{Δυναμική Λύπη} = \sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x_t^*)$$

Το πρόβλημα αυτό, χωρίς περαιτέρω προυποθέσεις, δεν έχει νόημα διότι άμα οι συναρτήσεις f_t είναι ασυσχέτιστες μεταξύ τους, ο παίκτης παίζει τυφλά καθώς πρώτα αποφασίζει την στρατηγική του x_t^* και έπειτα παρατηρεί την συνάρτηση f_t. Μπορούμε να θεωρήσουμε την εξης ποσότητα (L_T) η οποία μας επιτρέπει να εφαρμόσουμε μια συσχέτιση μεταξύ των συναρτήσεων f_t.

$$L_T = \sum_{t=1}^{T} \left\| x_t^* - x_{t-1}^* \right\|$$

Η οποία μας δείχνει πόσο κοντά η μακριά είναι τα σημεία που ελαχιστοποιούνται οι συναρτήσεις f_t. Μεγάλες τιμές αυτής της παραμέτρου σημαίνει ότι βρίσκονται μακριά, και έτσι κάθε αλγόριθμος θα πρέπει αναγκαστικά να έχει μεγάλο κάτω φράγμα. Αν απο την άλλη είναι μικρή, τότε γνωρίζουμε ότι μια στρατηγική που ελαχιστοποιεί την συνάρτηση f_t θα είναι καλή ώστε να ελαχιστοποιήσει και την συνάρτηση f_{t+1}. Η μέθοδος της άμεσης κλίσης μπορεί να επιτύχει το φράγμα $O(1 + L_T)$ στην δυναμική λύπη του παίκτη όταν οι συναρτήσεις f_t έχουν γ-καλή κατάσταση.

Κεφάλαιο 2

Άμεση Κυρτή Βελτιστοποίηση με κόστη μετάβασης

Σε αυτό το κεφάλαιο θα εστιάσουμε στο πρόβλημα της Άμεσης Κυρτής Βελτιστοποίησης με κόστη μετάβασης. Θα ξεκινήσουμε με τον ορισμό του προβλήματος:

Του προβλήμα αποτελείται από ένα σταθερό χώρο απόφασης, ένα κυρτό σύνολο $X \in \mathbb{R}^n$ και μια ακολουθία μη αρνητικών κυρτών συναρτήσεων $f_t(x),\ t \in [T]$. Στον γύρο t, ο παίκτης παρατηρεί τη συνάρτηση $f_t(x)$ και επιλέγει ένα σημείο $x_t \in X$ που επιφέρει ένα κόστος υπηρεσίας $f_t(x_t)$ και ένα κόστος αλλαγής $\|x_t - x_{t-1}\|$. Υποθέτουμε ότι τόσο ο παίκτης όσο και ο αντίπαλος ξεκινούν από την αρχή των αξόνων. Το συνολικό κόστος του παίκτη είναι:

$$\sum_{t=1}^{T} f_t(x_t) + \|x_t - x_{t-1}\|$$

ενώ το βέλτιστο κόστος είναι:

$$\min_{\{x_t\}_{t=1}^T \in X} \sum_{t=1}^{T} f_t(x_t) + \|x_t - x_{t-1}\|$$

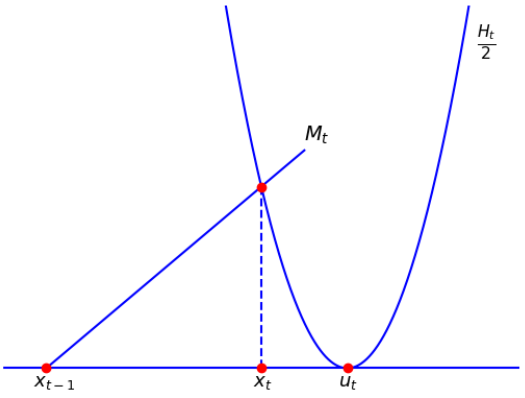
Θα αναφέρουμε τις αποφάσεις του αντιπάλου ως $\{x_t^*\}_{t=1}^T$. Παρατηρήστε ότι το πρόβλημα με όλη την γνώση έως τον χρόνο T είναι ένα κυρτό πρόβλημα βελτιστοποίησης και μπορεί να λυθεί αποδοτικά. Επίσης, το πρόβλημα μπορεί να δηλωθεί ως ένα πρόβλημα που δεν έχει περιορισμούς όταν εξετάζουμε την επέκταση της συνάρτησης f_t στο $\mathbb{R}^n$ που παίρνει την τιμή $f_t(x),\ x \in X$ και την τιμή $+\infty,\ x \notin X$. Ο επιγράφος της συνάρτησης είναι ίδιος με εκείνος της f_t και έτσι διατηρείται η κυρτότητα. Αυτό το γεγονός θα είναι χρήσιμο στην ανάλυση των αλγορίθμων που θα ακολουθήσουν.

Θα ξεκινήσουμε με έναν αλγόριθμο στην 1 διάσταση οποίος έχει σταθερό λόγο ανταγωνισμού ίσο με 3. Μάλιστα, αυτό ο λόγος ανταγωνισμού είναι και ο καλύτερος που μπορεί

να επιτευχθεί. Η ιδέα του αλγορίθμου είναι να πάρει μια απόφαση x_t έτσι ώστε να εξισορροπήσει το αντικειμενικό κόστος και το κόστος μετάβασης στον γύρο t. Σε αυτό το σημείο να σημειωθεί ότι ένας αλγόριθμος που θα αποφάσιζε ώστε να ελαχιστοποιήσει το άθροισμα των δύο κόστων στο γύρο T έχει κάτω φράγμα ίσο με $\Omega(T)$.

Αλγόριθμος 3: Άμεση μέθοδος εξισορρόπησης στην 1 διάσταση

1: **Για** $t = 1 : T$:
2: Έστω x_m οτι ελαχιστοποιεί την $f_t(x)$
3: Κινήσου πρός το x_m μέχρι να φτάσεις σε ένα σημείο x έτσι ώστε $M_t = \frac{H_t}{2}$ η να φτάσεις στο σημείο x_m
4: Αποφάσισε x_t



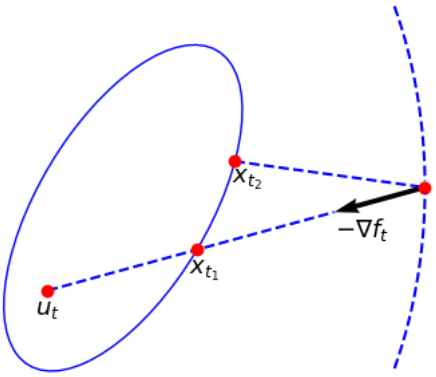
Σχημα 2.1: Γεωμετρική αναπαράσταση του Αλγορίθμου 1 για $f_t(u_t) = 0$

Θεώρημα 2.1. *Ο Αλγόριθμος 3 είναι 3-ανταγωνιστικός για το πρόβλημα της ΑΚΒ με κόστη μετάβασης στην 1 διάσταση.*

Ενδιαφέρον φυσικά παρουσιάζει το πρόβλημα σε διάσταση μεγαλύτερη του 1. Παρολαυτά σε αυτήν την περίπτωση δεν μπορούμε να επιτύχουμε ένα σταθερό λόγο ανταγωνισμού. Αυτό μας το δίνει το παρακάτω θεώρημα:

Θεώρημα 2.2. *Κάθε αλγόριθμος για το πρόβλημα της ΑΚΒ με κόστη μετάβασης έχει ανταγωνιστικό λόγο $\Omega(\sqrt{d})$ όταν το κόστος μετάβασης είναι η l_2 νόρμα.*

Μέχρι και σήμερα, παραμένει ανοικτό πρόβλημα, αν μπορούμε να επιτύχουμε έναν λόγο ανταγωνισμού $O(\sqrt{d})$ στην γενική περίπτωση και εν γένει δεν υπάρχουν αποτελέσματα για το πρόβλημα της ΑΚΒ με κόστη μετάβασης για γενικές κυρτές συναρτήσεις. Για αυτόν τον λόγο, για να επιτύχουμε έναν σταθερό ανταγωνιστικό λόγο πρέπει να εστιάσουμε την προσοχή μας σε μια συγκεκριμένη κλάση κυρτών συναρτήσεων. Το άνω κάτω φράγμα βασίζεται



Σχημα 2.2: Γεωμετρική απεικόνιση της αποτυχίας μεθόδων που δεν παίρνουν υπόψιν την γεωμετρία της ισουψής της συνάρτησης f_t στο x_t. Έχουμε $H_t(x_{t_1}) = H_t(x_{t_2})$ αλλά $M_t(x_{t_1}) > M_t(x_{t_2})$

σε συναρτήσεις που ελαχιστοποιούνται σε ένα κυρτό σύνολο X (π.χ ένα υπερεπίπεδο) και όχι ένα σημείο του χώρου. Έτσι μπορούμε να θεωρήσουμε συναρτήσεις που αυξάνονται τουλάχιστον γραμμικά πέρα απο το σημείο ελαχιστοποίησης, δηλαδή συναρτήσεις της μορφής: $f_t(x) = g_t(x) + a\,\|x - u_t\|$ όπου g_t μη αρνητική και έχει ελαχιστοποιητή το σημείο u_t. Για τέτοιου είδους συναρτήσεις, μπορούμε να επιτύχουμε έναν σταθερό λόγο ανταγωνισμού, που δεν εξαρτάται απο την διάσταση της εισόδου.

Ο αλγόριθμος που επιτυγχάνει έναν σταθερό λόγο ανταγωνισμού είναι φυσική επέκταση του αλγορίθμου στην 1 διάσταση. Συγκεκριμένα στην 1 διάσταση ο αλγόριθμος μπορεί να εκφραστεί ώς εξής: Θεωρούμε το σύνολο $\{x \in \mathbb{R}^n : f_t(x) \leq l\}$ το οποίο είναι ένα διάστημα στην 1 διάσταση. Αντι να μετακινηθούμε απο το x_{t-1} έως το u_t ώστε να επιτευχθεί η ισορροπία, αυξάνουμε συνεχώς το l απο $f_t(u_t)$ έως $f_t(x_{t-1})$ και προβάλλουμε το σημείο x_{t-1} στο διάστημα μέχρι να επιτευχθεί η ισορροπία. Οι δύο μέθοδοι είναι ισοδύναμες. Αυτός ο αλγόριθμος γενικεύεται σε διαστάση μεγαλύτερη του 1 ώς εξής:

Αλγόριθμος 4: Άμεση μέθοδος εξισορρόπησης

Για $t = 1 : T$:
    Έστω u_t ότι ελαχιστοποιεί την f_t
    Αν $\|u_t - x_{t-1}\| < \beta f_t(u_t)$
        Θέσε $x_t = u_t$
    αλλιώς
        Ας είναι $x(l) = \Pi_{X_t^l}(x_{t-1})$. Αύξησε l από $f_t(u_t)$ μέχρι $\|x(l) - x_{t-1}\| = \beta l$
        όπου X_t^l είναι το σύνολο $\{x \in \mathbb{R}^n : f_t(x) \leq l\}$
        Αποφάσισε $x_t = x(l)$

Θεώρημα 2.3. *Ο Αλγόριθμος 4 για* $\beta = \frac{1}{2} + \frac{1}{\alpha+2}$ *επιτυγχάνει ανταγωνιστικό λόγο* $3 + O(\frac{1}{\alpha})$ *για* α*-πολυεδρικές συναρτήσεις για το πρόβλημα της AKB με κόστη μετάβασης.*

Bibliography

[1] Jacob Abernethy, Peter L. Bartlett, Niv Buchbinder, and Isabelle Stanton. A regularization approach to metrical task systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6331 LNAI, pages 270–284, 2010.

[2] Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. *ACM Trans. Algorithms*, 13(2):21:1–21:20, 2017.

[3] Lachlan L H Andrew, Siddharth Barman, Katrina Ligett, Minghong Lin, Adam Meyerson, Alan Roytman, and Adam Wierman. A Tale of Two Metrics: Simultaneous Bounds on Competitiveness and Regret. *The 26th Annual Conference on Learning Theory*, 30:741–763, 2013.

[4] Antonios Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, Kevin Schewior, and Michele Scquizzato. Chasing convex bodies and functions. In *LATIN 2016: Theoretical Informatics - 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings*, pages 68–81, 2016.

[5] Antonios Antoniadis and Kevin Schewior. A tight lower bound for online convex optimization with switching costs. *International Workshop on Approximation and Online Algorithms*, WAOA 2017: Approximation and Online Algorithms pp 164-175.

[6] Yossi Azar, Ilan Reuven Cohen, and Debmalya Panigrahi. Online covering with convex objectives and applications. *CoRR*, abs/1412.3507, 2014.

[7] Masoud Badiei, Na Li, and Adam Wierman. Online convex optimization with ramp constraints. In *54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 2015*, pages 6730–6736, 2015.

[8] N.a Bansal, A.b Gupta, R.c Krishnaswamy, K.d Pruhs, K.e Schewior, and C.f Stein. A 2-competitive algorithm for online convex optimization with switching costs. *Leibniz International Proceedings in Informatics, LIPIcs*, 40:96–109, 2015.

[9] Shai Ben-David and Shai Shalev-Shwartz. *Understanding Machine Learning: From Theory to Algorithms*. 2014.

[10] D. Bertsekas. *Nonlinear Programming*. 1999.

[11] O. Besbes, Y. Gur, and A. Zeevi. Non-stationary stochastic optimization. *Operations Research, no. 5, pp. 1227–1244*, vol. 63, 2015.

[12] Avrim Blum and Carl Burch. Online Learning and Metrical Task System problem.pdf. 1998.

[13] Allan Borodin and Ran El-Yaniv. Online Computation and Competitive Analysis, 1998.

[14] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM*, 39(4):745–763, 1992.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*, volume 25. 2010.

[16] Sébastien Bubeck. Convex Optimization: Algorithms and Complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[17] Sébastien Bubeck, Michael B. Cohen, James R. Lee, Yin Tat Lee, and Aleksander Madry. k-server via multiscale entropic regularization. *CoRR*, abs/1711.01085, 2017.

[18] N.a Buchbinder, S.b Chen, and J.b Naor. Competitive analysis via regularization. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 436–444, 2014.

[19] Niv Buchbinder. Unified Algorithms for Online Learning and Competitive Analysis. *Colt*, 23:1–18, 2012.

[20] Niv Buchbinder, Shahar Chen, Anupam Gupta, Viswanath Nagarajan, Joseph, and Naor. Online Packing and Covering Framework with Convex Objectives. pages 1–33, 2014.

[21] Niv Buchbinder and Joseph (Seffi) Naor. *The design of competitive online algorithms via a primal-dual approach*, volume 3. 2007.

[22] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. 2006.

[23] Niangjun Chen, Joshua Comden, Zhenhua Liu, Anshul Gandhi, and Adam Wierman. Using predictions in online optimization: Looking forward with an eye on the past. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science, Antibes Juan-Les-Pins, France, June 14-18, 2016*, pages 193–206, 2016.

[24] Niangjun Chen, Gautam Goel, and Adam Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. *CoRR*, abs/1803.10366, 2018.

[25] Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM J. Comput.*, 32(6):1403–1422, June 2003.

[26] Joel Friedman and Nathan Linial. On convex body chasing. *Discrete and Computational Geometry*, page 9(3):293–321, Mar 1993.

[27] Anupam Gupta, Kunal Talwar, and Udi Wieder. Changing bases: Multistage optimization for matroids and matchings. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8572 LNCS(PART 1):563–575, 2014.

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

[29] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. In *Machine Learning*, volume 69, pages 169–192, 2007.

[30] Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. Online optimization : Competing with dynamic comparators. *CoRR*, abs/1501.06225, 2015.

[31] Lei Jiao, Antonia Maria Tulino, Jaime Llorca, Yue Jin, and Alessandra Sala. Smoothed online resource allocation in multi-tier distributed cloud networks. *IEEE/ACM Trans. Netw.*, 25(4):2556–2570, August 2017.

[32] Vinay Joseph and Gustavo de Veciana. Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs. In *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, pages 567–575, 2012.

[33] Seung-Jun Kim and Georgios B. Giannakis. Real-time electricity pricing for demand response using online convex optimization. In *IEEE PES Innovative Smart Grid Technologies Conference, ISGT 2014, Washington, DC, USA, February 19-22, 2014*, pages 1–5, 2014.

[34] Y. Li, G. Qu, and N. Li. Online Optimization with Predictions and Switching Costs: Fast Algorithms and the Fundamental Limit. *ArXiv e-prints*, January 2018.

[35] Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional data centers. *2011 Proceedings IEEE INFOCOM*, pages 1098–1106, 2011.

[36] Minghong Lin, Adam Wierman, Lachlan L.H. Andrew, and Eno Thereska. Online dynamic capacity provisioning in data centers. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2011*, pages 1159–1163, 2011.

[37] Minghong Lin, Adam Wierman, Alan Roytman, Adam Meyerson, and Lachlan L.H. Andrew. Online optimization with switching cost. *ACM SIGMETRICS Performance Evaluation Review*, 40(3):98, 2012.

[38] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H. Low, and Lachlan L. H. Andrew. Greening geographical load balancing. *IEEE/ACM Trans. Netw.*, 23(2):657–671, 2015.

[39] Tan Lu, Minghua Chen, and Lachlan L. H. Andrew. Simple and effective dynamic provisioning for power-proportional data centers. *IEEE Trans. Parallel Distrib. Syst.*, 24(6):1161–1171, 2013.

[40] Aryan Mokhtari, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro. Online Optimization in Dynamic Environments: Improved Regret Rates for Strongly Convex Problems. 2016.

[41] Shai Shalev-Shwartz. Online Learning and Online Convex Optimization. *Found. Trends Mach. Learn.*, 4(2):107–194, 2012.

[42] Isidoros Tziotis. Online shortest path with switching cost. *UOA thesis*, 2017.

[43] Kai Wang, Minghong Lin, Florin Ciucu, Adam Wierman, and Chuang Lin. Characterizing the impact of the workload on the value of dynamic resizing in data centers. *CoRR*, abs/1207.6295, 2012.

[44] Lydia Zakinthinou. Online facility location with switching costs. *UOA thesis*, 2017.

[45] Martin Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. *Machine Learning*, 20(February):421–422, 2003.