

A Modern Introduction to Online Learning

Francesco Orabona
Boston University
`francesco@orabona.com`

May 25, 2020

Contents

Abstract	vi
1 What is Online Learning?	1
2 Online Subgradient Descent	6
2.1 Online Learning with Convex Differentiable Losses	6
2.1.1 Convex Analysis Bits: Convexity	7
2.1.2 Online Gradient Descent	9
2.2 Online Subgradient Descent	11
2.2.1 Convex Analysis Bits: Subgradients	12
2.2.2 Analysis with Subgradients	13
2.3 From Convex Losses to Linear Losses	14
2.4 History Bits	14
2.5 Exercises	15
3 Online-to-Batch Conversion	16
3.1 Agnostic PAC Learning	18
3.2 Bits on Concentration Inequalities	19
3.3 From Regret to Agnostic PAC	20
3.4 History Bits	22
3.5 Exercises	22
4 Beyond \sqrt{T} Regret	23
4.1 Strong Convexity and Online Subgradient Descent	23
4.1.1 Convex Analysis Bits: Strong Convexity	23
4.1.2 Online Subgradient Descent for Strongly Convex Losses	24
4.2 Adaptive Algorithms: L^* bounds and AdaGrad	26
4.2.1 Adaptive Learning Rates for Online Subgradient Descent	26
4.2.2 Convex Analysis Bits: Smooth functions	27
4.2.3 L^* bounds	28
4.2.4 AdaGrad	28
4.3 History Bits	30
4.4 Exercises	31
5 Lower bounds for Online Linear Optimization	32
5.1 Lower bounds for Bounded OLO	32
5.2 Unconstrained Online Linear Optimization	33
5.2.1 Convex Analysis Bits: Fenchel Conjugate	33
5.2.2 Lower Bound for the Unconstrained Case	34
5.3 History Bits	36
5.4 Exercises	37

6	Online Mirror Descent	38
6.1	Subgradients are not Informative	38
6.2	Reinterpreting the Online Subgradient Descent Algorithm	38
6.3	Convex Analysis Bits: Bregman Divergence	40
6.4	Online Mirror Descent	41
6.4.1	The “Mirror” Interpretation	44
6.4.2	Yet Another Way to Write the Online Mirror Descent Update	45
6.5	Example of OMD: Exponentiated Gradient	46
6.6	Example of OMD: p -norm Algorithms	48
6.7	An Application of Online Mirror Descent: Learning with Expert Advice	49
6.8	History Bits	50
6.9	Exercises	51
7	Follow-The-Regularized-Leader	52
7.1	Follow-the-Regularized-Leader	52
7.1.1	Convex Analysis Bits: Properties of Strongly Convex Functions	53
7.1.2	An Explicit Regret Bound for Strongly Convex Functions	54
7.1.3	FTRL with Linearized Losses	55
7.1.4	FTRL with Linearized Losses Can Be Equivalent to OMD	57
7.2	Exponentiated Gradient with FTRL: No Need to know T	57
7.3	AdaHedge*	58
7.4	Composite Losses	61
7.5	FTRL with Strongly Convex Functions	63
7.6	Online Newton Step	64
7.7	Online Regression: Vovk-Azoury-Warmuth Forecaster	67
7.8	Optimistic FTRL	68
7.8.1	Regret that Depends on the Variance of the Subgradients	70
7.8.2	Online Convex Optimization with Gradual Variations	70
7.9	History Bits	72
7.10	Exercises	73
8	Online Linear Classification	74
8.1	Online Randomized Classifier	74
8.2	The Perceptron Algorithm	75
8.3	History Bits	78
8.4	Exercises	78
9	Parameter-free Online Linear Optimization	79
9.1	Coin-Betting Game	79
9.2	Parameter-free 1d OCO through Coin-Betting	81
9.2.1	KT as a 1d Online Convex Optimization Algorithm	83
9.3	Coordinate-wise Parameter-free OCO	85
9.4	Parameter-free in Any Norm	85
9.5	Combining Online Convex Optimization Algorithms	87
9.6	Reduction to Learning with Experts	88
9.6.1	A Betting Strategy that Loses at Most a Constant Fraction of Money	90
9.7	History Bits	92
9.8	Exercises	93

10 Multi-Armed Bandit	94
10.1 Adversarial Multi-Armed Bandit	94
10.1.1 Exponential-weight algorithm for Exploration and Exploitation: Exp3	96
10.1.2 Optimal Regret Using OMD with Tsallis Entropy	100
10.2 Stochastic Bandits	101
10.2.1 Concentration Inequalities Bits	102
10.2.2 Explore-Then-Commit Algorithm	103
10.2.3 Upper Confidence Bound Algorithm	105
10.3 History Bits	108
10.4 Exercises	108
A Appendix	110
A.1 The Lambert Function and Its Applications	110

List of Definitions

2.2	Definition (Convex Set)	7
2.3	Definition (Convex Function)	7
2.14	Definition (Proper Function)	12
2.15	Definition (Subgradient)	12
2.21	Definition (Lipschitz Function)	13
3.9	Definition (Agnostic-PAC-learnable)	19
3.10	Definition (Martingale)	19
4.1	Definition (Strongly Convex Function)	23
4.15	Definition (Dual Norm)	27
4.18	Definition (Smooth Function)	27
5.2	Definition (Closed Function)	33
5.4	Definition (Fenchel Conjugate)	33
6.2	Definition (Strictly Convex Function)	40
6.3	Definition (Bregman Divergence)	41
10.8	Definition (Subgaussian Random Variable)	102

List of Algorithms

2.1	Projected Online Gradient Descent	9
2.2	Projected Online Subgradient Descent	13
4.1	AdaGrad for Hyperrectangles	29
6.1	Online Mirror Descent	42
6.2	Exponentiated Gradient	47
6.3	Learning with Expert Advice through Randomization	50
7.1	Follow-the-Regularized-Leader	52
7.2	Follow-the-Regularized-Leader on Linearized Losses	56
7.3	AdaHedge	60
7.4	Follow-the-Regularized-Leader on “Quadratized” Losses	64
7.5	Online Newton Step	66
7.6	Vovk-Azoury-Warmuth Forecaster	67
7.7	Optimistic Follow-the-Regularized-Leader	69
8.1	Randomized Online Linear Classifier through FTRL	75
8.2	Perceptron Algorithm	76
9.1	Krichevsky-Trofimov bettor	80
9.2	Krichevsky-Trofimov OCO	83
9.3	OCO with Coordinate-Wise Krichevsky-Trofimov	85
9.4	Learning Magnitude and Direction Separately	86
10.1	Exponential Weights with Explicit Exploration for Multi-Armed Bandit	95
10.2	Exp3	96
10.3	INF Algorithm	100
10.4	Explore-Then-Commit Algorithm	103
10.5	Upper Confidence Bound Algorithm	105

Abstract

Disclaimer: This is work in progress, I plan to add more material and/or change/reorganize the content.

In this monograph, I introduce the basic concepts of Online Learning through a modern view of Online Convex Optimization. Here, online learning refers to the framework of regret minimization under worst-case assumptions. I present first-order and second-order algorithms for online learning with convex losses, in Euclidean and non-Euclidean settings. All the algorithms are clearly presented as instantiation of Online Mirror Descent or Follow-The-Regularized-Leader and their variants. Particular attention is given to the issue of tuning the parameters of the algorithms and learning in unbounded domains, through adaptive and parameter-free online learning algorithms. Non-convex losses are dealt through convex surrogate losses and through randomization. The bandit setting is also briefly discussed, touching on the problem of adversarial and stochastic multi-armed bandits. These notes do not require prior knowledge of convex analysis and all the required mathematical tools are rigorously explained. Moreover, all the proofs have been carefully chosen to be as simple and as short as possible.

I want to thank all the people that checked the proofs and reasonings in these notes. In particular, the students in my class that mercilessly pointed out my mistakes, Nicolás Campolongo that found all the typos in my formulas, and Jake Abernethy for the brainstorming on how to make the Tsallis proof even simpler. Other typos and errors were found by Alon Gonen, Gergely Imreh, Kwang-Sung Jun.

Chapter 1

What is Online Learning?

Imagine the following repeated game:

In each round $t = 1, \dots, T$

- An adversary choose a real number in $y_t \in [0, 1]$ and he keeps it secret;
- You try to guess the real number, choosing $x_t \in [0, 1]$;
- The adversary's number is revealed and you pay the squared difference $(x_t - y_t)^2$.

Basically, we want to guess a sequence of numbers as precisely as possible. To be a game, we now have to decide what is the “winning condition”. Let's see what makes sense to consider as winning condition.

First, let's simplify a bit the game. Let's assume that the adversary is drawing the numbers i.i.d. from some fixed distribution over $[0, 1]$. However, he is still free to decide which distribution at the beginning of the game. If we knew the distribution, we could just predict each round the mean of the distribution and in expectation we would pay $\sigma^2 T$, where σ^2 is the variance of the distribution. We cannot do better than that! However, given that we do not know the distribution, it is natural to benchmark our strategy with respect to the optimal one. That is, it is natural to measure the quantity

$$\mathbb{E}_Y \left[\sum_{t=1}^T (x_t - Y)^2 \right] - \sigma^2 T, \quad (1.1)$$

or equivalently considering the average

$$\frac{1}{T} \mathbb{E}_Y \left[\sum_{t=1}^T (x_t - Y)^2 \right] - \sigma^2. \quad (1.2)$$

Clearly these quantities are positive and they seem to be a good measure, because they are somehow normalized with respect to the “difficulty” of the numbers generated by the adversary, through the variance of the distribution. It is not the only possible choice to measure our “success”, but for sure it is a reasonable one. It would make sense to consider a strategy “successful” if the difference in (1.1) grows sublinearly over time and, equivalently, if the difference in (1.2) goes to zero as the number of rounds T goes to infinity. That is, on average on the number of rounds, we would like our algorithm to be able to approach the optimal performance.

Minimizing Regret. Given that we have converged to what it seems a good measure of success of the algorithm. Let's now rewrite (1.1) in an equivalent way

$$\mathbb{E} \left[\sum_{t=1}^T (x_t - Y)^2 \right] - \min_{x \in [0,1]} \mathbb{E} \left[\sum_{t=1}^T (x - Y)^2 \right].$$

Now, the last step: let's remove the assumption on how the data is generated, consider any arbitrary sequence of y_t , and keep using the same measure of success. Of course, we can remove the expectation because there is no stochasticity anymore. So, we get that we will win the game if

$$\text{Regret}_T := \sum_{t=1}^T (x_t - y_t)^2 - \min_{x \in [0,1]} \sum_{t=1}^T (x - y_t)^2$$

grows sublinearly with T . The quantity above is called the *Regret*, because it measures how much the algorithm regrets for not sticking on all the rounds to the optimal choice in hindsight. We will denote it by Regret_T . Our reasoning should provide sufficient justification for this metric, however in the following we will see that this also makes sense from both a convex optimization and machine learning point of view.

Let's now generalize the online game a bit more, considering that the algorithm outputs a vector in $x_t \in V \subseteq \mathbb{R}^d$ and it pays a loss $\ell_t : V \rightarrow \mathbb{R}$ that measures how good was the prediction of the algorithm in each round. The set V is called the *feasible set*. Also, let's consider an arbitrary predictor u in $V \subseteq \mathbb{R}^d$ and let's parameterize the regret with respect to it: $\text{Regret}_T(u)$. So, to summarize, Online Learning is nothing else than designing and analyzing algorithms to minimize the Regret over a sequence of loss functions with respect to an arbitrary competitor $u \in V \subseteq \mathbb{R}^d$:

$$\text{Regret}_T(u) := \sum_{t=1}^T \ell_t(x_t) - \sum_{t=1}^T \ell_t(u) .$$

This framework is pretty powerful, and it allows to reformulate a bunch of different problems in machine learning and optimization as similar games. More in general, with the regret framework we can analyze situations in which the data are not independent and identically distributed from a distribution, yet I would like to guarantee that the algorithm is "learning" something. For example, online learning can be used to analyze

- Click prediction problems;
- Routing on a network;
- Convergence to equilibrium of repeated games.

It can *also* be used to analyze stochastic algorithms, e.g., Stochastic Gradient Descent, but the adversarial nature of the analysis might give you suboptimal results. For example, it can be used to analyze momentum algorithms, but the adversarial nature of the losses essentially forces you to prove a convergence guarantee that treats the momentum term as a vanishing disturbance that does not help the algorithm in any way.

Let's now go back to our number guessing game and let's try a strategy to win it. Of course, this is one of the simplest example of online learning, without a real application. Yet, going through it we will uncover most of the key ingredients in online learning algorithms and their analysis.

A Winning Strategy. Can we win the number guessing name game? Note that we didn't assume anything on how the adversary is deciding the numbers. In fact, the numbers can be generated in any way, even in an adaptive way based on our strategy. Indeed, they can be chosen *adversarially*, that is explicitly trying to make us lose the game. This is why we call the mechanism generating the number the *adversary*.

The fact that the numbers are adversarially chosen means that we can immediately rule out any strategy based on any statistical modelling of the data. In fact, it cannot work because the moment we estimate something and act on our estimate, the adversary can immediately change the way he is generating the data, ruining us. So, we have to think about something else. Yet, many times online learning algorithms will look like classic ones from statistical estimation, even if they work for different reasons.

Now, let's try to design a strategy to make the regret provably sublinear in time, *regardless of how the adversary chooses the numbers*. The first thing to do is to take a look at the best strategy in hindsight, that is argmin of the second term of the regret. It should be immediate to see that

$$x_T^* := \operatorname{argmin}_{x \in [0,1]} \sum_{t=1}^T (x - y_t)^2 = \frac{1}{T} \sum_{t=1}^T y_t .$$

¹In same cases, we can make the game easier for the algorithm letting it choose the prediction from a set $W \supset V$.

Now, given that we do not know the future, for sure we cannot use x_T^* as our guess in each round. However, we do know the past, so a reasonable strategy in each round could be to output the best number over the past. Why such strategy would work? For sure, the reason why it could work is not because we expect the future to be like the past, because it is not true! Instead, we want to leverage the fact that the optimal guess should not change too much between rounds, so we can try to “track” it over time.

Hence, on each round t our strategy is to guess $x_t = x_{t-1}^* = \frac{1}{t-1} \sum_{i=1}^{t-1} y_i$. Such strategy is usually called *Follow-the-Leader* (FTL), because you are following what would have been the optimal thing to do on the past rounds (the Leader).

Let’s now try to show that indeed this strategy will allow us to win the game. Only for this simple example, we will prove its regret guarantee using first principles. Instead, in the following, we will introduce and use very general proof methods. First, we will need a small lemma.

Lemma 1.1. *Let $V \subseteq \mathbb{R}^d$ and $\ell_t : V \rightarrow \mathbb{R}$ an arbitrary sequence of loss functions. Denote by x_t^* a minimizer of the cumulative losses over the previous t rounds in V . Then, we have*

$$\sum_{t=1}^T \ell_t(x_t^*) \leq \sum_{t=1}^T \ell_t(x_T^*).$$

Proof. We prove it by induction over T . The base case is

$$\ell_1(x_1^*) \leq \ell_1(x_1^*),$$

that is trivially true. Now, for $T \geq 2$, we assume that $\sum_{t=1}^{T-1} \ell_t(x_t^*) \leq \sum_{t=1}^{T-1} \ell_t(x_{T-1}^*)$ is true and we must prove the stated inequality, that is

$$\sum_{t=1}^T \ell_t(x_t^*) \leq \sum_{t=1}^T \ell_t(x_T^*).$$

This inequality is equivalent to

$$\sum_{t=1}^{T-1} \ell_t(x_t^*) \leq \sum_{t=1}^{T-1} \ell_t(x_T^*), \tag{1.3}$$

where we removed the last element of the sums because they are the same. Now observe that

$$\sum_{t=1}^{T-1} \ell_t(x_t^*) \leq \sum_{t=1}^{T-1} \ell_t(x_{T-1}^*),$$

by induction hypothesis, and

$$\sum_{t=1}^{T-1} \ell_t(x_{T-1}^*) \leq \sum_{t=1}^{T-1} \ell_t(x_T^*)$$

because x_{T-1}^* is a minimizer of the left hand side in V and $x_T^* \in V$. Chaining these two inequalities, we have that (1.3) is true, and so the theorem is proven. \square

Basically, the above lemma quantifies the idea the knowing the future and being adaptive to it is typically better than not being adaptive to it.

With this lemma, we can now prove that the regret will grow sublinearly, in particular it will be *logarithmic* in time. Note that we will not prove that our strategy is minimax optimal, even if it is possible to show that the logarithmic dependency on time is unavoidable for this problem.

Theorem 1.2. *Let $y_t \in [0, 1]$ for $t = 1, \dots, T$ an arbitrary sequence of numbers. Let the algorithm’s output $x_t = x_{t-1}^* := \frac{1}{t-1} \sum_{i=1}^{t-1} y_i$. Then, we have*

$$\text{Regret}_T = \sum_{t=1}^T (x_t - y_t)^2 - \min_{x \in [0,1]} \sum_{t=1}^T (x - y_t)^2 \leq 4 + 4 \ln T.$$

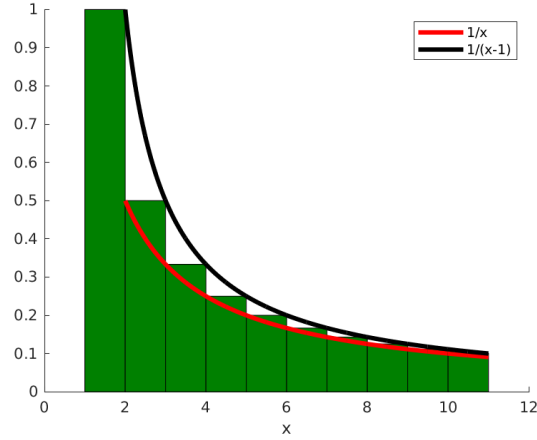


Figure 1.1: Upper bounding the sum with an integral.

Proof. We use Lemma 1.1 to upper bound the regret:

$$\sum_{t=1}^T (x_t - y_t)^2 - \min_{x \in [0,1]} \sum_{t=1}^T (x - y_t)^2 = \sum_{t=1}^T (x_{t-1}^* - y_t)^2 - \sum_{t=1}^T (x_T^* - y_t)^2 \leq \sum_{t=1}^T (x_{t-1}^* - y_t)^2 - \sum_{t=1}^T (x_t^* - y_t)^2.$$

Now, let's take a look at each difference in the sum in the last equation. We have that

$$\begin{aligned} (x_{t-1}^* - y_t)^2 - (x_t^* - y_t)^2 &= (x_{t-1}^*)^2 - 2y_t x_{t-1}^* - (x_t^*)^2 + 2y_t x_t^* \\ &= (x_{t-1}^* + x_t^* - 2y_t)(x_{t-1}^* - x_t^*) \\ &\leq |x_{t-1}^* + x_t^* - 2y_t| |x_{t-1}^* - x_t^*| \\ &\leq 2|x_{t-1}^* - x_t^*| \\ &= 2 \left| \frac{1}{t-1} \sum_{i=1}^{t-1} y_i - \frac{1}{t} \sum_{i=1}^t y_i \right| \\ &= 2 \left| \left(\frac{1}{t-1} - \frac{1}{t} \right) \sum_{i=1}^{t-1} y_i - \frac{y_t}{t} \right| \\ &\leq 2 \left| \frac{1}{t(t-1)} \sum_{i=1}^{t-1} y_i \right| + \frac{2|y_t|}{t} \\ &\leq \frac{2}{t} + \frac{2|y_t|}{t} \\ &\leq \frac{4}{t}. \end{aligned}$$

Hence, overall we have

$$\sum_{t=1}^T (x_t - y_t)^2 - \min_{x \in [0,1]} \sum_{t=1}^T (x - y_t)^2 \leq 4 \sum_{t=1}^T \frac{1}{t}.$$

To upper bound the last sum, observe that we are trying to find an upper bound to the green area in Figure 1.1. As you can see from the picture, it can be upper bounded by 1 plus the integral of $\frac{1}{t-1}$ from 2 to $T+1$. So, we have

$$\sum_{t=1}^T \frac{1}{t} \leq 1 + \int_2^{T+1} \frac{1}{t-1} dt = 1 + \ln T.$$

□

There are few things to stress on this strategy. The strategy does not have parameters to tune (e.g. learning rates, regularizers). Note that the presence of parameters does not make sense in online learning: We have only one stream of data and we cannot run our algorithm over it multiple times to select the best parameter! Also, it does not need to maintain a complete record of the past, but only a “summary” of it, through the running average. This gives a computationally efficient algorithm. When we design online learning algorithms, we will strive to achieve all these characteristics. Last thing that I want to stress is that the algorithm does not use gradients: Gradients are useful and we will use them a lot, but they do not constitute the entire world of online learning.

Before going on I want to remind you that, as seen above, this is different from the classic setting in statistical machine learning. So, for example, “overfitting” has absolutely no meaning here. Same for “generalization gap” and similar ideas linked to a training/testing scenario.

In the next chapters, we will introduce several algorithms for online optimization and one of them will be a strict generalization of the strategy we used in the example above.

Exercises

Problem 1.1. *Extend the previous algorithm and analysis to the case when the adversary selects a vector $\mathbf{y}_t \in \mathbb{R}^d$ such that $\|\mathbf{y}_t\|_2 \leq 1$, the algorithm guesses a vector $\mathbf{x}_t \in \mathbb{R}^d$, and the loss function is $\|\mathbf{x}_t - \mathbf{y}_t\|_2^2$. Show an upper bound to the regret logarithmic in T and that does not depend on d . Among the other things, you will probably need the Cauchy-Schwarz inequality: $\langle \mathbf{x}, \mathbf{y} \rangle \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2$.*

Chapter 2

Online Subgradient Descent

In this chapter, we will introduce the Online Subgradient Descent algorithm that is a generic online algorithm to solve online problems with convex losses. First, we will introduce Online Gradient Descent for convex differentiable functions.

2.1 Online Learning with Convex Differentiable Losses

To summarize what we said in the first chapter, let's define an online learning as the following general game

- For $t = 1, \dots, T$
 - Outputs $\mathbf{x}_t \in V \subseteq \mathbb{R}^d$
 - Receive $\ell_t : V \rightarrow \mathbb{R}$
 - Pay $\ell_t(\mathbf{x}_t)$
- End for

The aim of this game is to minimize the regret with respect to any competitor $\mathbf{u} \in V$:

$$\text{Regret}_T(\mathbf{u}) := \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}).$$

We also said that the way the losses ℓ_t are decided is adversarial. Now, without additional assumption we cannot hope to solve this problem. Hence, we have to understand what are the *reasonable* assumptions we can make. Typically, we will try to restrict the choice of the loss functions in some way. This is considered reasonable because most of the time we can decide the set from which the loss functions are picked. So, for example, we will consider only *convex* loss functions. However, convexity might not be enough, so we might restrict the class a bit more to, for example, Lipschitz convex functions. On the other hand, assuming to know something about the future is not considered a reasonable assumption, because we very rarely have any control on the future. In general, the stronger the assumptions the better will be the upper bound on the regret we can prove. The best algorithms we will see will guarantee a sublinear regret against the weakest assumption we can make, guaranteeing *at the same time* a smaller regret for *easy* adversaries.

It is also important to remember why minimizing the regret is a good objective: Given that we do not assume anything on how the adversary generates the loss functions, minimizing the regret is a good metric that takes into account the difficulty of the problem. If an online learning algorithm is able to guarantee a sublinear regret, it means that its performance on average will approach the performance of any fixed strategy. As said, we will see that in many situations if the adversary is “weak”, for example it is a fixed stochastic distribution over the loss functions, being prepared for the worst-case scenario will not preclude us to get the best guarantee anyway.

For a while, we will focus on the case that ℓ_t are convex, and this problem will be called **Online Convex Optimization** (OCO). Later, we will later see how to *convexify* some specific non-convex online problems.

Remark 2.1. *I will now introduce some math concepts. If you have a background in Convex Analysis, this will be easy stuff for you. On the other hand, if you never saw these things before they might look a bit scary. Let me tell you the right way to look at them: these are tools that will make our job easier. Without these tools, it would be basically impossible to design any online learning algorithm. And, no, it is not enough to test random algorithms on some machine learning dataset, because fixed datasets are not adversarial. Without a correct proof, you might never realize that your online algorithm fail on particular sequences of losses, as it happened to Adam [Reddi et al., 2018]. I promise you that once you understand the key mathematical concepts, online learning is actually easy.*

2.1.1 Convex Analysis Bits: Convexity

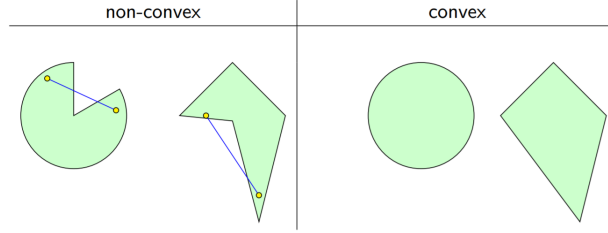


Figure 2.1: Convex and non-convex sets.

Definition 2.2 (Convex Set). $V \subset \mathbb{R}^d$ is **convex** if for any $\mathbf{x}, \mathbf{y} \in V$ and any $\lambda \in (0, 1)$, we have $\lambda\mathbf{x} + (1 - \lambda)\mathbf{y} \in V$.

In words, this means that *the set V has no holes*, see Figure 2.1.

We will make use of **extended-real-valued functions**, that is function that take value in $\mathbb{R} \cup \{-\infty, +\infty\}$. For f an extended-real-valued function on \mathbb{R}^d , its **domain** is the set $\text{dom } f = \{\mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}) < +\infty\}$.

Extended-real-valued functions allow us to easily consider constrained set and are a standard notation in Convex Optimization, see, e.g., Boyd and Vandenberghe [2004]. For example, if I want the predictions of the algorithm x_t and the competitor \mathbf{u} to be in a set $V \subset \mathbb{R}^d$, I can just add $i_V(\mathbf{x})$ to all the losses, where $i_V : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ is the **indicator function of the set V** defined as

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

In this way, the only way for the algorithm and for the competitor to suffer finite loss is to predict inside the set V . Also, extended-real-valued functions will make the use of Fenchel conjugates more direct, see Section 5.2.1.

Convex functions will be an essential ingredient in online learning.

Definition 2.3 (Convex Function). Let $f : \mathbb{R}^d \rightarrow [-\infty, +\infty]$. f is **convex** if the *epigraph* of the function, $\{(\mathbf{x}, y) \in \mathbb{R}^{d+1} | y \geq f(\mathbf{x})\}$, is convex.

We can see a visualization of this definition in Figure 2.2. Note that the definition implies that the domain of a convex function is convex. Also, observe that if $f : V \subseteq \mathbb{R}^d \rightarrow \mathbb{R}$ is convex, $f + i_V : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ is also convex. Note that $i_V(\mathbf{x})$ is convex iff V is convex, so each convex set is associated with a convex function.

The definition above gives rise to the following characterization for convex functions that do not assume the value $-\infty$.

Theorem 2.4 ([Rockafellar, 1970, Theorem 4.1]). Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ and $\text{dom } f$ is a convex set. Then f is convex iff, for any $0 < \lambda < 1$, we have

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

Example 2.5. The simplest example of convex functions are the affine function: $f(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle + b$.

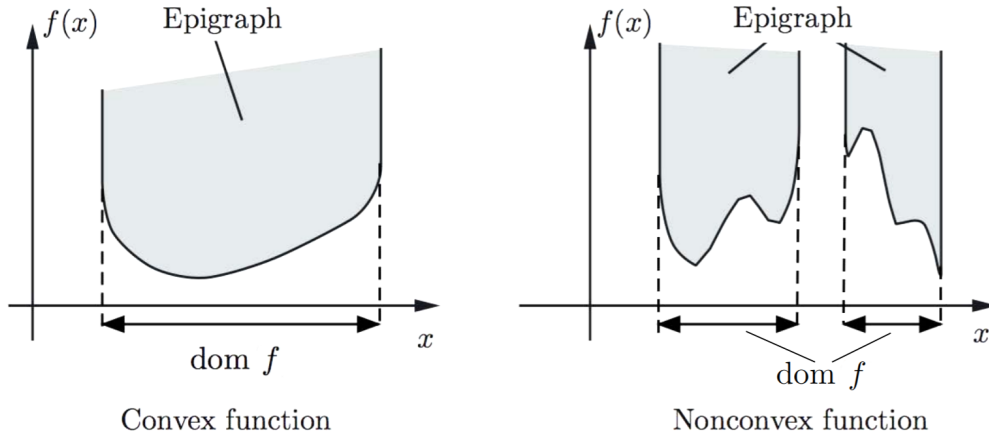


Figure 2.2: Convex and nonconvex functions.

Example 2.6. Norms are always convex, the proof is left as exercise.

How to recognize a convex function? In the most general case, you have to rely on the definition. However, most of the time we will recognize them as composed by operations that preserve the convexity. For example:

- f and g convex, then the linear combination with non-negative weights is also convex.
- The composition with an affine transformation preserves the convexity.
- Pointwise supremum of convex functions is convex.

The proofs are left as exercises.

A very important property of differentiable convex functions is that we can construct linear lower bound to the function.

Theorem 2.7 ([Rockafellar, 1970, Theorem 25.1 and Corollary 25.1.1]). Suppose $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ a convex function and let $\mathbf{x} \in \text{int dom } f$. If f is differentiable at \mathbf{x} then

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle, \mathbf{y} \in \mathbb{R}^d.$$

We will also use the first-order optimality condition for convex functions:

Theorem 2.8. Let V a convex non-empty set, $\mathbf{x}^* \in V$, and f a convex function, differentiable over an open set that contains V . Then $\mathbf{x}^* = \text{argmin}_{\mathbf{x} \in V} f(\mathbf{x})$ iff $\langle \nabla f(\mathbf{x}^*), \mathbf{y} - \mathbf{x}^* \rangle \geq 0, \forall \mathbf{y} \in V$.

In words, at the constrained minimum, the gradient makes an angle of 90° or less with all the feasible variations $\mathbf{y} - \mathbf{x}^*$, hence we cannot minimize more the function moving inside V .

Another critical property of convex functions is Jensen's inequality.

Theorem 2.9. Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ be a measurable convex function and \mathbf{x} be an \mathbb{R}^d -valued random element on some probability space such that $\mathbb{E}[\mathbf{x}]$ exists and $\mathbf{x} \in \text{dom } f$ with probability 1. Then

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

We can now see our first OCO algorithm in the case that the functions are convex and differentiable.

2.1.2 Online Gradient Descent

In the first chapter, we saw a simple strategy to obtain a logarithmic regret in the guessing game. The strategy was to use the best over the past, that is the *Follow-the-Leader* strategy. In formulas,

$$\mathbf{x}_t = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{t-1} \ell_i(\mathbf{x}),$$

and in the first round we can play any admissible point. One might wonder if this strategy always works, but the answer is negative!

Example 2.10 (Failure of FTL). *Let $V = [-1, 1]$ and consider the sequence of losses $\ell_t(x) = z_t x + i_V(x)$, where*

$$\begin{aligned} z_1 &= -0.5, \\ z_t &= 1, \quad t = 2, 4, \dots \\ z_t &= -1, \quad t = 3, 5, \dots \end{aligned}$$

Then, a part from the first round where the prediction of FTL is arbitrary in $[-1, 1]$, the predictions of FTL will be $x_t = 1$ for t even and $x_t = -1$ for t odd. The cumulative loss of the FTL algorithm will therefore be T while the cumulative loss of the fixed solution $u = 0$ is 0. Thus, the regret of FTL is T .

Hence, we will show an alternative strategy that guarantees sublinear regret for convex Lipschitz functions. Later, we will also prove that the dependency on T is optimal. The strategy is called Projected Online Gradient Descent, or just Online Gradient Descent, see Algorithm 2.1. It consists in updating the prediction of the algorithm at each time step moving in the negative direction of the gradient of the loss received and projecting back onto the feasible set. It is similar to Stochastic Gradient Descent, but it is not the same thing: here the loss functions are different at each step. We will later see that Online Gradient Descent can *also* be used as Stochastic Gradient Descent.

Algorithm 2.1 Projected Online Gradient Descent

Require: Non-empty closed convex set $V \subseteq \mathbb{R}^d$, $\mathbf{x}_1 \in V$, $\eta_1, \dots, \eta_T > 0$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Output \mathbf{x}_t
 - 3: Receive $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ and pay $\ell_t(\mathbf{x}_t)$
 - 4: Set $\mathbf{g}_t = \nabla \ell_t(\mathbf{x}_t)$
 - 5: $\mathbf{x}_{t+1} = \Pi_V(\mathbf{x}_t - \eta_t \mathbf{g}_t) = \operatorname{argmin}_{\mathbf{y} \in V} \|\mathbf{x}_t - \eta_t \mathbf{g}_t - \mathbf{y}\|_2$
 - 6: **end for**
-

First, we show the following two Lemmas. The first lemma proves that Euclidean projections always decrease the distance with points inside the set.

Proposition 2.11. *Let $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in V$, where $V \subseteq \mathbb{R}^d$ is a non-empty closed convex set and define $\Pi_V(\mathbf{x}) := \operatorname{argmin}_{\mathbf{y} \in V} \|\mathbf{x} - \mathbf{y}\|_2$. Then, $\|\Pi_V(\mathbf{x}) - \mathbf{y}\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_2$.*

Proof. From the optimality condition of Theorem 2.8, we obtain

$$\langle \Pi_V(\mathbf{x}) - \mathbf{x}, \mathbf{y} - \Pi_V(\mathbf{x}) \rangle \geq 0.$$

Therefore,

$$\begin{aligned} \|\mathbf{y} - \mathbf{x}\|_2^2 &= \|\mathbf{y} - \Pi_V(\mathbf{x}) + \Pi_V(\mathbf{x}) - \mathbf{x}\|_2^2 \\ &= \|\mathbf{y} - \Pi_V(\mathbf{x})\|_2^2 + 2\langle \mathbf{y} - \Pi_V(\mathbf{x}), \Pi_V(\mathbf{x}) - \mathbf{x} \rangle + \|\Pi_V(\mathbf{x}) - \mathbf{x}\|_2^2 \\ &\geq \|\mathbf{y} - \Pi_V(\mathbf{x})\|_2^2. \end{aligned}$$

□

The next lemma upper bounds the regret in one iteration of the algorithm.

Lemma 2.12. Let $V \subseteq \mathbb{R}^d$ a non-empty closed convex set and $\ell_t : V \rightarrow \mathbb{R}$ a convex function differentiable in a open set that contains V . Set $\mathbf{g}_t = \nabla \ell_t(\mathbf{x}_t)$. Then, $\forall \mathbf{u} \in V$, the following inequality holds

$$\eta_t(\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \eta_t \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq \frac{1}{2} \|\mathbf{x}_t - \mathbf{u}\|_2^2 - \frac{1}{2} \|\mathbf{x}_{t+1} - \mathbf{u}\|_2^2 + \frac{\eta_t^2}{2} \|\mathbf{g}_t\|_2^2.$$

Proof. From Proposition 2.11 and Theorem 2.7, we have that

$$\begin{aligned} \|\mathbf{x}_{t+1} - \mathbf{u}\|_2^2 - \|\mathbf{x}_t - \mathbf{u}\|_2^2 &\leq \|\mathbf{x}_t - \eta_t \mathbf{g}_t - \mathbf{u}\|_2^2 - \|\mathbf{x}_t - \mathbf{u}\|_2^2 \\ &= -2\eta_t \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle + \eta_t^2 \|\mathbf{g}_t\|_2^2 \\ &\leq -2\eta_t(\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) + \eta_t^2 \|\mathbf{g}_t\|_2^2. \end{aligned}$$

Reordering, we have the stated bound. \square

We can prove the following regret guarantee.

Theorem 2.13. Let $V \subseteq \mathbb{R}^d$ a non-empty closed convex set with diameter D , i.e. $\max_{\mathbf{x}, \mathbf{y} \in V} \|\mathbf{x} - \mathbf{y}\|_2 \leq D$. Let ℓ_1, \dots, ℓ_T an arbitrary sequence of convex functions $\ell_t : V \rightarrow \mathbb{R}$ differentiable in open sets containing V for $t = 1, \dots, T$. Pick any $\mathbf{x}_1 \in V$ assume $\eta_{t+1} \leq \eta_t$, $t = 1, \dots, T$. Then, $\forall \mathbf{u} \in V$, the following regret bound holds

$$\sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \frac{D^2}{2\eta_T} + \sum_{t=1}^T \frac{\eta_t}{2} \|\mathbf{g}_t\|_2^2.$$

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

$$\sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \frac{\|\mathbf{u} - \mathbf{x}_1\|_2^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_2^2.$$

Proof. Dividing the inequality in Lemma 2.12 by η_t and summing over $t = 1, \dots, T$, we have

$$\begin{aligned} \sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) &\leq \sum_{t=1}^T \left(\frac{1}{2\eta_t} \|\mathbf{x}_t - \mathbf{u}\|_2^2 - \frac{1}{2\eta_t} \|\mathbf{x}_{t+1} - \mathbf{u}\|_2^2 \right) + \sum_{t=1}^T \frac{\eta_t}{2} \|\mathbf{g}_t\|_2^2 \\ &= \frac{1}{2\eta_1} \|\mathbf{x}_1 - \mathbf{u}\|_2^2 - \frac{1}{2\eta_T} \|\mathbf{x}_{T+1} - \mathbf{u}\|_2^2 + \sum_{t=1}^{T-1} \left(\frac{1}{2\eta_{t+1}} - \frac{1}{2\eta_t} \right) \|\mathbf{x}_{t+1} - \mathbf{u}\|_2^2 + \sum_{t=1}^T \frac{\eta_t}{2} \|\mathbf{g}_t\|_2^2 \\ &\leq \frac{1}{2\eta_1} D^2 + D^2 \sum_{t=1}^{T-1} \left(\frac{1}{2\eta_{t+1}} - \frac{1}{2\eta_t} \right) + \sum_{t=1}^T \frac{\eta_t}{2} \|\mathbf{g}_t\|_2^2 \\ &= \frac{1}{2\eta_1} D^2 + D^2 \left(\frac{1}{2\eta_T} - \frac{1}{2\eta_1} \right) + \sum_{t=1}^T \frac{\eta_t}{2} \|\mathbf{g}_t\|_2^2 \\ &= \frac{D^2}{2\eta_T} + \sum_{t=1}^T \frac{\eta_t}{2} \|\mathbf{g}_t\|_2^2. \end{aligned}$$

In the same way, when the η_t is constant, we have

$$\begin{aligned} \sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) &\leq \sum_{t=1}^T \left(\frac{1}{2\eta} \|\mathbf{x}_t - \mathbf{u}\|_2^2 - \frac{1}{2\eta} \|\mathbf{x}_{t+1} - \mathbf{u}\|_2^2 \right) + \frac{\eta}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_2^2 \\ &= \frac{1}{2\eta} \|\mathbf{x}_1 - \mathbf{u}\|_2^2 - \frac{1}{2\eta} \|\mathbf{x}_{T+1} - \mathbf{u}\|_2^2 + \frac{\eta}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_2^2 \\ &\leq \frac{1}{2\eta} \|\mathbf{x}_1 - \mathbf{u}\|_2^2 + \frac{\eta}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_2^2. \end{aligned} \quad \square$$

We can immediately observe few things.

- If we want to use time-varying learning rates you need a bounded domain V . However, this assumption is false in most of the machine learning applications. However, in the stochastic setting you can still use a time-varying learning rate in SGD with an unbounded domain if you use a non-uniform averaging. We will see this in Chapter 3.
- Another important observation is that the regret bound helps us choosing the learning rates η_t . Indeed, it is the only guideline we have. Any other choice that is not justified by a regret analysis it is not justified at all.

As we said, the presence of parameters like the learning rates make no sense in online learning. So, we have to decide a strategy to set them. A simple choice is to find the constant learning rate that minimizes the bounds for a fixed number of iterations. We have to consider the expression

$$\frac{\|\mathbf{u} - \mathbf{x}_1\|_2^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_2^2$$

and minimize with respect to η . It is easy to see that the optimal η is $\frac{\|\mathbf{u} - \mathbf{x}_1\|_2}{\sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_2^2}}$, that would give the regret bound

$$\|\mathbf{u} - \mathbf{x}_1\|_2 \sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_2^2}.$$

However, we have a problem: in order to use this stepsize, we should know all the future gradients and the distance between the optimal solution and the initial point! This is clearly impossible: Remember that the adversary can choose the sequence of functions. Hence, it can observe your choice of learning rates and decide the sequence so that your learning rate is now the wrong one!

Indeed, it turns out that this kind of rate is completely impossible because it is ruled out by a lower bound. Yet, we will see that it is indeed possible to achieve very similar rates using *adaptive* (Section 4.2) and *parameter-free* algorithms (Chapter 9). For the moment, we can observe that we might be happy to minimize a looser upper bound. In particular, assume that the norm of the gradients is bounded by L , that is $\|\mathbf{g}_t\|_2 \leq L$. Also, assuming a bounded diameter, we can upper bound $\|\mathbf{u} - \mathbf{x}_1\|$ by D . Hence, we have

$$\eta^* = \operatorname{argmin}_{\eta} \frac{D^2}{2\eta} + \frac{\eta L^2 T}{2} = \frac{D}{L\sqrt{T}},$$

that gives a regret bound of

$$DL\sqrt{T}. \tag{2.1}$$

So, indeed the regret is sublinear in time.

In the next Section, we will see how to remove the differentiability assumption through the use of *subgradients*.

2.2 Online Subgradient Descent

In the previous section, we have introduced Projected Online Gradient Descent. However, the differentiability assumption for the ℓ_t is quite strong. What happens when the losses are convex but not differentiable? For example $\ell_t(x) = |x - 10|$. Note that this situation is more common than one would think. For example, the hinge loss, $\ell_t(\mathbf{w}) = \max(1 - y\langle \mathbf{w}, \mathbf{x} \rangle, 0)$, and the ReLU activation function used in neural networks, $\ell_t(x) = \max(x, 0)$, are not differentiable. It turns out that we can just use Online Gradient Descent, substituting the *subgradients* to the gradients. For this, we need some more convex analysis!

2.2.1 Convex Analysis Bits: Subgradients

First, we need a technical definition.

Definition 2.14 (Proper Function). *If a function f is nowhere $-\infty$ and finite somewhere, then f is called **proper**.*

In this book, we are mainly interested in convex proper functions, that basically better conform to our intuition of what a convex function looks like.

Let's first define formally what is a subgradient.

Definition 2.15 (Subgradient). *For a proper function $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$, we define a **subgradient** of f in $\mathbf{x} \in \mathbb{R}^d$ as a vector $\mathbf{g} \in \mathbb{R}^d$ that satisfies*

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

Basically, a subgradient of f in \mathbf{x} is any vector \mathbf{g} that allows us to construct a linear lower bound to f . Note that the subgradient is not unique, so we denote the *set* of subgradients of f in \mathbf{x} by $\partial f(\mathbf{x})$, called **subdifferential of f at \mathbf{x}** .

Observe that if f is proper and convex, then $\partial f(\mathbf{x})$ is empty for $\mathbf{x} \notin \text{dom } f$, because the inequality cannot be satisfied when $f(\mathbf{x}) = +\infty$. Also, the domain of ∂f , denoted by $\text{dom } \partial f$, is the set of all $\mathbf{x} \in \mathbb{R}^d$ such that $\partial f(\mathbf{x})$ is nonempty; it is a subset of $\text{dom } f$. A proper convex function f is always subdifferentiable in $\text{int dom } f$ [Rockafellar, 1970, Theorem 23.4].

If the function is convex, differentiable in \mathbf{x} , and f is finite in \mathbf{x} , we have that the subdifferential is composed by a unique element equal to $\nabla f(\mathbf{x})$ [Rockafellar, 1970, Theorem 25.1].

Also, we can also calculate the subgradient of sum of functions.

Theorem 2.16 ([Rockafellar, 1970, Theorem 23.8], [Bauschke and Combettes, 2011, Corollary 16.39]). *Let f_1, \dots, f_m be proper convex functions on \mathbb{R}^d , and $f = f_1 + \dots + f_m$. Then $\partial f(\mathbf{x}) \supset \partial f_1(\mathbf{x}) + \dots + \partial f_m(\mathbf{x}), \forall \mathbf{x}$. If $\text{dom } f_m \cap \bigcap_{i=1}^{m-1} \text{int dom } f_i \neq \{\}$, then actually $\partial f(\mathbf{x}) = \partial f_1(\mathbf{x}) + \dots + \partial f_m(\mathbf{x}), \forall \mathbf{x}$.*

Example 2.17. *Let $f(x) = |x|$, then the subdifferential set $\partial f(x)$ is*

$$\partial f(x) = \begin{cases} \{1\}, & x > 0, \\ [-1, 1], & x = 0, \\ \{-1\}, & x < 0. \end{cases}$$

Example 2.18. *Let's calculate the subgradient of the indicator function for a non-empty convex set $V \subset \mathbb{R}^d$. By definition, $\mathbf{g} \in \partial i_V(\mathbf{x})$ if*

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

*This condition implies that $\mathbf{x} \in V$ and $0 \geq \langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle, \forall \mathbf{y} \in V$ (because for $\mathbf{y} \notin V$ the inequality is always verified). The set of all \mathbf{g} that satisfies the above inequality is called the **normal cone of V at \mathbf{x}** . Note that the normal cone for any $\mathbf{x} \in \text{int } V = \{\mathbf{0}\}$ (Hint: take $\mathbf{y} = \mathbf{x} + \epsilon \mathbf{g}$). For example, for $V = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x}\|_2 \leq 1\}$, $\partial i_V(\mathbf{x}) = \{\alpha \mathbf{x} \mid \alpha \geq 0\}$ for all $\mathbf{x} : \|\mathbf{x}\| = 1$.*

Another useful theorem is to calculate the subdifferential of the pointwise maximum of convex functions.

Theorem 2.19 ([Bauschke and Combettes, 2011, Theorem 18.5]). *Let $(f_i)_{i \in I}$ be a finite set of convex functions from \mathbb{R}^d to $(-\infty, +\infty]$ and suppose $\mathbf{x} \in \bigcap_{i \in I} \text{dom } f_i$ and f_i continuous at \mathbf{x} . Set $F = \max_{i \in I} f_i$ and let $A(\mathbf{x}) = \{i \in I \mid f_i(\mathbf{x}) = F(\mathbf{x})\}$ the set of the active functions. Then*

$$\partial F(\mathbf{x}) = \text{conv} \bigcup_{i \in A(\mathbf{x})} \partial f_i(\mathbf{x}).$$

Example 2.20 (Subgradients of the Hinge loss). *Consider the loss $\ell(\mathbf{x}) = \max(1 - \langle \mathbf{z}, \mathbf{x} \rangle, 0)$ for $\mathbf{z} \in \mathbb{R}^d$. The subdifferential set is*

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

Definition 2.21 (Lipschitz Function). Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ is *L-Lipschitz* over a set V w.r.t a norm $\|\cdot\|$ if $|f(\mathbf{x}) - f(\mathbf{y})| \leq L\|\mathbf{x} - \mathbf{y}\|$, $\forall \mathbf{x}, \mathbf{y} \in V$.

We also have this handy result that upper bounds the norm of subgradients of convex Lipschitz functions.

Theorem 2.22. Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ proper and convex. Then, f is *L-Lipschitz* in $\text{int dom } f$ w.r.t. the L_2 norm iff for all $\mathbf{x} \in \text{int dom } f$ and $\mathbf{g} \in \partial f(\mathbf{x})$ we have $\|\mathbf{g}\|_2 \leq L$.

Proof. Assume f *L-Lipschitz*, then $|f(\mathbf{x}) - f(\mathbf{y})| \leq L\|\mathbf{x} - \mathbf{y}\|_2$, $\forall \mathbf{x}, \mathbf{y} \in \text{dom } f$. For $\epsilon > 0$ small enough $\mathbf{y} = \mathbf{x} + \epsilon \frac{\mathbf{g}}{\|\mathbf{g}\|_2} \in \text{int dom } f$, then

$$L\epsilon = L\|\mathbf{x} - \mathbf{y}\|_2 \geq |f(\mathbf{y}) - f(\mathbf{x})| \geq f(\mathbf{y}) - f(\mathbf{x}) \geq \langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle = \epsilon \|\mathbf{g}\|_2,$$

that implies that $\|\mathbf{g}\|_2 \leq L$.

For the other implication, the definition of subgradient and Cauchy-Schwartz inequalities gives us

$$f(\mathbf{x}) - f(\mathbf{y}) \leq \|\mathbf{g}\|_2 \|\mathbf{x} - \mathbf{y}\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2,$$

for any $\mathbf{x}, \mathbf{y} \in \text{int dom } f$. Taking $\mathbf{g} \in \partial f(\mathbf{y})$, we also get

$$f(\mathbf{y}) - f(\mathbf{x}) \leq L\|\mathbf{x} - \mathbf{y}\|_2,$$

that completes the proof. □

2.2.2 Analysis with Subgradients

As I promised you, with the proper mathematical tools, the analyzing online algorithms becomes easy. Indeed, switching from gradient to subgradient comes for free! In fact, our analysis of OGD with differentiable losses holds as is using subgradients instead of gradients. The reason is that the only property of the gradients that we used in the proof of Theorem 2.13 was that

$$\ell_t(\mathbf{x}) - \ell_t(\mathbf{u}) \leq \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle,$$

where $\mathbf{g}_t = \nabla \ell_t(\mathbf{x}_t)$. However, the exact same property holds when $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$. So, we can state the Online Subgradient descent algorithm in the following way, where the only difference is line 4.

Algorithm 2.2 Projected Online Subgradient Descent

Require: Non-empty closed convex set $V \subseteq \mathbb{R}^d$, $\mathbf{x}_1 \in V$, $\eta_1, \dots, \eta_T > 0$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Output \mathbf{x}_t
 - 3: Receive $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ and pay $\ell_t(\mathbf{x}_t)$
 - 4: Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$
 - 5: $\mathbf{x}_{t+1} = \Pi_V(\mathbf{x}_t - \eta_t \mathbf{g}_t) = \operatorname{argmin}_{\mathbf{y} \in V} \|\mathbf{x}_t - \eta_t \mathbf{g}_t - \mathbf{y}\|_2$
 - 6: **end for**
-

Also, the regret bounds we proved holds as well, just changing differentiability with subdifferentiability and gradients with subgradients. In particular, we have the following Lemma.

Lemma 2.23. Let $V \subseteq \mathbb{R}^d$ a non-empty closed convex set and $\ell_t : V \rightarrow \mathbb{R}$ a convex function subdifferentiable in a open set that contains V . Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$. Then, $\forall \mathbf{u} \in V$, the following inequality holds

$$\eta_t(\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \eta_t \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq \frac{1}{2} \|\mathbf{x}_t - \mathbf{u}\|_2^2 - \frac{1}{2} \|\mathbf{x}_{t+1} - \mathbf{u}\|_2^2 + \frac{\eta_t^2}{2} \|\mathbf{g}_t\|_2^2.$$

2.3 From Convex Losses to Linear Losses

Let's take a deeper look at this step

$$\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u}) \leq \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle, \forall \mathbf{u} \in \mathbb{R}^d.$$

And summing over time, we have

$$\sum_{t=1}^T \ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u}) \leq \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle, \forall \mathbf{u} \in \mathbb{R}^d.$$

Now, define the linear (and convex) losses $\tilde{\ell}_t(\mathbf{x}) := \langle \mathbf{g}_t, \mathbf{x} \rangle$, so we have

$$\sum_{t=1}^T \ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u}) \leq \sum_{t=1}^T \tilde{\ell}_t(\mathbf{x}_t) - \tilde{\ell}_t(\mathbf{u}).$$

This is more powerful than what it seems: We upper bounded the regret with respect to the convex losses ℓ_t with a regret with respect to another sequence of linear losses. This is important because it implies that we can build online algorithms that deal only with linear losses, and through the reduction above they can be seamlessly used as OCO algorithms! Note that this does not imply that this reduction is always optimal, it isn't! But, it allows us to easily construct optimal OCO algorithms in many interesting cases.

So, we will often consider just the problem of minimizing the linear regret

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle, \forall \mathbf{u} \in V \subseteq \mathbb{R}^d.$$

This problem is called **Online Linear Optimization (OLO)**.

Example 2.24. Consider the guessing game of the first class, we can solve easily it with Online Gradient Descent. Indeed, we just need to calculate the gradients, prove that they are bounded, and find a way to calculate the projection of a real number in $[0, 1]$. So, $\ell'_t(x) = 2(x - y_t)$, that is bounded for $x, y_t \in [0, 1]$. The projection on $[0, 1]$ is just $\Pi_{[0,1]}(x) = \min(\max(x, 0), 1)$. With the optimal learning rate, the resulting regret would be $O(\sqrt{T})$. This is worse than the one we found in the first class, showing that the reduction not always gives the best possible regret.

Example 2.25. Consider again the guessing game of the first class, but now change the loss function to the absolute loss of the difference: $\ell_t(x) = |x - y_t|$. Now we will need to use Online Subgradient Descent, because the functions are non-differentiable. We can easily see that

$$\partial \ell_t(x) = \begin{cases} \{1\}, & x > y_t \\ [-1, 1], & x = y_t \\ \{-1\}, & x < y_t. \end{cases}$$

Again, running Online Subgradient Descent with the optimal learning rate on this problem will give us immediately a regret of $O(\sqrt{T})$, without having to think of a particular strategy for it.

2.4 History Bits

The Projected Online Subgradient Descent with time-varying learning rate and the concept itself of Online Convex Optimization was introduced by Zinkevich [2003]. Note that, if we consider the optimization literature, the optimization people never restricted themselves to the bounded case.

2.5 Exercises

Problem 2.1. Prove that $\sum_{t=1}^T \frac{1}{\sqrt{t}} \leq 2\sqrt{T} - 1$.

Problem 2.2. Using the inequality in the previous exercise, prove that a learning rate $\eta_t \propto \frac{1}{\sqrt{t}}$ gives rise to a regret only a constant multiplicative factor worse than the one in (2.1).

Problem 2.3. Calculate the subdifferential set of the ϵ -insensitive loss: $f(x) = \max(|x - y| - \epsilon, 0)$. It is a loss used in regression problems where we do not want to penalize predictions x within $\pm\epsilon$ of the correct value y .

Problem 2.4. Using the definition of subgradient, find the subdifferential set of $f(\mathbf{x}) = \|\mathbf{x}\|_2$, $\mathbf{x} \in \mathbb{R}^d$.

Problem 2.5. Consider Projected Online Subgradient Descent for the Example 2.10 on the failure of Follow-the-Leader: Can we use it on that problem? Would it guarantee sublinear regret? How the behaviour of the algorithm would differ from FTL?

Chapter 3

Online-to-Batch Conversion

It is a good moment to take a break from online learning theory and see some application of online learning to other domains. For example, we may wonder what is the connection between online learning and stochastic optimization. Given that Projected Online (Sub)Gradient Descent looks basically the same as Projected Stochastic (Sub)Gradient Descent, they must have something in common. Indeed, we can show that, for example, we can reduce stochastic optimization of convex functions to OCO. Let's see how.

Theorem 3.1. *Let $F(\mathbf{x}) = \mathbb{E}[f(\mathbf{x}, \boldsymbol{\xi})]$ where the expectation is w.r.t. $\boldsymbol{\xi}$ drawn from ρ over some vector space X and $f : \mathbb{R}^d \times X \rightarrow (-\infty, +\infty]$ is convex in the first argument. Draw T samples $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_T$ i.i.d. from ρ and construct the sequence of losses $\ell_t(\mathbf{x}) = \alpha_t f(\mathbf{x}, \boldsymbol{\xi}_t)$, where $\alpha_t > 0$ are deterministic. Run any OCO algorithm over the losses ℓ_t , to construct the sequence of predictions $\mathbf{x}_1, \dots, \mathbf{x}_{T+1}$. Then, we have*

$$\mathbb{E} \left[F \left(\frac{1}{\sum_{t=1}^T \alpha_t} \sum_{t=1}^T \alpha_t \mathbf{x}_t \right) \right] \leq F(\mathbf{u}) + \frac{\mathbb{E}[\text{Regret}_T(\mathbf{u})]}{\sum_{t=1}^T \alpha_t}, \quad \forall \mathbf{u} \in \mathbb{R}^d,$$

where the expectation is with respect to $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_T$.

Proof. We first show that

$$\mathbb{E} \left[\sum_{t=1}^T \alpha_t F(\mathbf{x}_t) \right] = \mathbb{E} \left[\sum_{t=1}^T \ell_t(\mathbf{x}_t) \right]. \quad (3.1)$$

In fact, from the linearity of the expectation we have

$$\mathbb{E} \left[\sum_{t=1}^T \ell_t(\mathbf{x}_t) \right] = \sum_{t=1}^T \mathbb{E} [\ell_t(\mathbf{x}_t)].$$

Then, from the law of total expectation, we have

$$\mathbb{E} [\ell_t(\mathbf{x}_t)] = \mathbb{E} [\mathbb{E}[\ell_t(\mathbf{x}_t) | \boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_{t-1}]] = \mathbb{E} [\mathbb{E}[\alpha_t f(\mathbf{x}_t, \boldsymbol{\xi}_t) | \boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_{t-1}]] = \mathbb{E} [\alpha_t F(\mathbf{x}_t)],$$

where we used the fact that \mathbf{x}_t and α_t depend only on $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_{t-1}$. Hence, (3.1) is proved.

It remains only to use Jensen's inequality, using the fact that F is convex, to have

$$F \left(\frac{1}{\sum_{t=1}^T \alpha_t} \sum_{t=1}^T \alpha_t \mathbf{x}_t \right) \leq \frac{1}{\sum_{t=1}^T \alpha_t} \sum_{t=1}^T \alpha_t F(\mathbf{x}_t).$$

Dividing the regret by $\sum_{t=1}^T \alpha_t$ and using the above inequalities gives the stated theorem. \square

Let's see now some applications of this result: Let's see how to use the above theorem to transform Online Subgradient Descent in Stochastic Subgradient Descent to minimize the training error of a classifier.

Example 3.2. Consider a problem of binary classification, with inputs $\mathbf{z}_i \in \mathbb{R}^d$ and outputs $y_i \in \{-1, 1\}$. The loss function is the hinge loss: $f(\mathbf{x}, (\mathbf{z}, y)) = \max(1 - y\langle \mathbf{z}, \mathbf{x} \rangle, 0)$. Suppose that you want to minimize the training error over a training set of N samples, $\{(\mathbf{z}_i, y_i)\}_{i=1}^N$. Also, assume the maximum L_2 norm of the samples is R . That is, we want to minimize

$$\min_{\mathbf{x}} F(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^N \max(1 - y_i \langle \mathbf{z}_i, \mathbf{x} \rangle, 0).$$

Run the reduction described in Theorem 3.1 for T iterations using OGD. In each iteration, construct $\ell_t(\mathbf{x}) = \max(1 - y_t \langle \mathbf{z}_t, \mathbf{x} \rangle, 0)$ sampling a training point uniformly at random from 1 to N . Set $\mathbf{x}_1 = \mathbf{0}$ and $\eta = \frac{1}{R\sqrt{T}}$. We have that

$$\mathbb{E} \left[F \left(\frac{1}{T} \sum_{t=1}^T \mathbf{x}_t \right) \right] - F(\mathbf{x}^*) \leq R \frac{\|\mathbf{x}^*\|^2 + 1}{2\sqrt{T}}.$$

In words, we used an OCO algorithm to stochastically optimize a function, transforming the regret guarantee into a convergence rate guarantee.

In example last time, we had to use a constant learning rate to be able to minimize the training error over the entire space \mathbb{R}^d . In the next one, we will see a different approach, that allows to use a varying learning rate without the need of a bounded feasible set.

Example 3.3. Consider the same setting of the previous example, and let's change the way in which we construct the online losses. Now use $\ell_t(\mathbf{x}) = \frac{1}{R\sqrt{t}} \max(1 - y_t \langle \mathbf{z}_t, \mathbf{x} \rangle, 0)$ and step size $\eta = 1$. Hence, we have

$$\mathbb{E} \left[F \left(\sum_{t=1}^T \frac{1}{R\sqrt{t}} \mathbf{x}_t \right) \right] - F(\mathbf{x}^*) \leq \frac{\|\mathbf{x}^*\|_2^2}{2 \sum_{t=1}^T \frac{1}{R\sqrt{t}}} + \frac{1}{2 \sum_{t=1}^T \frac{1}{R\sqrt{t}}} \sum_{t=1}^T \frac{1}{t} \leq R \frac{\|\mathbf{x}^*\|_2^2 + 1 + \ln T}{4\sqrt{T} + 1 - 4},$$

where we used $\sum_{t=1}^T \frac{1}{\sqrt{t}} \geq 2\sqrt{T+1} - 2$.

I stressed the fact that the only meaningful way to define a regret is with respect to an arbitrary point in the feasible set. This is obvious in the case we consider unconstrained OLO, because the optimal competitor is unbounded. But, it is also true in unconstrained OCO. Let's see an example of this.

Example 3.4. Consider a problem of binary classification, with inputs $\mathbf{z}_i \in \mathbb{R}^d$ and outputs $y_i \in \{-1, 1\}$. The loss function is the logistic loss: $f(\mathbf{x}, (\mathbf{z}, y)) = \ln(1 + \exp(-y\langle \mathbf{z}, \mathbf{x} \rangle))$. Suppose that you want to minimize the training error over a training set of N samples, $\{(\mathbf{z}_i, y_i)\}_{i=1}^N$. Also, assume the maximum L_2 norm of the samples is R . That is, we want to minimize

$$\min_{\mathbf{x}} F(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^N \ln(1 + \exp(-y_i \langle \mathbf{z}_i, \mathbf{x} \rangle)).$$

So, run the reduction described in Theorem 3.1 for T iterations using OSD. In each iteration, construct $\ell_t(\mathbf{x}) = \ln(1 + \exp(-y_t \langle \mathbf{z}_t, \mathbf{x} \rangle))$ sampling a training point uniformly at random from 1 to N . Set $\mathbf{x}_1 = \mathbf{0}$ and $\eta = \frac{1}{R\sqrt{T}}$. We have that

$$\mathbb{E} \left[F \left(\frac{1}{T} \sum_{t=1}^T \mathbf{x}_t \right) \right] \leq \frac{R}{2\sqrt{T}} + \min_{\mathbf{u} \in \mathbb{R}^d} F(\mathbf{u}) + R \frac{\|\mathbf{u}\|^2}{2\sqrt{T}}.$$

In words, we will be $\frac{R}{2\sqrt{T}}$ away from the optimal value of regularized empirical risk minimization problem, where the weight of the regularization is $\frac{R}{2\sqrt{T}}$. Now, let's consider the case that the training set is linearly separable, this means that the infimum of F is 0 and the optimal solution does not exist, i.e., it has norm equal to infinity. So, any convergence guarantee that depends on \mathbf{x}^* would be vacuous. On the other hand, our guarantee above still makes perfectly sense.

Note that the above examples only deal with training error. However, there is a more interesting application of the online-to-batch conversion, that is to directly minimize the generalization error.

3.1 Agnostic PAC Learning

We now consider a different setting from what we have seen till now. We will assume that we have a prediction strategy $\phi_{\mathbf{x}}$ parametrized by a vector \mathbf{x} and we want to learn the relationship between an input \mathbf{z} and its associated label y . Moreover, we will assume that (\mathbf{z}, y) is drawn from a joint probability distribution ρ . Also, we are equipped with a loss function that measures how good is our prediction $\hat{y} = \phi_{\mathbf{x}}(\mathbf{z})$ compared to the true label y , that is $\ell(\hat{y}, y)$. So, learning the relationship can be cast as minimizing the expected loss of our predictor

$$\min_{\mathbf{x} \in V} \mathbb{E}_{(\mathbf{z}, y) \sim \rho} [\ell(\phi_{\mathbf{x}}(\mathbf{z}), y)] .$$

In machine learning terms, the object above is nothing else than the *test error* and our predictor.

Note that the above setting assumes labeled samples, but we can generalize it even more considering the *Vapnik's general setting of learning*, where we collapse the prediction function and the loss in a unique function. This allows, for example, to treat supervised and unsupervised learning in the same unified way. So, we want to minimize the *risk*

$$\min_{\mathbf{x} \in V} (\text{Risk}(\mathbf{x}) := \mathbb{E}_{\xi \sim \rho} [f(\mathbf{x}, \xi)]),$$

where ρ is an unknown distribution over D and $f : \mathbb{R}^d \times D \rightarrow \mathbb{R}$ is measurable w.r.t. the second argument. Also, the set \mathbb{F} of all predictors that can be expressed by vectors \mathbf{x} in V is called the *hypothesis class*.

Example 3.5. In a linear regression task where the loss is the square loss, we have $\xi = (\mathbf{z}, y) \in \mathbb{R}^d \times \mathbb{R}$ and $\phi_{\mathbf{x}}(\mathbf{z}) = \langle \mathbf{z}, \mathbf{x} \rangle$. Hence, $f(\mathbf{x}, \xi) = (\langle \mathbf{z}, \mathbf{x} \rangle - y)^2$.

Example 3.6. In linear binary classification where the loss is the hinge loss, we have $\xi = (\mathbf{z}, y) \in \mathbb{R}^d \times \{-1, 1\}$ and $\phi_{\mathbf{x}}(\mathbf{z}) = \langle \mathbf{z}, \mathbf{x} \rangle$. Hence, $f(\mathbf{x}, \xi) = \max(1 - y\langle \mathbf{z}, \mathbf{x} \rangle, 0)$.

Example 3.7. In binary classification with a neural network with the logistic loss, we have $\xi = (\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, 1\}$ and $\phi_{\mathbf{x}}$ is the network corresponding to the weights \mathbf{x} . Hence, $f(\mathbf{x}, \xi) = \ln(1 + \exp(-y\phi_{\mathbf{x}}(\mathbf{z})))$.

The key difficulty of the above problem is that we do not know the distribution ρ . Hence, there is no hope to exactly solve this problem. Instead, we are interested in understanding *what is the best we can do if we have access to T samples drawn i.i.d. from ρ* . More in details, we want to upper bound the *excess risk*

$$\text{Risk}(\mathbf{x}_T) - \min_{\mathbf{x}} \text{Risk}(\mathbf{x}),$$

where \mathbf{x}_T is a predictor that was *learned* using T samples.

It should be clear that this is just an optimization problem and we are interested in upper bounding the suboptimality gap. In this view, the objective of machine learning can be considered as a particular optimization problem.

Remark 3.8. Note that this is not the only way to approach the problem of learning. Indeed, the regret minimization model is an alternative model to learning. Moreover, another approach would be to try to estimate the distribution ρ and then solve the risk minimization problem, the approach usually taken in Statistics. No approach is superior to the other and each of them has its pros and cons.

Given that we have access to the distribution ρ through samples drawn from it, any procedure we might think to use to minimize the risk will be stochastic in nature. This means that we cannot assure a deterministic guarantee. Instead, we can try to prove that with high probability our minimization procedure will return a solution that is close to the minimizer of the risk. It is also intuitive that the precision and probability we can guarantee must depend on how many samples we draw from ρ .

Quantifying the dependency of precision and probability of failure on the number of samples used is the objective of the **Agnostic Probably Approximately Correct** (PAC) framework, where the keyword “agnostic” refers to the fact that we do not assume anything on the best possible predictor. More in details, given a precision parameter ϵ and a probability of failure δ , we are interested in characterizing the *sample complexity of the hypothesis class \mathbb{F}* that is defined as the number of samples T necessary to guarantee with probability at least $1 - \delta$ that the best learning algorithm using the hypothesis class \mathbb{F} outputs a solution \mathbf{x}_T that has an excess risk upper bounded by ϵ . Note that the

sample complexity does not depend on ρ , so it is a worst-case measure w.r.t. all the possible distributions. This makes sense if you think that we know nothing about the distribution ρ , so if your guarantee holds for the worst distribution it will also hold for any other distribution. Mathematically, we will say that the hypothesis class is agnostic PAC-learnable if such sample complexity function exists.

Definition 3.9 (Agnostic-PAC-learnable). *We will say that a function class $F = \{f(\mathbf{x}, \cdot) : \mathbf{x} \in \mathbb{R}^d\}$ is Agnostic-PAC-learnable if there exists an algorithm \mathcal{A} and a function $T(\epsilon, \delta)$ such that when \mathcal{A} is used with $T \geq T(\epsilon, \delta)$ samples drawn from ρ , with probability at least $1 - \delta$ the solution \mathbf{x}_T returned by the algorithm has excess risk at most ϵ .*

Note that the Agnostic PAC learning setting does not say what is the procedure we should follow to find such sample complexity. The approach most commonly used in machine learning to solve the learning problem is the so-called *Empirical Risk Minimization (ERM) problem*. It consist of drawing T samples i.i.d. from ρ and minimizing the *empirical risk*:

$$\widehat{\text{Risk}}(\mathbf{x}) := \min_{\mathbf{x} \in V} \frac{1}{T} \sum_{t=1}^T f(\mathbf{x}; \xi_t) .$$

In words, ERM is nothing else than minimize the error on a training set. However, in many interesting cases we can have that $\text{argmin}_{\mathbf{x} \in V} \frac{1}{T} \sum_{t=1}^T f(\mathbf{x}; \xi_t)$ can be very far from the true optimum $\text{argmin}_{\mathbf{x} \in V} \mathbb{E}[f(\mathbf{x}; \xi)]$, even with an infinite number of samples! So, we need to modify the ERM formulation in some way, e.g., using a *regularization* term or a Bayesian prior of \mathbf{x} , or find conditions under which ERM works.

The ERM approach is so widespread that machine learning itself is often wrongly identified with some kind of minimization of the training error. We now show that ERM is not the entire world of ML, showing that *the existence of a no-regret algorithm, that is an online learning algorithm with sublinear regret, guarantee Agnostic-PAC learnability*. More in details, we will show that an online algorithm with sublinear regret can be used to solve machine learning problems. This is not just a curiosity, for example this gives rise to computationally efficient parameter-free algorithms, that can be achieved through ERM only running a two-step procedure, i.e. running ERM with different parameters and selecting the best solution among them.

We already mentioned this possibility when we talked about the online-to-batch conversion, but this time we will strengthen it proving high probability guarantees rather than expectation ones.

So, we need some more bits on concentration inequalities.

3.2 Bits on Concentration Inequalities

We will use a concentration inequality to prove the high probability guarantee, but we will need to go beyond the sum of i.i.d.random variables. In particular, we will use the concept of *martingales*.

Definition 3.10 (Martingale). *A sequence of random variables X_1, X_2, \dots is called a **martingale** if for all $t \geq 1$ it satisfies:*

$$\mathbb{E}[|X_t|] < \infty, \quad \mathbb{E}[X_{t+1} | X_1, \dots, X_t] = X_t .$$

Example 3.11. *Consider a fair coin c_t and a betting algorithm that bets $|x_t|$ money on each round on the side of the coin equal to $\text{sign}(x_t)$. We win or lose money 1:1, so the total money we won up to round t is $Z_t = \sum_{i=1}^t c_i x_i$. Z_1, \dots, Z_t is a martingale. Indeed, we have*

$$\mathbb{E}[Z_t | Z_1, \dots, Z_{t-1}] = \mathbb{E}[Z_{t-1} + x_t c_t | Z_1, \dots, Z_{t-1}] = Z_{t-1} + \mathbb{E}[x_t c_t | Z_1, \dots, Z_{t-1}] = 0 .$$

For bounded martingales we can prove high probability guarantees as for bounded i.i.d. random variables. The following Theorem will be the key result we will need.

Theorem 3.12 (Hoeffding-Azuma inequality). *Let Z_1, \dots, Z_T be a martingale of T random variables that satisfy $|Z_t - Z_{t+1}| \leq B, t = 1, \dots, T - 1$ almost surely. Then, we have*

$$\mathbb{P}[Z_T - Z_0 \geq \epsilon] \leq \exp\left(-\frac{\epsilon^2}{2B^2 T}\right) .$$

Also, the same upper bounds hold on $\mathbb{P}[Z_0 - Z_T \geq \epsilon]$.

3.3 From Regret to Agnostic PAC

We now show how the online-to-batch conversion we introduced before gives us high probability guarantee for our machine learning problem.

Theorem 3.13. *Let $\text{Risk}(\mathbf{x}) = \mathbb{E}[f(\mathbf{x}, \boldsymbol{\xi})]$, where the expectation is w.r.t. $\boldsymbol{\xi}$ drawn from ρ with support over some vector space D and $f : \mathbb{R}^d \times D \rightarrow [0, 1]$. Draw T samples $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_T$ i.i.d. from ρ and construct the sequence of losses $\ell_t(\mathbf{x}) = f(\mathbf{x}, \boldsymbol{\xi}_t)$. Run any online learning algorithm over the losses ℓ_t , to construct the sequence of predictions $\mathbf{x}_1, \dots, \mathbf{x}_{T+1}$. Then, we have with probability at least $1 - \delta$, it holds that*

$$\frac{1}{T} \sum_{t=1}^T \text{Risk}(\mathbf{x}_t) \leq \inf_{\mathbf{u} \in \mathbb{R}^d} \text{Risk}(\mathbf{u}) + \frac{\text{Regret}_T(\mathbf{u})}{T} + 2\sqrt{\frac{2 \ln \frac{2}{\delta}}{T}}.$$

Proof. Define $Z_t = \sum_{i=1}^t (\text{Risk}(\mathbf{x}_i) - \ell_i(\mathbf{x}_i))$. We claim that Z_t is a martingale. In fact, we have

$$\mathbb{E}[\ell_t(\mathbf{x}_t) | \boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_{t-1}] = \mathbb{E}[f(\mathbf{x}_t, \boldsymbol{\xi}_t) | \boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_{t-1}] = \text{Risk}(\mathbf{x}_t),$$

where we used the fact that \mathbf{x}_t depends only on $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_{t-1}$. Hence, we have

$$\mathbb{E}[Z_{t+1} | Z_1, \dots, Z_t] = Z_t + \mathbb{E}[\text{Risk}(\mathbf{x}_{t+1}) - \ell_{t+1}(\mathbf{x}_{t+1}) | Z_1, \dots, Z_t] = Z_t,$$

that proves our claim.

Hence, using Theorem 3.12, we have

$$\mathbb{P} \left[\sum_{t=1}^T (\text{Risk}(\mathbf{x}_t) - \ell_t(\mathbf{x}_t)) \geq \epsilon \right] = \mathbb{P} [Z_T - Z_0 \geq \epsilon] \leq \exp \left(-\frac{\epsilon^2}{2T} \right).$$

This implies that, with probability at least $1 - \delta/2$, we have

$$\sum_{t=1}^T \text{Risk}(\mathbf{x}_t) \leq \sum_{t=1}^T \ell_t(\mathbf{x}_t) + \sqrt{2T \ln \frac{2}{\delta}}.$$

or equivalently

$$\frac{1}{T} \sum_{t=1}^T \text{Risk}(\mathbf{x}_t) \leq \frac{1}{T} \sum_{t=1}^T \ell_t(\mathbf{x}_t) + \sqrt{\frac{2 \ln \frac{2}{\delta}}{T}}.$$

We now use the definition of regret w.r.t. any \mathbf{u} , to have

$$\frac{1}{T} \sum_{t=1}^T \ell_t(\mathbf{x}_t) = \frac{\text{Regret}_T(\mathbf{u})}{T} + \frac{1}{T} \sum_{t=1}^T \ell_t(\mathbf{u}).$$

The last step is to upper bound with high probability $\frac{1}{T} \sum_{t=1}^T \ell_t(\mathbf{u})$ with $\text{Risk}(\mathbf{u})$. This is easier than the previous upper bound because \mathbf{u} is a fixed vector, so $\ell_t(\mathbf{u})$ are i.i.d. random variables, so for sure $Z_t = \sum_{i=1}^t (\text{Risk}(\mathbf{u}) - \ell_i(\mathbf{u}))$ forms a martingale. So, reasoning as above, we have that with probability at least $1 - \delta/2$ it holds that

$$\frac{1}{T} \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \text{Risk}(\mathbf{u}) + \sqrt{\frac{2 \ln \frac{2}{\delta}}{T}}.$$

Putting all together and using the union bound, we have the stated bound. \square

The theorem above upper bounds the average risk of the T predictors, while we are interested in producing a single predictor. If the risk is a convex function and V is convex, then we can lower bound the l.h.s. of the inequalities in the theorem with the risk evaluated on the average of the \mathbf{x}_t . That is

$$\text{Risk}\left(\frac{1}{T} \sum_{t=1}^T \mathbf{x}_t\right) \leq \frac{1}{T} \sum_{t=1}^T \text{Risk}(\mathbf{x}_t).$$

If the risk is not a convex function, we need a way to generate a single solution with small risk. One possibility is to construct a *stochastic classifier* that samples one of the \mathbf{x}_t with uniform probability and predicts with it. For this classifier, we immediately have

$$\text{Risk}(\{\mathbf{x}_1, \dots, \mathbf{x}_T\}) = \frac{1}{T} \sum_{t=1}^T \text{Risk}(\mathbf{x}_t),$$

where the expectation in the definition of the risk of the stochastic classifier is also with respect to the random index.

Yet another way, is to select among the T predictors, the one with the smallest risk. This works because the average is lower bounded by the minimum. This is easily achieved using $T/2$ samples for the online learning procedure and $T/2$ samples to generate a validation set to evaluate the solution and pick the best one. The following Theorem shows that selecting the predictor with the smallest empirical risk on a validation set will give us a predictor close to the best one with high probability.

Theorem 3.14. *We have a finite set of predictors $S = \{\mathbf{x}_1, \dots, \mathbf{x}_{|S|}\}$ and a dataset of T samples drawn i.i.d. from p . Denote by $\hat{\mathbf{x}} = \arg\min_{\mathbf{x} \in S} \widehat{\text{Risk}}(\mathbf{x})$. Then, with probability at least $1 - \delta$, we have*

$$\text{Risk}(\hat{\mathbf{x}}) \leq \min_{\mathbf{x} \in S} \text{Risk}(\mathbf{x}) + \sqrt{\frac{2 \ln(|S|/\delta)}{T}}.$$

Proof. We want to calculate the probability that the hypothesis that minimizes the validation error is far from the best hypothesis in the set. We cannot do it directly because we do not have the required independence to use a concentration. Instead, we will upper bound the probability that there exists at least one function whose empirical risk is far from the risk. So, we have

$$\mathbb{P}\left[\exists \mathbf{x} \in S : |\text{Risk}(\mathbf{x}) - \widehat{\text{Risk}}(\mathbf{x})| > \frac{\epsilon}{2}\right] \leq \sum_{i=1}^{|S|} \mathbb{P}\left[|\text{Risk}(\mathbf{x}_i) - \widehat{\text{Risk}}(\mathbf{x}_i)| > \frac{\epsilon}{2}\right] \leq |S| \exp\left(-\frac{\epsilon^2 T}{2}\right).$$

Hence, with probability at least $1 - \delta$, we have that for all $\mathbf{x} \in S$

$$|\text{Risk}(\mathbf{x}) - \widehat{\text{Risk}}(\mathbf{x})| \leq \sqrt{\frac{2 \ln(|S|/\delta)}{T}}.$$

We are now able to upper bound the risk of $\hat{\mathbf{x}}$, just using the fact that the above applies to $\hat{\mathbf{x}}$ too. So, we have

$$\text{Risk}(\hat{\mathbf{x}}) \leq \widehat{\text{Risk}}(\hat{\mathbf{x}}) + \epsilon/2 \leq \widehat{\text{Risk}}(\mathbf{x}^*) + \epsilon/2 \leq \text{Risk}(\mathbf{x}^*) + \epsilon,$$

where in the last inequality we used the fact that $\hat{\mathbf{x}}$ minimizes the empirical risk. \square

Using this theorem, we can use $T/2$ samples for the training and $T/2$ samples for the validation. Denoting by $\hat{\mathbf{x}}_T$ the predictor with the best empirical risk on the validation set among the $T/2$ generated during the online procedure, we have with probability at least $1 - 2\delta$ that

$$\text{Risk}(\hat{\mathbf{x}}_T) \leq \inf_{\mathbf{u} \in \mathbb{R}^d} \text{Risk}(\mathbf{u}) + \frac{\text{Regret}_T(\mathbf{u})}{T} + 3\sqrt{\frac{2 \ln(T/\delta)}{T}}.$$

It is important to note that with any of the above three methods to select one \mathbf{x}_t among the T generated by the online learning procedure, the sample complexity guarantee we get matches the one we would have obtained by ERM, up to polylogarithmic factors. In other words, there is nothing special about ERM compared to the online learning approach to statistical learning.

Another important point is that the above guarantee does not imply the existence of online learning algorithms with sublinear regret for any learning problem. It just says that, if it exists, it can be used in the statistical setting too.

3.4 History Bits

The specific shape of Theorem 3.1 is new, but I wouldn't be surprised if it appeared somewhere in the literature. In particular, the uniform averaging is from Cesa-Bianchi et al. [2004], but was proposed for the absolute loss in Blum et al. [1999]. The non-uniform averaging, that we will use next time, is from Zhang [2004], even if there it is not proposed explicitly as a online-to-batch conversion.

The non-uniform averaging of Example 3.3 is from Zhang [2004], even if there it is not proposed explicitly as an online-to-batch conversion. Instead, the non-uniform averaging of Example 4.12 is from Lacoste-Julien et al. [2012], but again there is not proposed as an online-to-batch conversion. The basic idea of solving the problem of SVM with OSD and online-to-batch conversion of Example 4.12 was the Pegasos algorithm [Shalev-Shwartz et al., 2007], for many years the most used optimizer for SVMs.

A more recent method to do online-to-batch conversion has been introduced in Cutkosky [2019a]. The new method allows to prove the convergence of the last iterate rather than the one of the weighted average, with a small change in any online learning algorithm.

Theorem 3.13 is from Cesa-Bianchi et al. [2004]. Theorem 3.14 is nothing else than the Agnostic PAC learning guarantee for ERM for hypothesis classes with finite cardinality. Cesa-Bianchi et al. [2004] gives also an alternative procedure to select a single hypothesis among the T generated during the online procedure that does not require splitting the data in training and validation. However, the obtained guarantee matches the one we have proved.

3.5 Exercises

Problem 3.1. *Implement the algorithm in Example 3.2 in any language you like: implementing an algorithm is the perfect way to see if you understood all the details of the algorithm.*

Chapter 4

Beyond \sqrt{T} Regret

4.1 Strong Convexity and Online Subgradient Descent

Let's now go back to online convex optimization theory. The example in the first chapter showed us that it is possible to get logarithmic regret in time. However, we saw that we get only \sqrt{T} -regret with Online Subgradient Descent (OSD) on the same game. What is the reason? It turns out that the losses in the first game, $\ell_t(x) = (x - y_t)^2$ on $[0, 1]$, are not just Lipschitz. They also possess some *curvature* that can be exploited to achieve a better regret. In a moment we will see that the only change we will need to OSD is a different learning rate, dictated as usual by the regret analysis.

The key concept we will need is the one of *strong convexity*.

4.1.1 Convex Analysis Bits: Strong Convexity

Here, we introduce a stronger concept of convexity, that allows to build better lower bound to a function. Instead of the linear lower bound achievable through the use of subgradients, we will make use of *quadratic* lower bound.

Definition 4.1 (Strongly Convex Function). *Let $\mu \geq 0$. A proper function $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ is μ -strongly convex over a convex set $V \subseteq \text{int dom } f$ w.r.t. $\|\cdot\|$ if*

$$\forall \mathbf{x}, \mathbf{y} \in V, \mathbf{g} \in \partial f(\mathbf{x}), \quad f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2.$$

Remark 4.2. *You might find another definition of strong convexity, that resembles the one of convexity. However, it is easy to show that these two definitions are equivalent.*

Note that the any convex function is 0-strongly convex, by the definition of subgradient. Also, any μ -strongly convex function is also λ -strongly convex for any $0 \leq \lambda \leq \mu$.

In words, the definition above tells us that a strongly convex function can be lower bounded by a quadratic, where the linear term is the usual one constructed through the subgradient, and the quadratic term depends on the strong convexity. Hence, we have a tighter lower bound to the function w.r.t. simply using convexity. This is what we would expect using a Taylor expansion on a twice-differentiable convex function and lower bounding the smallest eigenvalue of the Hessian. Indeed, we have the following Theorem.

Theorem 4.3 ([Shalev-Shwartz, 2007, Lemma 14]). *Let $V \subseteq \mathbb{R}^d$ convex and $f : V \rightarrow \mathbb{R}$ twice differentiable. Then, a sufficient condition for μ -strong convexity in V w.r.t. $\|\cdot\|$ is that for all \mathbf{x}, \mathbf{y} we have $\langle \nabla^2 f(\mathbf{x}) \mathbf{y}, \mathbf{y} \rangle \geq \mu \|\mathbf{y}\|^2$, where $\nabla^2 f(\mathbf{x})$ is the Hessian matrix of f at \mathbf{x} .*

However, here there is the important difference that we do not assume the function to be twice differentiable. Indeed, we do not even need plain differentiability. Hence, the use of the subgradient implies that this lower bound does not have to be uniquely determined, as in the next Example.

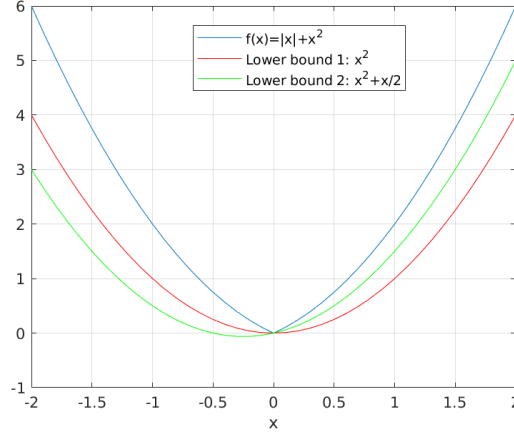


Figure 4.1: Possible lower bounds to the strongly convex non-differentiable function $f(x) = |x| + x^2$.

Example 4.4. Consider the strongly convex function $f(x) = |x| + x^2$. In Figure 4.1, we show two possible quadratic lower bounds to the function in $x = 0$.

We also have the following easy but useful property on the sum of strong convex functions.

Theorem 4.5. Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ be μ_1 -strongly convex and $g : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ a μ_2 -strongly convex function in a non-empty convex set $V \subseteq \text{int dom } f \cap \text{int dom } g$ w.r.t. $\|\cdot\|_2$. Then, $f + g$ is $\mu_1 + \mu_2$ -strongly convex in V w.r.t. $\|\cdot\|_2$.

Proof. Note that the assumption on V give us that the subdifferential set of the sum is equal to the sum of the subdifferential sets. Hence, the proof is immediate from the definition of strong convexity. \square

Example 4.6. Let $f(x) = \frac{1}{2}\|x\|_2^2$. Using Theorem 4.3, we have that f is 1-strongly convex w.r.t. $\|\cdot\|_2$ in \mathbb{R}^d .

4.1.2 Online Subgradient Descent for Strongly Convex Losses

Theorem 4.7. Let V a non-empty closed convex set in \mathbb{R}^d . Assume that the functions $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ are μ_t -strongly convex w.r.t $\|\cdot\|_2$ over $V \subseteq \bigcap_{i=1}^T \text{int dom } \ell_i$, where $\mu_t > 0$. Use OSD in Algorithm 2.2 with stepsizes equal to $\eta_t = \frac{1}{\sum_{i=1}^t \mu_i}$. Then, for any $u \in V$, we have the following regret guarantee

$$\sum_{t=1}^T \ell_t(x_t) - \sum_{t=1}^T \ell_t(u) \leq \frac{1}{2} \sum_{t=1}^T \eta_t \|g_t\|_2^2.$$

Proof. From the assumption of μ_t -strong convexity of the functions ℓ_t , we have that

$$\ell_t(x_t) - \ell_t(u) \leq \langle g_t, x_t - u \rangle - \frac{\mu_t}{2} \|x_t - u\|_2^2.$$

From the fact that $\eta_t = \frac{1}{\sum_{i=1}^t \mu_i}$, we have

$$\begin{aligned} \frac{1}{2\eta_1} - \frac{\mu_1}{2} &= 0, \\ \frac{1}{2\eta_t} - \frac{\mu_t}{2} &= \frac{1}{2\eta_{t-1}}, t = 2, \dots, T. \end{aligned}$$

Hence, use Lemma 2.23 and sum from $t = 1, \dots, T$, to obtain

$$\begin{aligned} \sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) &\leq \sum_{t=1}^T \left(\frac{1}{2\eta_t} \|\mathbf{x}_t - \mathbf{u}\|_2^2 - \frac{1}{2\eta_t} \|\mathbf{x}_{t+1} - \mathbf{u}\|_2^2 - \frac{\mu_t}{2} \|\mathbf{x}_t - \mathbf{u}\|^2 + \frac{\eta_t}{2} \|\mathbf{g}_t\|_2^2 \right) \\ &= -\frac{1}{2\eta_1} \|\mathbf{x}_2 - \mathbf{u}\|_2^2 + \sum_{t=2}^T \left(\frac{1}{2\eta_{t-1}} \|\mathbf{x}_t - \mathbf{u}\|_2^2 - \frac{1}{2\eta_t} \|\mathbf{x}_{t+1} - \mathbf{u}\|_2^2 \right) + \sum_{t=1}^T \frac{\eta_t}{2} \|\mathbf{g}_t\|_2^2. \end{aligned}$$

Observing that the first sum on the left hand side is a telescopic sum, we have the stated bound. \square

Remark 4.8. Notice that the theorem requires a bounded domain, otherwise the loss functions will not be Lipschitz given that they are also strongly convex.

Corollary 4.9. Under the assumptions of Theorem 4.7, if in addition we have $\mu_t = \mu > 0$ and ℓ_t is L -Lipschitz w.r.t. $\|\cdot\|_2$, for $t = 1, \dots, T$, then we have

$$\sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \frac{L^2}{2\mu} (1 + \ln T).$$

Remark 4.10. Corollary 4.9 does not imply that for any finite T the regret will be smaller than using learning rates $\propto \frac{1}{L\sqrt{t}}$. Instead, asymptotically the regret in Corollary 4.9 is always better than to one of OSD with Lipschitz losses.

Example 4.11. Consider once again the example in the first class: $\ell_t(x) = (x - y_t)^2$. Note that the loss functions are 2-strongly convex w.r.t. $|\cdot|$. Hence, setting $\eta_t = \frac{1}{2t}$ and $\ell'_t(x) = 2(x - y_t)$ gives a regret of $\ln(T) + 1$.

Let's now use again the online-to-batch conversion on strongly convex stochastic problems.

Example 4.12. As done before, we can use the online-to-batch conversion to use Corollary 4.9 to obtain stochastic subgradient descent algorithms for strongly convex stochastic functions. For example, consider the classic Support Vector Machine objective

$$\min_{\mathbf{x}} F(\mathbf{x}) := \frac{\lambda}{2} \|\mathbf{x}\|_2^2 + \frac{1}{N} \sum_{i=1}^N \max(1 - y_i \langle \mathbf{z}_i, \mathbf{x} \rangle, 0),$$

or any other regularized formulation like regularized logistic regression:

$$\min_{\mathbf{x}} F(\mathbf{x}) := \frac{\lambda}{2} \|\mathbf{x}\|_2^2 + \frac{1}{N} \sum_{i=1}^N \ln(1 + \exp(-y_i \langle \mathbf{z}_i, \mathbf{x} \rangle)),$$

where $\mathbf{z}_i \in \mathbb{R}^d$, $\|\mathbf{z}_i\|_2 \leq R$, and $y_i \in \{-1, 1\}$. First, notice that the minimizer of both expressions as to be in the L_2 ball of radius proportional to $\sqrt{\frac{1}{\lambda}}$ (proof left as exercise). Hence, we can set V equal to this set. Then, setting $\ell_t(\mathbf{x}) = \frac{\lambda}{2} \|\mathbf{x}\|_2^2 + \max(1 - y_i \langle \mathbf{z}_i, \mathbf{x} \rangle, 0)$ or $\ell_t(\mathbf{x}) = \frac{\lambda}{2} \|\mathbf{x}\|_2^2 + \ln(1 + \exp(-y_i \langle \mathbf{z}_i, \mathbf{x} \rangle))$ results in λ -strongly convex loss functions. Using Corollary 4.9 and Theorem 3.1 gives immediately

$$\mathbb{E} \left[F \left(\frac{1}{T} \sum_{t=1}^T \mathbf{x}_t \right) \right] - \min_{\mathbf{x}} F(\mathbf{x}) \leq O \left(\frac{\ln T}{\lambda T} \right).$$

However, we can do better! Indeed, $\ell_t(\mathbf{x}) = \frac{\lambda t}{2} \|\mathbf{x}\|_2^2 + t \max(1 - y_i \langle \mathbf{z}_i, \mathbf{x} \rangle, 0)$ or $\ell_t(\mathbf{x}) = \frac{\lambda t}{2} \|\mathbf{x}\|_2^2 + t \ln(1 + \exp(-y_i \langle \mathbf{z}_i, \mathbf{x} \rangle))$ results in λt -strongly convex loss functions. Using Corollary 4.9, we have that $\eta_t = \frac{2}{\lambda t(t+1)}$ and Theorem 3.1 gives immediately

$$\mathbb{E} \left[F \left(\frac{2}{T(T+1)} \sum_{t=1}^T t \mathbf{x}_t \right) \right] - \min_{\mathbf{x}} F(\mathbf{x}) \leq O \left(\frac{1}{\lambda T} \right),$$

that is asymptotically better because it does not have the logarithmic term.

4.2 Adaptive Algorithms: L^* bounds and AdaGrad

In this section, we will explore a bit more under which conditions we can get better regret upper bounds than $O(DL\sqrt{T})$. Also, we will obtain this improved guarantees in an *automatic* way. That is, the algorithm will be *adaptive* to characteristics of the sequence of loss functions, without having to rely on information about the future.

4.2.1 Adaptive Learning Rates for Online Subgradient Descent

Consider the minimization of the linear regret

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle .$$

Using Online Subgradient Descent (OSD), we said that the regret for bounded domains can be upper bounded by

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq \frac{D^2}{2\eta_T} + \frac{1}{2} \sum_{t=1}^T \eta_t \|\mathbf{g}_t\|_2^2 .$$

With a fixed learning rate, the learning rate that minimizes this upper bound on the regret is

$$\eta_T^* = \frac{D}{\sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_2^2}} .$$

Unfortunately, as we said, this learning rate cannot be used because it assumes the knowledge of the future rounds. However, we might be lucky and we might try to just approximate it in each round using the knowledge up to time t . That is, we might try to use

$$\eta_t = \frac{D}{\sqrt{\sum_{i=1}^t \|\mathbf{g}_i\|_2^2}} . \quad (4.1)$$

Observe that $\eta_T = \eta_T^*$, so the first term of the regret would be exactly what we need! For the other term, the optimal learning rate would give us

$$\frac{1}{2} \sum_{t=1}^T \eta_t^* \|\mathbf{g}_t\|_2^2 = \frac{1}{2} D \sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_2^2} .$$

Now let's see what we obtain with our approximation.

$$\frac{1}{2} \sum_{t=1}^T \eta_t \|\mathbf{g}_t\|_2^2 = \frac{1}{2} D \sum_{t=1}^T \frac{\|\mathbf{g}_t\|_2^2}{\sqrt{\sum_{i=1}^t \|\mathbf{g}_i\|_2^2}} .$$

We need a way to upper bound that sum. The way to treat these sums, as we did in other cases, is to try to approximate them with integrals. So, we can use the following very handy Lemma that generalizes a lot of similar specific ones.

Lemma 4.13. *Let $a_0 \geq 0$ and $f : [0, +\infty) \rightarrow [0, +\infty)$ a nonincreasing function. Then*

$$\sum_{t=1}^T a_t f\left(a_0 + \sum_{i=1}^t a_i\right) \leq \int_{a_0}^{\sum_{i=0}^T a_i} f(x) dx .$$

Proof. Denote by $s_t = \sum_{i=0}^t a_i$.

$$a_t f\left(a_0 + \sum_{i=1}^t a_i\right) = a_t f(s_t) = \int_{s_{t-1}}^{s_t} f(s_t) dx \leq \int_{s_{t-1}}^{s_t} f(x) dx .$$

Summing over $t = 1, \dots, T$, we have the stated bound. \square

Using this Lemma, we have that

$$\frac{1}{2}D \sum_{t=1}^T \frac{\|g_t\|_2^2}{\sqrt{\sum_{i=1}^t \|g_i\|_2^2}} \leq D \sqrt{\sum_{t=1}^T \|g_t\|_2^2}.$$

Surprisingly, this term is only a factor of 2 worse than what we would have got from the optimal choice of η_T^* . However, this learning rate can be computed without knowledge of the future and it can actually be used! Overall, with this choice we get

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \frac{3}{2}D \sqrt{\sum_{t=1}^T \|g_t\|_2^2}. \quad (4.2)$$

Note that it is possible improve the constant in front of the bound to $\sqrt{2}$ by multiplying the learning rates by $\frac{\sqrt{2}}{2}$. So, putting all together we have the following theorem.

Theorem 4.14. *Let $V \subseteq \mathbb{R}^d$ a closed non-empty convex set with diameter D , i.e. $\max_{\mathbf{x}, \mathbf{y} \in V} \|\mathbf{x} - \mathbf{y}\|_2 \leq D$. Let ℓ_1, \dots, ℓ_T an arbitrary sequence of convex functions $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ subdifferentiable in open sets containing V for $t = 1, \dots, T$. Pick any $\mathbf{x}_1 \in V$ and $\eta_t = \frac{\sqrt{2}D}{2\sqrt{\sum_{i=1}^t \|g_i\|_2^2}}$, $t = 1, \dots, T$. Then, $\forall \mathbf{u} \in V$, the following regret bound holds*

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \sqrt{2}D \sqrt{\sum_{t=1}^T \|g_t\|_2^2} = \sqrt{2} \min_{\eta > 0} \frac{D^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^T \|g_t\|_2^2.$$

The second equality in the theorem clearly show the advantage of this learning rates: We obtain (almost) the same guarantee we would have got knowing the future gradients!

This is an interesting result on its own: it gives a principled way to set the learning rates with an almost optimal guarantee. However, there are also other consequences of this simple regret. First, we will specialize this result to the case that the losses are *smooth*.

4.2.2 Convex Analysis Bits: Smooth functions

We now consider a family of loss functions that have the characteristic of being lower bounded by the squared norm of the subgradient. We will also introduce the concept of *dual norms*. While dual norms are not strictly needed for this topic, they give more generality and at the same time they allows me to slowly introduce some of the concepts that will be needed for the chapter on Online Mirror Descent.

Definition 4.15 (Dual Norm). *The **dual norm** $\|\cdot\|_*$ of a norm $\|\cdot\|$ is defined as $\|\theta\|_* = \max_{\mathbf{x}: \|\mathbf{x}\| \leq 1} \langle \theta, \mathbf{x} \rangle$.*

Example 4.16. *The dual norm of the L_2 norm is the L_2 norm. Indeed, $\|\theta\|_* = \max_{\mathbf{x}: \|\mathbf{x}\|_2 \leq 1} \langle \theta, \mathbf{x} \rangle \leq \|\theta\|_2$ by Cauchy-Schwarz inequality. Also, set $\mathbf{v} = \frac{\theta}{\|\theta\|_2}$, so $\max_{\mathbf{x}: \|\mathbf{x}\|_2 \leq 1} \langle \theta, \mathbf{x} \rangle \geq \langle \theta, \mathbf{v} \rangle = \|\theta\|_2$.*

If you do not know the concept of *operator norms*, the concept of dual norm can be a bit weird at first. One way to understand it is that it is a way to measure how “big” are linear functionals. For example, consider the linear function $f(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle$, we want to try to understand how big it is. So, we can measure $\max_{\mathbf{x} \neq 0} \frac{\langle \mathbf{z}, \mathbf{x} \rangle}{\|\mathbf{x}\|}$ that is we measure how big is the output of the linear functional compared to its input \mathbf{x} , where \mathbf{x} is measured with some norm. Now, you can show that the above is equivalent to the dual norm of \mathbf{z} .

Remark 4.17. *The definition of dual norm immediately implies $\langle \theta, \mathbf{x} \rangle \leq \|\theta\|_* \|\mathbf{x}\|$.*

Now we can introduce smooth functions, using the dual norms defined above.

Definition 4.18 (Smooth Function). *Let $f : V \rightarrow \mathbb{R}$ differentiable. We say that f is **M -smooth** w.r.t. $\|\cdot\|$ if $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_* \leq M\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in V$.*

Keeping in mind the intuition above on dual norms, taking the dual norm of a gradient makes sense if you associate each gradient with the linear functional $\langle \nabla f(\mathbf{y}), \mathbf{x} \rangle$, that is the one needed to create a linear approximation of f .

Smooth functions have many properties, for example a smooth function can be upper bounded by a quadratic. However, in the following we will need the following property.

Theorem 4.19 ([e.g., Lemma 4.1 Srebro et al., 2010]). *Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be M -smooth and bounded from below, then for all $\mathbf{x} \in \mathbb{R}^d$*

$$\|\nabla f(\mathbf{x})\|_*^2 \leq 2M(f(\mathbf{x}) - \inf_{\mathbf{y} \in \mathbb{R}^d} f(\mathbf{y})) .$$

4.2.3 L^* bounds

Assume now that the loss functions ℓ_1, \dots, ℓ_T are bounded from below and M smooth. Without loss of generality, we can assume that each of them is bounded from below by 0. Under these assumptions, from the regret in (4.2) and Theorem 4.19 we immediately obtain

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq 2D \sqrt{M \sum_{t=1}^T \ell_t(\mathbf{x}_t)} .$$

This is an implicit bound, in the sense that $\sum_{t=1}^T \ell_t(\mathbf{x}_t)$ appears on both sides of the inequality. To make it explicit, we will use the following simple Lemma (proof left as an exercise).

Lemma 4.20. *Let $a, c > 0$, $b \geq 0$, and $x \geq 0$ such that $x - \sqrt{ax + b} \leq c$. Then $x \leq a + c + 2\sqrt{b + ac}$.*

So, we have the following theorem.

Theorem 4.21. *Let $V \subseteq \mathbb{R}^d$ a closed non-empty convex set with diameter D , i.e. $\max_{\mathbf{x}, \mathbf{y} \in V} \|\mathbf{x} - \mathbf{y}\|_2 \leq D$. Let ℓ_1, \dots, ℓ_T an arbitrary sequence of non-negative convex functions $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ M -smooth in open sets containing V for $t = 1, \dots, T$. Pick any $\mathbf{x}_1 \in V$ and $\eta_t = \frac{\sqrt{2D}}{2\sqrt{\sum_{i=1}^t \|\mathbf{g}_i\|_2^2}}$, $t = 1, \dots, T$. Then, $\forall \mathbf{u} \in V$, the following regret bound holds*

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq 4MD^2 + 4D \sqrt{M \sum_{t=1}^T \ell_t(\mathbf{u})} .$$

This regret guarantee is very interesting because in the worst case it is still of the order $O(\sqrt{T})$, but in the best case scenario it becomes a constant! In fact, if there exists a $\mathbf{u} \in V$ such that $\sum_{t=1}^T \ell_t(\mathbf{u}) = 0$ we get a constant regret. Basically, if the losses are “easy”, the algorithm *adapts* to this situation and gives us a better regret.

These kind of guarantees are called L^* bounds because they depend on the cumulative competitor loss that can be denoted by L^* .

4.2.4 AdaGrad

We now present another application of the regret bound in (4.2). **AdaGrad**, that stands for Adaptive Gradient, is an Online Convex Optimization algorithm proposed independently by McMahan and Streeter [2010] and Duchi et al. [2010]. It aims at being adaptive to the sequence of gradients. It is usually known as a stochastic optimization algorithm, but in reality it was proposed for Online Convex Optimization (OCO). To use it as a stochastic algorithm, you should use an online-to-batch conversion, otherwise you do not have any guarantee of convergence.

We will present a proof that only allows hyperrectangles as feasible sets V , on the other hand the restriction makes the proof almost trivial. Let's see how it works.

AdaGrad has key ingredients:

- A coordinate-wise learning process;

- The adaptive learning rates in (4.1).

For the first ingredient, as we said, the regret of any OCO problem can be upper bounded by the regret of the Online Linear Optimization (OLO) problem. That is,

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

Now, the essential observation is to explicitly write the inner product as a sum of product over the single coordinates:

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle = \sum_{t=1}^T \sum_{i=1}^d g_{t,i} x_{t,i} - \sum_{t=1}^T \sum_{i=1}^d g_{t,i} u_i = \sum_{i=1}^d \left(\sum_{t=1}^T g_{t,i} x_{t,i} - \sum_{t=1}^T g_{t,i} u_i \right) = \sum_{i=1}^d \text{Regret}_{T,i}(u_i),$$

where we denoted by $\text{Regret}_{T,i}(u_i)$ the regret of the 1-dimensional OLO problem over coordinate i , that is $\sum_{t=1}^T g_{t,i} x_{t,i} - \sum_{t=1}^T g_{t,i} u_i$. In words, we can decompose the original regret as the sum of d OLO regret minimization problems and we can try to focus on each one of them separately.

A good candidate for the 1-dimensional problems is OSD with the learning rates in (4.1). We can specialize the regret in (4.2) to the 1-dimensional case for linear losses, so we get for each coordinate i

$$\sum_{t=1}^T g_{t,i} x_{t,i} - \sum_{t=1}^T g_{t,i} u_i \leq \sqrt{2} D_i \sqrt{\sum_{t=1}^T g_{t,i}^2}.$$

This choice gives us the AdaGrad algorithm in Algorithm 4.1.

Algorithm 4.1 AdaGrad for Hyperrectangles

Require: $V = \{\mathbf{x} : a_i \leq x_i \leq b_i\}, \mathbf{x}_1 \in V$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Output \mathbf{x}_t
 - 3: Receive $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ and pay $\ell_t(\mathbf{x}_t)$
 - 4: Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$
 - 5: Set $\eta_{t,i} = \frac{D_i}{\sqrt{\sum_{j=1}^t g_{j,i}^2}}$
 - 6: $\mathbf{x}_{t+1,i} = \max(\min(x_{t,i} - \eta_{t,i} g_{t,i}, b_i), a_i), i = 1, \dots, d$
 - 7: **end for**
-

Putting all together, we have immediately the following regret guarantee.

Theorem 4.22. Let $V = \{\mathbf{x} : a_i \leq x_i \leq b_i\}$ with diameters along each coordinate equal to $D_i = b_i - a_i$. Let ℓ_1, \dots, ℓ_T an arbitrary sequence of convex functions $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ subdifferentiable in open sets containing V for $t = 1, \dots, T$. Pick any $\mathbf{x}_1 \in V$ and $\eta_{t,i} = \frac{\sqrt{2} D_i}{2\sqrt{\sum_{j=1}^t g_{j,i}^2}}, t = 1, \dots, T$. Then, $\forall \mathbf{u} \in V$, the following regret bound holds

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \sqrt{2} \sum_{i=1}^d D_i \sqrt{\sum_{t=1}^T g_{t,i}^2}.$$

Is this a better regret bound compared to the one in Theorem 4.14? It depends! To compare the two we have to consider that V is a hyperrectangle because the analysis of AdaGrad above only works for hyperrectangle. Then, we have to compare

$$D \sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_2^2} \quad \text{versus} \quad \sum_{i=1}^d D_i \sqrt{\sum_{t=1}^T g_{t,i}^2}.$$

From Cauchy-Schwartz, we have that $\sum_{i=1}^d D_i \sqrt{\sum_{t=1}^T g_{t,i}^2} \leq D \sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_2^2}$. So, assuming the same sequence of subgradients, AdaGrad has always a better regret on hyperrectangles. Also, note that

$$\sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_2^2} \leq \sum_{i=1}^d \sqrt{\sum_{t=1}^T g_{t,i}^2} \leq \sqrt{d} \sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_2^2}$$

So, in the case that V is a hypercube we have $D_i = D_\infty = \max_{\mathbf{x}, \mathbf{y}} \|\mathbf{x} - \mathbf{y}\|_\infty$ and $D = \sqrt{d} D_\infty$, the bound of AdaGrad is between $1/\sqrt{d}$ and 1 times the bound of Theorem 4.14. In other words, if we are lucky with the subgradients, the particular **shape of the domain** might save us a factor of \sqrt{d} in the guarantee.

Note that the more general analysis of AdaGrad allows to consider arbitrary domains, but it does not change the general message that the best domains for AdaGrad are hyperrectangles. We will explore this issue of choosing the online algorithm based on the shape of the feasible set V when we will introduce Online Mirror Descent.

Hidden in the guarantee is that the biggest advantage of AdaGrad is the property of being coordinate-wise *scale-free*. That is, if each coordinate of the gradients are multiplied by different constants, the learning rate will automatically scale them back. This is hidden by the fact that the optimal solution \mathbf{u} would also scale accordingly, but the fixed diameters of the feasible set hide it. This might be useful in the case the ranges of coordinates of the gradients are vastly different one from the other. Indeed, this does happen in the stochastic optimization of deep neural networks, where the first layers have smaller magnitude of the gradients compared to the last layers.

4.3 History Bits

The logarithmic regret in Corollary 4.9 was shown for the first time in the seminal paper Hazan et al. [2006]. The general statement in Theorem 4.7 was proven by Hazan et al. [2008].

The adaptive learning rate in (4.1) first appeared in Streeter and McMahan [2010]. However, similar methods were used long time before. Indeed, the key observation to approximate oracle quantities with estimates up to time t was first proposed in the self-confident algorithms [Auer et al., 2002c], where the learning rate is inversely proportional to the square root of the cumulative loss of the algorithm, and for smooth losses it implies the L^* bounds similar to the one in Theorem 4.21.

AdaGrad was proposed in basically identically form independently by two groups at the same conference: McMahan and Streeter [2010] and Duchi et al. [2010]. The analysis presented here is the one in Streeter and McMahan [2010] that does not handle generic feasible sets and does not support “full-matrices”, i.e. full-matrix learning rates instead of diagonal ones. However, in machine learning applications AdaGrad is rarely used with a projection step (even if doing so provably destroys the worst-case performance [Orabona and Pál, 2018]). Also, in the adversarial setting full-matrices do not seem to offer advantages in terms of regret compared to diagonal ones.

Note that the AdaGrad learning rate is usually written as

$$\eta_{t,i} = \frac{D_i}{\epsilon + \sqrt{\sum_{j=1}^t g_{j,i}^2}},$$

where $\epsilon > 0$ is a small constant used to prevent division by zero. In reality, ϵ is not necessary and removing it makes the updates scale-free, as stressed in Orabona and Pál [2018].

AdaGrad inspired an incredible number of clones, most of them with similar, worse, or no regret guarantees. The keyword “adaptive” itself has shifted its meaning over time. It used to denote the ability of the algorithm to obtain the same guarantee as it knew in advance a particular property of the data (i.e. adaptive to the gradients/noise/scale = (almost) same performance as it knew the gradients/noise/scale in advance). Indeed, in Statistics this keyword is used with the same meaning. However, “adaptive” has changed its meaning over time. Nowadays, it seems to denote any kind of coordinate-wise learning rates that does not guarantee anything in particular.

4.4 Exercises

Problem 4.1. Prove that OSD in Example 4.11 with $x_1 = 0$ is exactly the Follow-the-Leader strategy for that particular problem.

Problem 4.2. Prove that $\ell_t(\mathbf{x}) = \|\mathbf{x} - \mathbf{z}_t\|_2^2$ is 2-strongly convex w.r.t. $\|\cdot\|_2$ and derive the OSD update for it and its regret guarantee.

Problem 4.3 (Difficult). Prove that $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_p^2$ is $p - 1$ -strongly convex w.r.t. $\|\cdot\|_p$ where $1 \leq p \leq 2$.

Problem 4.4. Prove that the dual norm of $\|\cdot\|_p$ is $\|\cdot\|_q$, where $\frac{1}{p} + \frac{1}{q} = 1$ and $p, q \geq 1$.

Problem 4.5. Show that using online subgradient descent with the learning rates in (4.1) with Lipschitz, smooth, and strongly convex functions you can get $O(\ln(1 + L^*))$ bounds.

Problem 4.6. Prove that the logistic loss $\ell(\mathbf{x}) = \ln(1 + \exp(-y\langle \mathbf{z}, \mathbf{x} \rangle))$, where $\|\mathbf{z}\|_2 \leq 1$ and $y \in \{-1, 1\}$ is $\frac{1}{4}$ -smooth w.r.t. $\|\cdot\|_2$.

Chapter 5

Lower bounds for Online Linear Optimization

In this chapter we will present some lower bounds for online linear optimization (OLO). Remembering that linear losses are convex, this immediately gives us lower bounds for online convex optimization (OCO). We will consider both the constrained and the unconstrained case. The lower bounds are important because they inform us on what are the optimal algorithms and where are the gaps in our knowledge.

5.1 Lower bounds for Bounded OLO

We will first consider the bounded constrained case. Finding a lower bound accounts to find a strategy for the adversary that forces a certain regret onto the algorithm, *no matter what the algorithm does*. We will use the probabilistic method to construct our lower bound.

The basic method relies on the fact that for any random variable z with domain V , and any function f

$$\sup_{x \in V} f(x) \geq \mathbb{E}[f(z)] .$$

For us, it means that we can lower bound the effect of the worst-case sequence of functions with an expectation over any distribution over the functions. If the distribution is chosen wisely, we can expect that gap in the inequality to be small. Why do you rely on expectations rather than actually constructing an adversarial sequence? Because the use of a stochastic loss functions makes very easy to deal with arbitrary algorithms. In particular, we will choose stochastic loss functions that makes the expected loss of the algorithm 0, independently from the strategy of the algorithm.

Theorem 5.1. *Let $V \subset \mathbb{R}^d$ be any non-empty bounded closed convex subset. Let $D = \sup_{v, w \in V} \|v - w\|_2$ be the diameter of V . Let \mathcal{A} be any (possibly randomized) algorithm for OLO on V . Let T be any non-negative integer. There exists a sequence of vectors g_1, \dots, g_T with $\|g_t\|_2 \leq L$ and $u \in V$ such that the regret of algorithm \mathcal{A} satisfies*

$$\text{Regret}_T(u) = \sum_{t=1}^T \langle g_t, x_t \rangle - \sum_{t=1}^T \langle g_t, u \rangle \geq \frac{\sqrt{2}LD\sqrt{T}}{4} .$$

Proof. Let's denote by $\text{Regret}_T = \max_{u \in V} \text{Regret}_T(u)$. Let $v, w \in V$ such that $\|v - w\|_2 = D$. Let $z = \frac{v-w}{\|v-w\|_2}$, so that $\langle z, v - w \rangle = D$. Let $\epsilon_1, \dots, \epsilon_T$ be i.i.d. Rademacher random variables, that is $\mathbb{P}[\epsilon_t = 1] = \mathbb{P}[\epsilon_t = -1] = 1/2$

and set the losses $g_t = L\epsilon_t \mathbf{z}$.

$$\begin{aligned}
\sup_{g_1, \dots, g_T} \text{Regret}_T &\geq \mathbb{E} \left[\sum_{t=1}^T L\epsilon_t \langle \mathbf{z}, \mathbf{x}_t \rangle - \min_{\mathbf{u} \in V} \sum_{t=1}^T L\epsilon_t \langle \mathbf{z}, \mathbf{u} \rangle \right] = \mathbb{E} \left[- \min_{\mathbf{u} \in V} \sum_{t=1}^T L\epsilon_t \langle \mathbf{z}, \mathbf{u} \rangle \right] \\
&= \mathbb{E} \left[\max_{\mathbf{u} \in V} \sum_{t=1}^T -L\epsilon_t \langle \mathbf{z}, \mathbf{u} \rangle \right] = \mathbb{E} \left[\max_{\mathbf{u} \in V} \sum_{t=1}^T L\epsilon_t \langle \mathbf{z}, \mathbf{u} \rangle \right] \\
&= \mathbb{E} \left[\max_{\mathbf{u} \in \{\mathbf{v}, \mathbf{w}\}} \sum_{t=1}^T L\epsilon_t \langle \mathbf{z}, \mathbf{u} \rangle \right] = \mathbb{E} \left[\frac{1}{2} \sum_{t=1}^T L\epsilon_t \langle \mathbf{z}, \mathbf{v} + \mathbf{w} \rangle + \frac{1}{2} \left| \sum_{t=1}^T L\epsilon_t \langle \mathbf{z}, \mathbf{v} - \mathbf{w} \rangle \right| \right] \\
&= \frac{L}{2} \mathbb{E} \left[\left| \sum_{t=1}^T \epsilon_t \langle \mathbf{z}, \mathbf{v} - \mathbf{w} \rangle \right| \right] = \frac{LD}{2} \mathbb{E} \left[\left| \sum_{t=1}^T \epsilon_t \right| \right] \geq \frac{\sqrt{2}LD\sqrt{T}}{4}.
\end{aligned}$$

where we used $\mathbb{E}[\epsilon_t] = 0$ and the ϵ_t are independent in the first equality, $\max(a, b) = \frac{a+b}{2} + \frac{|a-b|}{2}$ in the fourth equality, and Khintchine inequality in the last inequality. \square

We see that the lower bound is a constant multiplicative factor from the upper bound we proved for Online Subgradient Descent (OSD) with learning rates $\eta_t = \frac{D}{L\sqrt{t}}$ or $\eta = \frac{D}{L\sqrt{T}}$. This means that OSD is asymptotically optimal with both settings of the learning rate.

At this point there is an important consideration to do: How can this be the optimal regret when we managed to prove better regret, for example with adaptive learning rates? The subtlety is that, constraining the adversary to play L -Lipschitz losses, the adversary could always force on the algorithm at least the regret in Theorem 5.1. However, we can design algorithms that take advantage of *suboptimal plays of the adversary*. Indeed, for example, if the adversary plays in a way that all the subgradients have the same norm equal to L , there is nothing to adapt to!

5.2 Unconstrained Online Linear Optimization

We now move to the unconstrained case, however first we have to enrich our math toolbox with another essential tool, *Fenchel conjugates*.

5.2.1 Convex Analysis Bits: Fenchel Conjugate

Definition 5.2 (Closed Function). A function $f : V \subseteq \mathbb{R}^d \rightarrow [-\infty, +\infty]$ is **closed** iff $\{x : f(x) \leq \alpha\}$ is closed for every $\alpha \in \mathbb{R}$.

Note that in a Hausdorff space a function is closed iff it is lower semicontinuous [Bauschke and Combettes, 2011, Lemma 1.24].

Example 5.3. The indicator function of a set $V \subset \mathbb{R}^d$, is closed iff V is closed.

Definition 5.4 (Fenchel Conjugate). For a function $f : \mathbb{R}^d \rightarrow [-\infty, \infty]$, we define the **Fenchel conjugate** $f^* : \mathbb{R}^d \rightarrow [-\infty, \infty]$ as

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

From the definition we immediately obtain the Fenchel-Young's inequality

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

We have the following useful property for the Fenchel conjugate

Theorem 5.5 ([Rockafellar, 1970, Corollary 23.5.1 and Theorem 23.5]). Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ be convex, proper, and closed. Then

1. $\mathbf{x} \in \partial f^*(\boldsymbol{\theta})$ iff $\boldsymbol{\theta} \in \partial f(\mathbf{x})$.
2. $\langle \boldsymbol{\theta}, \mathbf{x} \rangle - f(\mathbf{x})$ achieves its supremum in \mathbf{x} at $\mathbf{x} = \mathbf{x}^*$ iff $\mathbf{x}^* \in \partial f^*(\boldsymbol{\theta})$.

Example 5.6. Let $f(x) = \exp(x)$, hence we have $f^*(\theta) = \max_x x\theta - \exp(x)$. Solving the optimization, we have that $x^* = \ln(\theta)$ if $\theta > 0$ and $f^*(\theta) = \theta \ln \theta - \theta$, $f^*(0) = 0$, and $f^*(\theta) = +\infty$ for $\theta < 0$.

Example 5.7. Consider the function $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|^2$, where $\|\cdot\|$ is a norm in \mathbb{R}^d , with dual norm $\|\cdot\|_*$. We can show that its conjugate is $f^*(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_*^2$. From

$$\langle \boldsymbol{\theta}, \mathbf{x} \rangle - \frac{1}{2}\|\mathbf{x}\|^2 \leq \|\boldsymbol{\theta}\|_* \|\mathbf{x}\| - \frac{1}{2}\|\mathbf{x}\|^2$$

for all \mathbf{x} . The right hand side is a quadratic function of $\|\mathbf{x}\|$, which has maximum value $\frac{1}{2}\|\boldsymbol{\theta}\|_*^2$. Therefore for all \mathbf{x} , we have

$$\langle \boldsymbol{\theta}, \mathbf{x} \rangle - \frac{1}{2}\|\mathbf{x}\|^2 \leq \frac{1}{2}\|\boldsymbol{\theta}\|_*^2,$$

which shows that $f^*(\boldsymbol{\theta}) \leq \frac{1}{2}\|\boldsymbol{\theta}\|_*^2$. To show the other inequality, let \mathbf{x} be any vector with $\langle \boldsymbol{\theta}, \mathbf{x} \rangle = \|\boldsymbol{\theta}\|_* \|\mathbf{x}\|$, scaled so that $\|\mathbf{x}\| = \|\boldsymbol{\theta}\|_*$. Then we have, for this \mathbf{x} ,

$$\langle \boldsymbol{\theta}, \mathbf{x} \rangle - \frac{1}{2}\|\mathbf{x}\|^2 = \frac{1}{2}\|\boldsymbol{\theta}\|_*^2,$$

which shows that $f^*(\boldsymbol{\theta}) \geq \frac{1}{2}\|\boldsymbol{\theta}\|_*^2$.

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

Proof. From the definition of conjugate function, we have

$$g^*(\boldsymbol{\theta}) = \sup_{\mathbf{x} \in \mathbb{R}^d} \langle \boldsymbol{\theta}, \mathbf{x} \rangle - af(\mathbf{x}) - b = -b + a \sup_{\mathbf{x} \in \mathbb{R}^d} \left\langle \frac{\boldsymbol{\theta}}{a}, \mathbf{x} \right\rangle - f(\mathbf{x}) = -b + af^*\left(\frac{\boldsymbol{\theta}}{a}\right). \quad \square$$

Lemma 5.9 ([Bauschke and Combettes, 2011, Example 13.7]). Let $f : \mathbb{R} \rightarrow (-\infty, +\infty]$ even. Then $(f \circ \|\cdot\|_2)^* = f^* \circ \|\cdot\|_2$.

5.2.2 Lower Bound for the Unconstrained Case

The above lower bound applies only to the constrained setting. In the unconstrained setting, we proved that OSD with $\mathbf{x}_1 = \mathbf{0}$ and constant learning rate of $\eta = \frac{1}{L\sqrt{T}}$ gives a regret of $\frac{1}{2}L(\|\mathbf{u}\|_2^2 + 1)\sqrt{T}$ for any $\mathbf{u} \in \mathbb{R}^d$. Is this regret optimal? It is clear that the regret must be at least linear in $\|\mathbf{u}\|_2$, but is linear enough?

The approach I will follow is to *reduce the OLO game to the online game of betting on a coin*, where the lower bounds are known. So, let's introduce the coin-betting online game:

- Start with an initial amount of money $\epsilon > 0$.
- In each round, the algorithm bets a fraction of its current wealth on the outcome of a coin.
- The outcome of the coin is revealed and the algorithm wins or lose its bet, 1 to 1.

The aim of this online game is to win as much money as possible. Also, as in all the online games we consider, we do not assume anything on how the outcomes of the coin are decided. Note that this game can also be written as OCO using the log loss.

We will denote by $c_t \in \{-1, 1\}$, $t = 1, \dots, T$ the outcomes of the coin. We will use the absolute value of $\beta_t \in [-1, 1]$ to denote the fraction of money to bet and its sign to denote on which side we are betting. The money

the algorithm has won from the beginning of the game till the end of round t will be denoted by r_t and given that the money are won or lost 1 to 1, we have

$$\underbrace{\text{Money at the end of round } t}_{r_t + \epsilon} = \underbrace{\text{Money at the beginning of round } t}_{r_{t-1} + \epsilon} + \underbrace{\text{Money won or lost}}_{c_t \beta_t (r_{t-1} + \epsilon)} = \epsilon \prod_{i=1}^t (1 + \beta_i c_i),$$

where we used the fact that $r_0 = 0$. We will also denote by $x_t = \beta_t(\epsilon + r_{t-1})$ the bet of the algorithm on round t .

If we got all the outcomes of the coin correct, we would double our money in each round, so that $\epsilon + r_T = \epsilon 2^T$. However, given the adversarial nature of the game, we can actually prove a stronger lower bound to the maximum wealth we can gain.

Theorem 5.10 ([Cesa-Bianchi and Lugosi, 2006, a simplified statement of Theorem 9.2]). *Let $T \geq 21$. Then, for any betting strategy with initial money $\epsilon > 0$ that bets fractions of its current wealth, there exists a sequence of coin outcomes c_t , such that*

$$\epsilon + \sum_{t=1}^T c_t x_t \leq \frac{1}{\sqrt{T}} \max_{-1 \leq \beta \leq 1} \epsilon \prod_{t=1}^T (1 + \beta c_t) \leq \frac{\epsilon}{\sqrt{T}} \exp \left(\frac{\ln 2 (\sum_{t=1}^T c_t)^2}{T} \right).$$

Now, let's connect the coin-betting game with OLO. Remember that proving a regret guarantee in OLO consists in showing

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq \psi_T(\mathbf{u}), \quad \forall \mathbf{u} \in \mathbb{R}^d,$$

for some function ψ_T , where we want the dependency on T to be sublinear. Using our new learned concept of Fenchel conjugate, this is equivalent to prove that

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle \leq \inf_{\mathbf{u}} \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle + \psi_T(\mathbf{u}) = - \sup_{\mathbf{u}} - \left\langle \sum_{t=1}^T \mathbf{g}_t, \mathbf{u} \right\rangle - \psi_T(\mathbf{u}) = -\psi_T^* \left(- \sum_{t=1}^T \mathbf{g}_t \right).$$

Hence, for a given online algorithm we can prove regret bounds proving that there exists a function ψ_T or equivalently finding its conjugate ψ_T^* . Similarly, proving a lower bound in unconstrained OLO means finding a sequence of \mathbf{g}_t and a function ψ_T or a function ψ_T^* that lower bound the regret or the cumulative losses of the algorithm respectively.

Without any other information, it can be challenging to guess what is the slowest increasing function ψ_T . So, we restrict our attention to online algorithms that guarantee a constant regret against the zero vector. This immediately imply the following important consequence.

Theorem 5.11. *Let $\epsilon(t)$ a non-decreasing function of the index of the rounds and \mathcal{A} an OLO algorithm that guarantees $\text{Regret}_t(\mathbf{0}) \leq \epsilon(t)$ for any sequence of $\mathbf{g}_1, \dots, \mathbf{g}_t \in \mathbb{R}^d$ with $\|\mathbf{g}_i\|_2 \leq 1$. Then, there exists β_t such that $\mathbf{x}_t = \beta_t(\epsilon(T) - \sum_{i=1}^{t-1} \langle \mathbf{g}_i, \mathbf{x}_i \rangle)$ and $\|\beta_t\|_2 \leq 1$ for $t = 1, \dots, T$.*

Proof. Define $r_t = - \sum_{i=1}^t \langle \mathbf{g}_i, \mathbf{x}_i \rangle$ the “reward” of the algorithm. So, we have

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle = -r_T + \left\langle \sum_{t=1}^T \mathbf{g}_t, \mathbf{u} \right\rangle.$$

Since, we assumed that $\text{Regret}_t(\mathbf{0}) \leq \epsilon(t)$, we always have $r_t \geq -\epsilon(t)$. Using this, we claim that $\|\mathbf{x}_t\|_2 \leq r_{t-1} + \epsilon(t)$ for all $t = 1, \dots, T$. To see this, assume that there is a sequence $\mathbf{g}_1, \dots, \mathbf{g}_{t-1}$ that gives $\|\mathbf{x}_t\|_2 > r_{t-1} + \epsilon(t)$. We then set $\mathbf{g}_t = \frac{\mathbf{x}_t}{\|\mathbf{x}_t\|_2}$. For this sequence, we would have $r_t = r_{t-1} - \|\mathbf{x}_t\|_2 < -\epsilon(t)$, that contradicts the observation that $r_t \geq -\epsilon(t)$.

So, from the fact that $\|\mathbf{x}_t\|_2 \leq r_{t-1} + \epsilon(t) \leq r_{t-1} + \epsilon(T)$ we have that there exists β_t such that $\mathbf{x}_t = \beta_t(\epsilon(T) + r_{t-1})$ for a β_t and $\|\beta_t\|_2 \leq 1$. \square

This theorem informs us of something important: *any OLO algorithm that suffer a non-decreasing regret against the null competitor must predict in the form of a “vectorial” coin-betting algorithm.* This immediately implies the following.

Theorem 5.12. *Let $T \geq 21$. For any OLO algorithm, under the assumptions of Theorem 5.11, there exist a sequence of $\mathbf{g}_1, \dots, \mathbf{g}_T \in \mathbb{R}^d$ with $\|\mathbf{g}_t\|_2 \leq 1$ and $\mathbf{u}^* \in \mathbb{R}^d$, such that*

$$\sum_{i=1}^t \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u}^* \rangle \geq 0.8 \|\mathbf{u}^*\|_2 \sqrt{T} \left(\sqrt{0.3 \ln \frac{0.8 \|\mathbf{u}^*\|_2 T}{\epsilon(T)}} - 1 \right) + \epsilon(T) \left(1 - \frac{1}{\sqrt{T}} \right).$$

Proof. The proof works by reducing the OLO game to a coin-betting game and then using the upper bound to the reward for coin-betting games.

First, set $\mathbf{g}_t = [-c_t, 0, \dots, 0]$, where $c_t \in \{-1, 1\}$ will be defined in the following, so that $\langle \mathbf{g}_t, \mathbf{x} \rangle = -c_t x_1$ for any $\mathbf{x} \in \mathbb{R}^d$. Given Theorem 5.11, we have that the first coordinate of \mathbf{x}_t has to satisfy

$$x_{t,1} = \beta_{t,1} \left(\epsilon(T) - \sum_{i=1}^{t-1} \langle \mathbf{g}_i, \mathbf{x}_i \rangle \right) = \beta_{t,1} \left(\epsilon(T) + \sum_{i=1}^{t-1} c_i x_{i,1} \right),$$

for some $\beta_{t,1}$ such that $|\beta_{t,1}| \leq 1$. Hence, the above is nothing else than a coin-betting algorithm that bets $x_{t,1}$ money on the outcome of a coin c_t , with initial money $\epsilon(T)$. This means that the upper bound to its reward in Theorem 5.10 applies: there exists a sequence of c_t such that

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \epsilon(T) = - \sum_{t=1}^T c_t x_{t,1} - \epsilon(T) \geq - \frac{\epsilon(T)}{\sqrt{T}} \exp \left(\frac{\ln 2 (\sum_{t=1}^T c_t)^2}{T} \right) = - \sum_{t=1}^T c_t u_1^* + f^*(|u_1^*|) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u}^* \rangle + f^*(\|\mathbf{u}^*\|_2),$$

where g^* is the Fenchel conjugate of the function $f(x) = \frac{\epsilon(T)}{\sqrt{T}} \exp \left(\frac{x^2 \ln 2}{T} \right)$ and $\mathbf{u}^* = f'(\sum_{t=1}^T c_t) \in R$ by Theorem 5.5 part 2. Using Theorem A.3 and reordering the terms, we get the stated bound. \square

From the above theorem we have that OSD with learning rate $\eta = \frac{\alpha}{L\sqrt{T}}$ does not have the optimal dependency on $\|\mathbf{u}\|_2$ for any $\alpha > 0$.

In the future classes, we will see that the connection between coin-betting and OLO can also be used to design OLO algorithm. This will give us *optimal unconstrained OLO algorithms with the surprising property of not requiring a learning rate at all.*

5.3 History Bits

The lower bound for OCO is quite standard, the proof presented is a simplified version of the one in Orabona and Pál [2018].

On the other hand, both the online learning literature and the optimization one almost ignored the issue of lower bounds for the unconstrained case. The connection between coin betting and OLO was first unveiled in Orabona and Pal [2016]. Theorem 5.11 is an unpublished result by Ashok Cutkosky, that proved similar and more general results in his PhD thesis [Cutkosky, 2018]. Theorem 5.12 is new, by me. McMahan and Abernethy [2013] implicitly also proposed using the conjugate function for lower bound in unconstrained OLO.

There is a caveat in the unconstrained lower bound: A stronger statement would be to choose the norm of \mathbf{u}^* beforehand. To do this, we would have to explicitly construct the sequence of the c_t . One way to do is to use Rademacher coins and then leverage again the hypothesis on the regret against the null competitor. This route was used in Streeter and McMahan [2012], but the proof relied on assuming the value of the global optimum of a non-convex function with an infinite number of local minima. The correct proof avoiding that step was then given in Orabona [2013]. Yet, the proof presented here, that converts reward upper bounds in regret lower bound, is simpler in spirit and (I hope!) more understandable. Given that, as far as I know, this is the first time that unconstrained OLO lower bounds are taught in a class, I valued simplicity over generality.

5.4 Exercises

Problem 5.1. Fix $U > 0$. Mimicking the proof of Theorem 5.1, prove that for any OCO algorithm there exists a \mathbf{u}^* and a sequence of loss functions such that $\text{Regret}_T(\mathbf{u}^*) \geq \frac{\sqrt{2}}{2} \|\mathbf{u}^*\|_2 L \sqrt{T}$ where $\|\mathbf{u}^*\|_2 = U$ and the loss functions are L -Lipschitz w.r.t. $\|\cdot\|_2$.

Problem 5.2. Extend the proof of Theorem 5.1 to an arbitrary norm $\|\cdot\|$ to measure the diameter of V and with $\|g_t\|_* \leq L$.

Problem 5.3. Assume $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ and $g : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ proper, where $f(\mathbf{x}) \leq g(\mathbf{x})$, $\forall \mathbf{x}$. Prove that $f^*(\boldsymbol{\theta}) \geq g^*(\boldsymbol{\theta})$, $\forall \boldsymbol{\theta}$.

Problem 5.4. Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ be even. Prove that f^* is even

Problem 5.5. Find the exact expression of the conjugate function of $f(x) = \alpha \exp(\beta x^2)$, for $\alpha, \beta > 0$. Hint: Wolfram Alpha or any other kind of symbolic solver can be very useful for this type of problems.

Chapter 6

Online Mirror Descent

In this chapter, we will introduce the Online Mirror Descent (OMD) algorithm. To explain its genesis, I think it is essential to understand what subgradients do. In particular, the negative subgradients are not always pointing towards a direction that minimizes the function.

6.1 Subgradients are not Informative

You know that in online learning we receive a sequence of loss functions and we have to output a vector before observing the loss function on which we will be evaluated. However, we can gain a lot of intuition if we consider the easy case that the sequence of loss functions is always a fixed function, i.e., $\ell_t(\mathbf{x}) = \ell(\mathbf{x})$. If our hypothetical online algorithm does not work in this situation, for sure it won't work on the more general case.

That said, we proved that the convergence of Online Subgradient Descent (OSD) depends on the following key property of the subgradients:

$$\ell(\mathbf{x}_t) - \ell(\mathbf{u}) \leq \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle. \quad (6.1)$$

In words, to minimize the left hand side of this equation, it is enough to minimize the right hand side, that is nothing else than the instantaneous linear regret on the linear function $\langle \mathbf{g}_t, \cdot \rangle$. This is the only reason why OSD works! However, I am sure you heard a million of times the (wrong) intuition that gradient points towards the minimum, and you might be tempted to think that the same (even more wrong) intuition holds for subgradients. Indeed, I am sure that even if we proved the regret guarantee based on (6.1), in the back of your mind you keep thinking “yeah, sure, it works because the subgradient tells me where to go to minimize the function”. Typically this idea is so strong that I have to present explicit counterexamples to fully convince a person.

So, take a look at the following examples that illustrate the fact that a subgradient does not always point in a direction where the function decreases.

Example 6.1. Let $f(\mathbf{x}) = \max[-x_1, x_1 - x_2, x_1 + x_2]$, see Figure 6.1. The vector $\mathbf{g} = [1, 1]$ is a subgradient in $\mathbf{x} = (1, 0)$ of $f(\mathbf{x})$. No matter how we choose the stepsize, moving in the direction of the negative subgradient will not decrease the objective function. An even more extreme example is in Figure 6.2, with the function $f(\mathbf{x}) = \max[x_1^2 + (x_2 + 1)^2, x_1^2 + (x_2 - 1)^2]$. Here, in the point $(1, 0)$, any positive step in the direction of the negative subgradient will increase the objective function.

6.2 Reinterpreting the Online Subgradient Descent Algorithm

How Online Subgradient Descent works? It works exactly as I told you before: thanks to (6.1). But, what does that inequality really mean?

A way to understand how OSD algorithms works is to think that it minimizes a local approximation of the original objective function. This is not unusual for optimization algorithms, for example the Netwon's algorithm constructs an

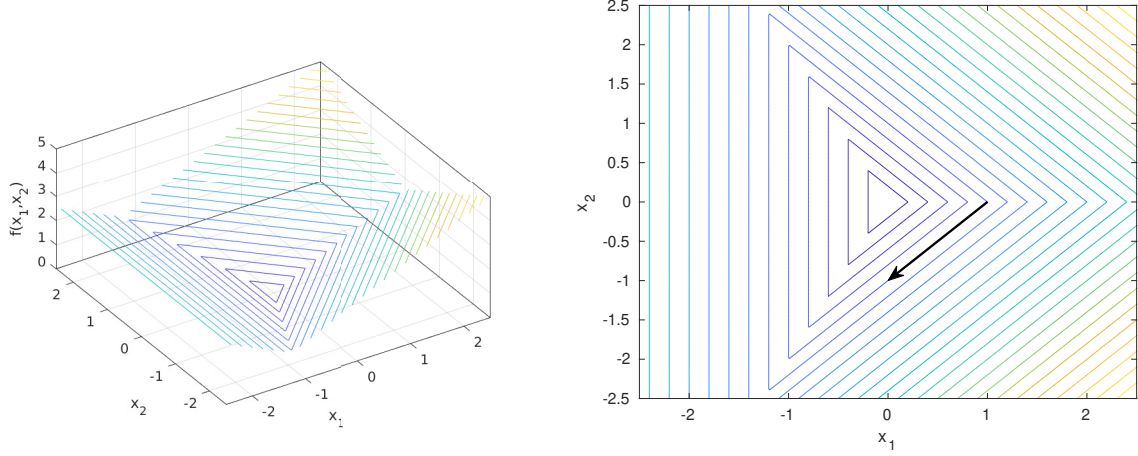


Figure 6.1: 3D plot (left) and level sets (right) of $f(\mathbf{x}) = \max[-x_1, x_1 - x_2, x_1 + x_2]$. A negative subgradient is indicated by the black arrow.

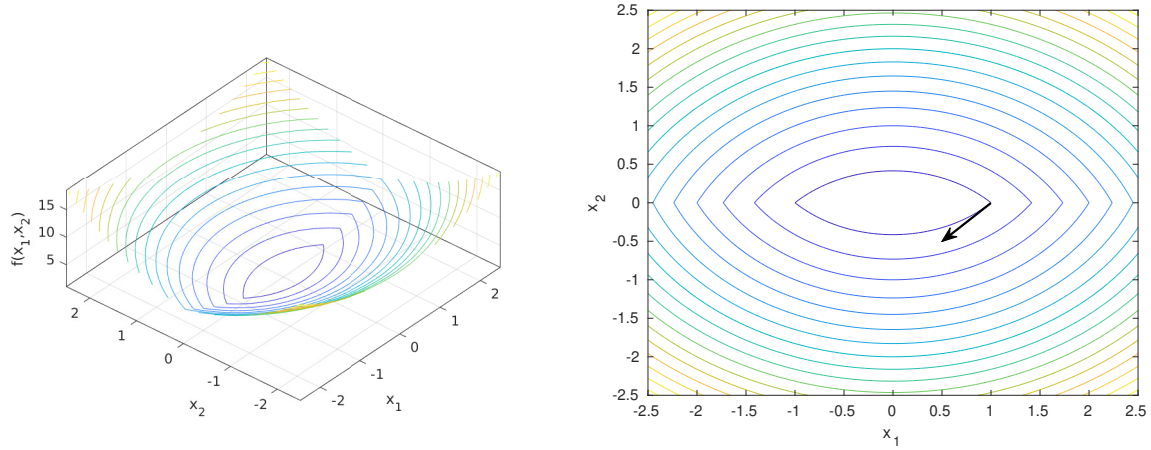


Figure 6.2: 3D plot (left) and level sets (right) of $f(\mathbf{x}) = \max[x_1^2 + (x_2 + 1)^2, x_1^2 + (x_2 - 1)^2]$. A negative subgradient is indicated by the black arrow.

approximation with a Taylor expansion truncated to the second term. Thanks to the definition of subgradients, we can immediately build a linear lower bound to a function f around \mathbf{x}_0 :

$$f(\mathbf{x}) \geq \tilde{f}(\mathbf{x}) := f(\mathbf{x}_0) + \langle \mathbf{g}, \mathbf{x} - \mathbf{x}_0 \rangle, \forall \mathbf{x} \in V.$$

So, in our setting, this would mean that we update the online algorithm with the minimizer of a linear approximation of the loss function you received. Unfortunately, minimizing a linear function is unlikely to give us a good online algorithm. Indeed, over unbounded domains the minimum of a linear function is $-\infty$.

So, let's introduce the other key concept: we constraint the minimization of this lower bound only in a neighborhood of \mathbf{x}_0 , where we have good reason to believe that the approximation is more precise. Coding the neighborhood constraint with a L_2 squared distance from \mathbf{x}_0 less than some positive number h , we might think to use the following update

$$\begin{aligned} \mathbf{x}_{t+1} = \operatorname{argmin}_{\mathbf{x} \in V} & f(\mathbf{x}_t) + \langle \mathbf{g}, \mathbf{x} - \mathbf{x}_t \rangle \\ \text{s.t. } & \|\mathbf{x}_t - \mathbf{x}\|^2 \leq h. \end{aligned}$$

Equivalently, for some $\eta > 0$, we can consider the unconstrained formulation

$$\operatorname{argmin}_{\mathbf{x} \in V} \hat{f}(\mathbf{x}) := f(\mathbf{x}_0) + \langle \mathbf{g}, \mathbf{x} - \mathbf{x}_0 \rangle + \frac{1}{2\eta} \|\mathbf{x}_0 - \mathbf{x}\|_2^2. \quad (6.2)$$

This is a well-defined update scheme, that hopefully moves \mathbf{x}_t closer to the optimum of f . See Figure 6.3 for a graphical representation in one-dimension.

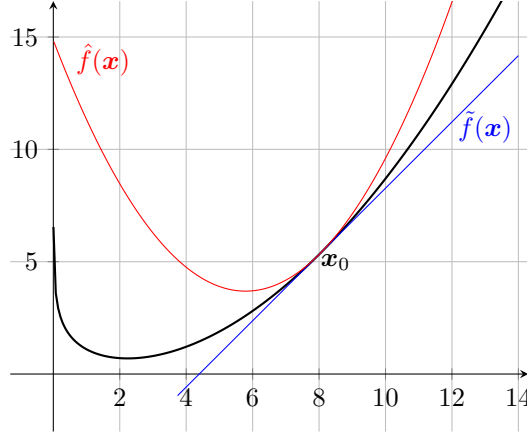


Figure 6.3: Approximations of $f(\mathbf{x})$.

And now the final element of our story: the argmin in (6.2) is exactly the update we used in OSD! Indeed, solving the argmin and completing the square, we get

$$\begin{aligned} \operatorname{argmin}_{\mathbf{x} \in V} \langle \mathbf{g}_t, \mathbf{x} \rangle + \frac{1}{2\eta_t} \|\mathbf{x}_t - \mathbf{x}\|_2^2 &= \operatorname{argmin}_{\mathbf{x} \in V} \|\eta_t \mathbf{g}_t\|^2 + 2\eta_t \langle \mathbf{g}_t, \mathbf{x} - \mathbf{x}_t \rangle + \|\mathbf{x}_t - \mathbf{x}\|_2^2 \\ &= \operatorname{argmin}_{\mathbf{x} \in V} \|\mathbf{x}_t - \eta_t \mathbf{g}_t - \mathbf{x}\|_2^2 \\ &= \Pi_V(\mathbf{x}_t - \eta_t \mathbf{g}_t), \end{aligned}$$

where Π_V is the Euclidean projection onto V , i.e. $\Pi_V(\mathbf{x}) = \operatorname{argmin}_{\mathbf{y} \in V} \|\mathbf{x} - \mathbf{y}\|_2$.

The new way to write the update of OSD in (6.2) will be the core ingredient for designing Online Mirror Descent. In fact, OMD is a strict generalization of that update when we use a different way to measure the locality of \mathbf{x} from \mathbf{x}_t . That is, we measured the distance between to the current point with the squared L_2 norm. What happens if we change the norm? Do we even have to use a norm?

To answer these questions we have to introduce another useful mathematical object: the *Bregman divergence*.

6.3 Convex Analysis Bits: Bregman Divergence

We first give a new definition, a slightly stronger notion of convexity.

Definition 6.2 (Strictly Convex Function). *Let $f : V \subseteq \mathbb{R}^d \rightarrow \mathbb{R}$ and V a convex set. f is **strictly convex** if*

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

From the definition, it is immediate to see that strong convexity w.r.t. any norm implies strict convexity. Note that for a differentiable function, strict convexity also implies that $f(\mathbf{y}) > f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle$ for $\mathbf{x} \neq \mathbf{y}$ [Bauschke and Combettes, 2011, Proposition 17.13].

We now define our new notion of “distance”.

Definition 6.3 (Bregman Divergence). *Let $\psi : X \rightarrow \mathbb{R}$ be strictly convex and continuously differentiable on $\text{int } X$. The **Bregman Divergence** w.r.t. ψ is $B_\psi : X \times \text{int } X \rightarrow \mathbb{R}$ defined as*

$$B_\psi(\mathbf{x}; \mathbf{y}) = \psi(\mathbf{x}) - \psi(\mathbf{y}) - \langle \nabla \psi(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle .$$

From the definition, we see that the Bregman divergence is always non-negative for $\mathbf{x}, \mathbf{y} \in \text{int } X$, from the convexity of ψ . However, something stronger holds. By the strict convexity of ψ , fixed a point $\mathbf{y} \in \text{int } X$ we have that $\psi(\mathbf{y}) \geq \psi(\mathbf{x}) + \langle \nabla \psi(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle$, $\forall \mathbf{y} \in X$, with equality only for $\mathbf{y} = \mathbf{x}$. Hence, the strict convexity allows us to use the Bregman divergence as a similarity measure between \mathbf{x} and \mathbf{y} . Moreover, this similarity measure *changes* with the reference point \mathbf{y} . This also implies that, as you can see from the definition, the Bregman divergence is not symmetric.

Let me give you some more intuition on the concept Bregman divergence. Consider the case that ψ is twice differentiable in a ball B around \mathbf{y} and $\mathbf{x} \in B$. So, by the Taylor's theorem, there exists $0 \leq \alpha \leq 1$ such that

$$B_\psi(\mathbf{x}; \mathbf{y}) = \psi(\mathbf{x}) - \psi(\mathbf{y}) - \nabla \psi(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) = \frac{1}{2}(\mathbf{x} - \mathbf{y})^\top \nabla^2 \psi(\mathbf{z})(\mathbf{x} - \mathbf{y}),$$

where $\mathbf{z} = \alpha \mathbf{x} + (1 - \alpha)\mathbf{y}$. Hence, we are using a *local norm* that depends on the Hessian of ψ . *Different areas of the space will have a different value of the Hessian, and so the Bregman will behave differently.*

We can also lower bound the Bregman divergence if the function ψ is strongly convex. In particular, if ψ is λ -strongly convex w.r.t. a norm $\|\cdot\|$ in $\text{int } X$, then we have

$$B_\psi(\mathbf{x}; \mathbf{y}) \geq \frac{\lambda}{2} \|\mathbf{x} - \mathbf{y}\|^2 . \quad (6.3)$$

Example 6.4. *If $\psi(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|_2^2$, then $B_\psi(\mathbf{x}; \mathbf{y}) = \frac{1}{2} \|\mathbf{x}\|_2^2 - \frac{1}{2} \|\mathbf{y}\|_2^2 - \langle \mathbf{y}, \mathbf{x} \rangle = \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|_2^2$.*

Example 6.5. *If $\psi(\mathbf{x}) = \sum_{i=1}^d x_i \ln x_i$ and $X = \{\mathbf{x} \in \mathbb{R}^d : x_i \geq 0, \|\mathbf{x}\|_1 = 1\}$, then $B_\psi(\mathbf{x}; \mathbf{y}) = \sum_{i=1}^d x_i \ln \frac{x_i}{y_i}$, that is the Kullback-Leibler divergence between the discrete distributions \mathbf{x} and \mathbf{y} .*

We also have the following lemma that links the Bregman divergences between 3 points.

Lemma 6.6 ([Chen and Teboulle, 1993]). *Let B_ψ the Bregman divergence w.r.t. $\psi : X \rightarrow \mathbb{R}$. Then, for any three points $\mathbf{x}, \mathbf{y} \in \text{int } X$ and $\mathbf{z} \in X$, the following identity holds*

$$B_\psi(\mathbf{z}; \mathbf{x}) + B_\psi(\mathbf{x}; \mathbf{y}) - B_\psi(\mathbf{z}; \mathbf{y}) = \langle \nabla \psi(\mathbf{y}) - \nabla \psi(\mathbf{x}), \mathbf{z} - \mathbf{x} \rangle .$$

6.4 Online Mirror Descent

Based on what we said before, we can start from the equivalent formulation of the OSD update

$$\mathbf{x}_{t+1} = \underset{\mathbf{x} \in V}{\operatorname{argmin}} \langle \mathbf{g}_t, \mathbf{x} \rangle + \frac{1}{2\eta_t} \|\mathbf{x}_t - \mathbf{x}\|_2^2,$$

and we can change the last term with another measure of distance. In particular, using the Bregman divergence w.r.t. a function ψ , we have

$$\mathbf{x}_{t+1} = \underset{\mathbf{x} \in V}{\operatorname{argmin}} \langle \mathbf{g}_t, \mathbf{x} \rangle + \frac{1}{\eta_t} B_\psi(\mathbf{x}; \mathbf{x}_t) .$$

These two updates are exactly the same when $\psi(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|_2^2$.

So we get the Online Mirror Descent algorithm in Algorithm 6.1.

However, without an additional assumption, this algorithm has a problem. Can you see it? The problem is that \mathbf{x}_{t+1} might be on the boundary of V and in the next step we would have to evaluate $B_\psi(\mathbf{x}; \mathbf{x}_{t+1})$ for a point on the boundary of V . Given that $V \subseteq X$, we might end up on the boundary of X where the Bregman is not defined!

Algorithm 6.1 Online Mirror Descent

Require: Non-empty closed convex $V \subseteq X \subseteq \mathbb{R}^d$, $\psi : X \rightarrow \mathbb{R}$ strictly convex and continuously differentiable on $\text{int } X$, $\mathbf{x}_1 \in V$ such that ψ is differentiable in $\mathbf{x}_1, \eta_1, \dots, \eta_T > 0$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Output \mathbf{x}_t
 - 3: Receive $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ and pay $\ell_t(\mathbf{x}_t)$
 - 4: Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$
 - 5: $\mathbf{x}_{t+1} = \operatorname{argmin}_{\mathbf{x} \in V} \langle \mathbf{g}_t, \mathbf{x} \rangle + \frac{1}{\eta_t} B_\psi(\mathbf{x}; \mathbf{x}_t)$
 - 6: **end for**
-

To fix this problem, we need either one of the following assumptions

$$\lim_{\mathbf{x} \rightarrow \partial X} \|\nabla \psi(\mathbf{x})\|_2 = +\infty, \quad (6.4)$$

$$V \subseteq \text{int } X. \quad (6.5)$$

If either of these conditions are true, the update is well-defined on all rounds (proof left as an exercise).

Now we have a well-defined algorithm, but does it guarantee a sublinear regret? We know that at least in one case it recovers the OSD algorithm, that does work. So, from an intuitive point of view, how well the algorithm work should depend on some characteristic on ψ . In particular, a key property will be the *strong convexity* of ψ .

To analyze OMD, we first prove a one step relationship, similar to the one we proved for Online Gradient Descent and OSD. Note how in this Lemma, we will use a lot of the concepts we introduced till now: strong convexity, dual norms, subgradients, Fenchel-Young inequality, etc. In a way, over the past sections I slowly prepared you to be able to prove this lemma.

Lemma 6.7. *Let B_ψ the Bregman divergence w.r.t. $\psi : X \rightarrow \mathbb{R}$ and assume ψ to be λ -strongly convex with respect to $\|\cdot\|$ in V . Let $V \subseteq X$ a non-empty closed convex set. Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$. Assume (6.4) or (6.5) hold. Then, $\forall \mathbf{u} \in V$ and with the notation in Algorithm 6.1, the following inequality holds*

$$\eta_t(\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \eta_t \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq B_\psi(\mathbf{u}; \mathbf{x}_t) - B_\psi(\mathbf{u}; \mathbf{x}_{t+1}) + \frac{\eta_t^2}{2\lambda} \|\mathbf{g}_t\|_*^2.$$

Proof. From the optimality condition for the update of OMD, we have

$$\langle \eta_t \mathbf{g}_t + \nabla \psi(\mathbf{x}_{t+1}) - \nabla \psi(\mathbf{x}_t), \mathbf{u} - \mathbf{x}_{t+1} \rangle \geq 0, \quad \forall \mathbf{u} \in V. \quad (6.6)$$

From the definition of subgradient, we have that

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

where in the second inequality we used (6.6), in the second equality we used Lemma 6.6, in the third inequality we used $\langle \mathbf{x}, \mathbf{y} \rangle \leq \frac{1}{2\lambda} \|\mathbf{x}\|^2 + \frac{\lambda}{2} \|\mathbf{y}\|_*^2, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, \lambda > 0$, and in the last inequality we used (6.3) because ψ is λ -strongly convex w.r.t. $\|\cdot\|$.

The lower bound with the function values is due, as usual, to the definition of subgradients. \square

Next time, we will see how to use this one step relationship to prove a regret bound, that will finally show us if and when this entire construction is a good idea. In fact, it is worth stressing that *the above motivation is not enough in any way to justify the existence of the OMD algorithm*. Also, next time we will explain why the algorithm is called Online “Mirror” Descent.

We can now prove a regret bound for OMD.

Theorem 6.8. Set $\mathbf{x}_1 \in V$ such that ψ is differentiable in \mathbf{x}_1 . Assume $\eta_{t+1} \leq \eta_t$, $t = 1, \dots, T$. Then, under the assumptions of Lemma 6.7 and $\forall \mathbf{u} \in V$, the following regret bounds hold

$$\sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \max_{1 \leq t \leq T} \frac{B_\psi(\mathbf{u}; \mathbf{x}_t)}{\eta_T} + \frac{1}{2\lambda} \sum_{t=1}^T \eta_t \|\mathbf{g}_t\|_*^2.$$

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

$$\sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \frac{B_\psi(\mathbf{u}; \mathbf{x}_1)}{\eta} + \frac{\eta}{2\lambda} \sum_{t=1}^T \|\mathbf{g}_t\|_*^2.$$

Proof. Fix $\mathbf{u} \in V$. As in the proof of OGD, dividing the inequality in Lemma 6.7 by η_t and summing from $t = 1, \dots, T$, we get

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

where we denoted by $D^2 = \max_{1 \leq t \leq T} B_\psi(\mathbf{u}; \mathbf{x}_t)$.

The second statement is left as an exercise. □

In words, OMD allows us to prove regret guarantees that depend on arbitrary couple of dual norms $\|\cdot\|$ and $\|\cdot\|_*$. In particular, the primal norm will be used to measure the feasible set V or the distance between the competitor and the initial point, and the dual norm will be used to measure the gradients. If you happen to know something about these quantities, we can choose the most appropriate couple of norm to guarantee a small regret. The only thing you need is a function ψ that is strongly convex with respect to the primal norm you have chosen ψ .

Overall, the regret bound is still of the order of \sqrt{T} for Lipschitz functions, that only difference is that now the Lipschitz constant is measured with a different norm. Also, everything we did for Online Subgradient Descent (OSD) can be trivially used here. So, for example, we can use stepsize of the form

$$\eta_t = \frac{D\sqrt{\lambda}}{\sqrt{\sum_{i=1}^t \|\mathbf{g}_i\|_*^2}}$$

to achieve a regret bound of $2 \frac{D}{\sqrt{\lambda}} \sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_*^2}$.

Next time, we will see practical examples of OMD that guarantee strictly better regret than OSD. As we did in the case of AdaGrad, the better guarantee will depend on the shape of the domain and the characteristics of the subgradients.

Instead, now we see the meaning of the “Mirror”.

6.4.1 The “Mirror” Interpretation

First, we need a couple of convex analysis results.

When we introduced the Fenchel conjugate, we said that $\mathbf{x} \in \partial f^*(\boldsymbol{\theta})$ iff $\boldsymbol{\theta} \in \partial f(\mathbf{x})$, that in words means that $(\partial f)^{-1} = \partial f^*$ in the sense of multivalued mappings. Now, we state a stronger result for the case that the function is strongly convex.

Theorem 6.9. *Let $\psi : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ be a proper, convex, and closed function, $\lambda > 0$ strongly convex w.r.t. $\|\cdot\|$. Then,*

1. ψ^* is finite everywhere and differentiable.
2. ψ^* is $\frac{1}{\lambda}$ -smooth w.r.t. $\|\cdot\|_*$.

We will also use the following optimality condition.

Theorem 6.10. *Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ proper. Then $\mathbf{x}^* \in \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$ iff $\mathbf{0} \in \partial f(\mathbf{x}^*)$.*

Proof. We have that

$$\mathbf{x}^* \in \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) \Leftrightarrow \forall \mathbf{y} \in \mathbb{R}^d, f(\mathbf{y}) \geq f(\mathbf{x}^*) = f(\mathbf{x}^*) + \langle \mathbf{0}, \mathbf{y} - \mathbf{x}^* \rangle \Leftrightarrow \mathbf{0} \in \partial f(\mathbf{x}^*). \quad \square$$

Hence, we can state the following theorem.

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

$$\operatorname{argmin}_{\mathbf{x} \in V} \langle \mathbf{g}_t, \mathbf{x} \rangle + \frac{1}{\eta_t} B_\psi(\mathbf{x}; \mathbf{x}_t) = \nabla \psi_V^*(\nabla \psi_V(\mathbf{x}_t) - \eta_t \mathbf{g}_t), \quad (6.7)$$

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

Proof. Consider the update rule in Algorithm 6.1 and let's see

$$\begin{aligned} \mathbf{x}_{t+1} &= \operatorname{argmin}_{\mathbf{x} \in V} \langle \mathbf{g}_t, \mathbf{x} \rangle + \frac{1}{\eta_t} B_\psi(\mathbf{x}; \mathbf{x}_t) \\ &= \operatorname{argmin}_{\mathbf{x} \in V} \eta_t \langle \mathbf{g}_t, \mathbf{x} \rangle + B_\psi(\mathbf{x}; \mathbf{x}_t) \\ &= \operatorname{argmin}_{\mathbf{x} \in V} \eta_t \langle \mathbf{g}_t, \mathbf{x} \rangle + \psi(\mathbf{x}) - \psi(\mathbf{x}_t) - \langle \nabla \psi(\mathbf{x}_t), \mathbf{x} - \mathbf{x}_t \rangle \\ &= \operatorname{argmin}_{\mathbf{x} \in V} \langle \eta_t \mathbf{g}_t - \nabla \psi(\mathbf{x}_t), \mathbf{x} \rangle + \psi(\mathbf{x}). \end{aligned}$$

Now, we want to use the first order optimality condition, so we have to use a little bit of subdifferential calculus. Given that $\operatorname{int} X \cap V \neq \{\}$, by the subdifferential calculus theorem we saw, we have $\partial(f + g) = \partial f + \partial g$. So, we have

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

that is

$$\nabla \psi(\mathbf{x}_t) - \eta_t \mathbf{g}_t \in (\nabla \psi + \partial i_V)(\mathbf{x}_{t+1}) = \partial \psi_V(\mathbf{x}_{t+1}).$$

or equivalently

$$\mathbf{x}_{t+1} \in \partial \psi_V^*(\nabla \psi(\mathbf{x}_t) - \eta_t \mathbf{g}_t).$$

Using that fact that $\psi_V := \psi + i_V$ is λ -strongly convex, we have that $\partial \psi_V^* = \{\nabla \psi_V^*\}$. Hence

$$\mathbf{x}_{t+1} = \nabla \psi_V^*(\nabla \psi(\mathbf{x}_t) - \eta_t \mathbf{g}_t).$$

Noting that $\psi_V \equiv \psi$ for vectors in V , we have the stated bound. \square

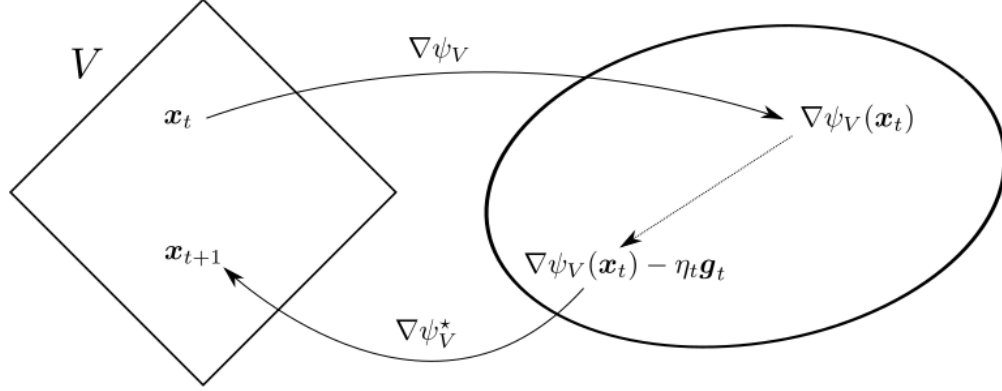


Figure 6.4: OMD update in terms of duality mappings.

Let's explain what this theorem says. We said that Online Mirror Descent extends the Online Subgradient Descent method to non-euclidean norms. Hence, the regret bound we proved contains dual norms, that measure the iterate and the gradients. We also said that it makes sense to use a dual norm to measure a gradient, because it is a natural way to measure how “big” is the linear functional $x \rightarrow \langle \nabla f(y), x \rangle$. In a more correct way, gradients actually live in the *dual space*, that is in a different space of the predictions x_t . Hence, we cannot sum iterates and gradients together, in the same way in which we cannot sum pear and apples together. So, why we were doing it in OSD? The reason is that in that case the dual space coincides with the primal space. But, it is a very particular case due to the fact that we used the L_2 norm. Instead, in the general case, iterates and gradients are in two different spaces.

So, in OMD we need a way to go from one space to the other. And this is exactly the role of $\nabla\psi$ and $\nabla\psi^*$, that are called *duality mappings*. We can now understand that the theorem tells us that OMD takes the primal vector x_t , transforms it into a dual vector through $\nabla\psi$, does a subgradient descent step in the dual space, and finally transforms the vector back to the primal space through $\nabla\psi^*$. This reasoning is summarized in Figure 6.4.

Example 6.12. Let $\psi : \mathbb{R}^d \rightarrow \mathbb{R}$ equal to $\psi(x) = \frac{1}{2}\|x\|_2^2 + i_V(x)$ where $V = \{x \in \mathbb{R}^d : \|x\|_2 \leq 1\}$. Then,

$$\psi^*(\theta) = \sup_{x \in V} \langle \theta, x \rangle - \frac{1}{2}\|x\|_2^2 = \sup_{-1 \leq \alpha \leq 1} \alpha \|\theta\|_2 - \frac{1}{2}\alpha^2.$$

Solving the constrained optimization problem, we have $\alpha^* = \min(1, \|\theta\|_2)$. Hence, we have

$$\psi^*(\theta) = \min \left(\frac{1}{2}\|\theta\|_2^2, \|\theta\|_2 - \frac{1}{2} \right),$$

that is finite everywhere and differentiable. So, we have $\nabla\psi(x) = x$ and

$$\nabla\psi^*(\theta) = \begin{cases} \theta, & \|\theta\|_2 \leq 1 \\ \frac{\theta}{\|\theta\|_2}, & \|\theta\|_2 > 1 \end{cases} = \Pi_V(\theta).$$

So, using (6.7), we obtain exactly the update of projected online subgradient descent.

6.4.2 Yet Another Way to Write the Online Mirror Descent Update

There exists yet another way to write the update of OMD. This third method uses the concept of *Bregman projections*. Extending the definition of Euclidean projections, we can define the projection with respect to a Bregman divergence. Let $\Pi_{V,\psi}$ be defined by

$$\Pi_{V,\psi}(x) = \operatorname{argmin}_{y \in V} B_\psi(y; x).$$

In the online learning literature, the OMD algorithm is typically presented with a two step update: first, solving the argmin over the entire space and then projecting back over X with respect to the Bregman divergence. In the following, we show that most of the time the two-step update is equivalent to the one-step update in (6.7).

First, we prove a general theorem that allows to break the constrained minimization of functions in the minimization over the entire space plus and Bregman projection step.

Theorem 6.13. *Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ proper, closed, strictly convex, and differentiable in $\text{int dom } f$. Also, let $V \subset \mathbb{R}^d$ a non-empty, closed convex set with $V \cap \text{dom } f \neq \{\}$ and assume that $\tilde{\mathbf{y}} = \arg\min_{\mathbf{z} \in \mathbb{R}^d} f(\mathbf{z})$ exists and $\tilde{\mathbf{y}} \in \text{int dom } f$. Denote by $\mathbf{y}' = \arg\min_{\mathbf{z} \in V} B_f(\mathbf{z}; \tilde{\mathbf{y}})$. Then the following hold:*

1. $\mathbf{y} = \arg\min_{\mathbf{z} \in V} f(\mathbf{z})$ exists and is unique.
2. $\mathbf{y} = \mathbf{y}'$.

Proof. For the first point, from [Bauschke and Combettes, 2011, Proposition 11.12] and the existence of $\tilde{\mathbf{y}}$, we have that f is coercive. So, from Bauschke and Combettes [2011, Proposition 11.14], the minimizer of f in V exists. Given that f is strictly convex, the minimizer must be unique too.

For the second point, from the definition of \mathbf{y} , we have $f(\mathbf{y}) \leq f(\mathbf{y}')$. On the other hand, from the first-order optimality condition, we have $\nabla f(\tilde{\mathbf{y}}) = \mathbf{0}$. So, we have

$$f(\mathbf{y}') - f(\tilde{\mathbf{y}}) = B_f(\mathbf{y}'; \tilde{\mathbf{y}}) \leq B_f(\mathbf{y}; \tilde{\mathbf{y}}) = f(\mathbf{y}) - f(\tilde{\mathbf{y}}),$$

that is $f(\mathbf{y}') \leq f(\mathbf{y})$. Given that f is strictly convex, $\mathbf{y}' = \mathbf{y}$. □

Now, note that, if $\tilde{\psi}(\mathbf{x}) = \psi(\mathbf{x}) + \langle \mathbf{g}, \mathbf{x} \rangle$, then

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

Also, defining $g(\mathbf{x}) = B_{\psi}(\mathbf{x}; \mathbf{y})$ is equal to $\psi(\mathbf{x}) + \langle \mathbf{z}, \mathbf{x} \rangle + K$ for some $K \in \mathbb{R}$ and $\mathbf{z} \in \mathbb{R}^d$. Hence, under the assumption of the above theorem, we have that $\mathbf{x}_{t+1} = \arg\min_{\mathbf{x} \in V} \langle \mathbf{g}_t, \mathbf{x} \rangle + \frac{1}{\eta_t} B_{\psi}(\mathbf{x}; \mathbf{x}_t)$ is equivalent to

$$\begin{aligned} \tilde{\mathbf{x}}_{t+1} &= \arg\min_{\mathbf{x} \in \mathbb{R}^d} \langle \eta_t \mathbf{g}_t, \mathbf{x} \rangle + B_{\psi}(\mathbf{x}; \mathbf{x}_t) \\ \mathbf{x}_{t+1} &= \arg\min_{\mathbf{x} \in V} B_{\psi}(\mathbf{x}; \tilde{\mathbf{x}}_{t+1}). \end{aligned}$$

The advantage of this update is that sometimes it gives two easier problems to solve rather than a single difficult one.

6.5 Example of OMD: Exponentiated Gradient

Let $X = \{\mathbf{x} \in \mathbb{R}^d : x_i \geq 0, \|\mathbf{x}\|_1 = 1\}$ and $\psi(\mathbf{x}) : X \rightarrow \mathbb{R}$ defined as the negative entropy $\psi(\mathbf{x}) = \sum_{i=1}^d x_i \ln x_i$, where we define $0 \ln(0) = 0$. Also, we set the feasibility set $V = X$. So, in words, we want to output discrete probability distributions over \mathbb{R}^d .

The Fenchel conjugate $\psi_V^*(\boldsymbol{\theta})$ is defined as

$$\psi_V^*(\boldsymbol{\theta}) = \sup_{\mathbf{x} \in V} \langle \boldsymbol{\theta}, \mathbf{x} \rangle - \psi(\mathbf{x}) = \sup_{\mathbf{x} \in V} \langle \boldsymbol{\theta}, \mathbf{x} \rangle - \sum_{i=1}^d x_i \ln x_i.$$

It is a constrained optimization problem, we can solve it using the KKT conditions. We first rewrite it as a minimization problem

$$\sup_{\mathbf{x} \in V} \langle \boldsymbol{\theta}, \mathbf{x} \rangle - \sum_{i=1}^d x_i \ln x_i = - \min_{\mathbf{x} \in V} \sum_{i=1}^d x_i \ln x_i - \langle \boldsymbol{\theta}, \mathbf{x} \rangle.$$

Algorithm 6.2 Exponentiated Gradient

Require: $\eta > 0$

- 1: Set $\mathbf{x}_1 = [1/d, \dots, 1/d]$
 - 2: **for** $t = 1$ **to** T **do**
 - 3: Output \mathbf{x}_t
 - 4: Receive ℓ_t and pay $\ell_t(\mathbf{x}_t)$
 - 5: Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$
 - 6: $x_{t+1,j} = \frac{x_{t,j} \exp(-\eta g_{t,j})}{\sum_{i=1}^d x_{t,i} \exp(-\eta g_{t,i})}$, $j = 1, \dots, d$
 - 7: **end for**
-

The KKT conditions are, for $i = 1, \dots, d$,

$$\begin{aligned} 1 + \ln x_i^* - \theta_i - \lambda_i^* + \nu^* &= 0 \\ \lambda_i^* &\geq 0 \\ \sum_{i=1}^d x_i^* &= 1 \\ \lambda_i^* x_i^* &= 0 \\ x_i^* &\geq 0. \end{aligned}$$

From the first one we get $x_i^* = \exp(\theta_i + \lambda_i^* - \nu^* - 1)$. Using the third one we have $\exp(\nu^* + 1) = \sum_{i=1}^d \exp(\theta_i + \lambda_i^*)$. Then, from the complementarity condition, fourth one, we get $\lambda_i^* = 0$. Putting all together, we have $x_i^* = \frac{\exp(\theta_i)}{\sum_{j=1}^d \exp(\theta_j)}$.

Denoting with $\alpha = \sum_{i=1}^d \exp(\theta_i)$, and substituting in the definition of the conjugate function we get

$$\begin{aligned} \psi_V^*(\boldsymbol{\theta}) &= \sum_{i=1}^d \left(\frac{1}{\alpha} \theta_i \exp(\theta_i) - \frac{1}{\alpha} \exp(\theta_i) (\theta_i - \ln(\alpha)) \right) \\ &= \ln(\alpha) \frac{1}{\alpha} \sum_{i=1}^d \exp(\theta_i) \\ &= \ln(\alpha) \\ &= \ln \left(\sum_{i=1}^d \exp(\theta_i) \right). \end{aligned}$$

We also have $(\nabla \psi_V^*)_j(\boldsymbol{\theta}) = \frac{\exp(\theta_j)}{\sum_{i=1}^d \exp(\theta_i)}$ and $(\nabla \psi_V)(\mathbf{x})_j = \ln(x_j) + 1$.

Putting all together, we have the online mirror descent update rule for entropic distance generating function.

$$x_{t+1,j} = \frac{\exp(\ln x_{t,j} + 1 - \eta_t g_{t,j})}{\sum_{i=1}^d \exp(\ln x_{t,i} + 1 - \eta_t g_{t,i})} = \frac{x_{t,j} \exp(-\eta_t g_{t,j})}{\sum_{i=1}^d x_{t,i} \exp(-\eta_t g_{t,i})}.$$

The algorithm is summarized in Algorithm 6.2. This algorithm is called *Exponentiated Gradient* (EG) because in the update rule we take the component-wise exponent of the gradient vector.

Let's take a look at the regret bound we get. First, as we said, observe that

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

that is the KL divergence between the 1-dimensional discrete distributions \mathbf{x} and \mathbf{y} . Now, the following Lemma tells us about the strong convexity of ψ .

Lemma 6.14 ([Shalev-Shwartz, 2007, Lemma 16]). $\psi(\mathbf{x}) = \sum_{i=1}^d x_i \ln x_i$ is 1-strongly convex with respect to the L_1 norm over the set $\{\mathbf{x} \in \mathbb{R}^d : x_i \geq 0, \|\mathbf{x}\|_1 = 1\}$.

Another thing to decide is the initial point \mathbf{x}_1 . We can set \mathbf{x}_1 to be the minimizer of ψ in V . In this way $B_\psi(\mathbf{u}; \mathbf{x}_1)$ simplifies to $\psi(\mathbf{u}) - \min_{\mathbf{x} \in V} \psi(\mathbf{x})$. Hence, we set $\mathbf{x}_1 = [1/d, \dots, 1/d] \in \mathbb{R}^d$. So, we have $B_\psi(\mathbf{u}; \mathbf{x}_1) = \sum_{i=1}^d u_i \ln u_i + \ln d \leq \ln d$.

Putting all together, we have

$$\sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \frac{\ln d}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2.$$

Assuming $\|\mathbf{g}_t\|_\infty \leq L_\infty$, we can set $\eta = \sqrt{\frac{2 \ln d}{L_\infty^2 T}}$, to obtain that a regret of $\frac{\sqrt{2}}{2} L_\infty \sqrt{T \ln d}$.

Remark 6.15. Note that the time-varying version of OMD with entropic distance generating function would give rise to a vacuous bound, can you see why? We will see how FTRL overcomes this issue using a time-varying regularizer rather than a time-varying learning rate.

How would Online Subgradient Descent (OSD) work on the same problem? First, it is important to realize that nothing prevents us to use OSD on this problem. We just have to implement the euclidean projection onto the simplex. The regret bound we would get from OSD is

$$\sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \frac{1}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_2^2,$$

where we set $\mathbf{x}_1 = \mathbf{0}$ and $\|\mathbf{u}\|_2 \leq 1$ for any $\mathbf{u} \in V$. Assuming $\|\mathbf{g}_t\|_\infty \leq L_\infty$, we have that in the worst case $\|\mathbf{g}_t\|_2^2 \leq d L_\infty^2$. Hence, we can set $\eta = \sqrt{\frac{1}{d L_\infty^2 T}}$, to obtain that a regret of $L_\infty \sqrt{dT}$. Hence, in a worst case sense, using an entropic distance generating function transforms a dependency on the dimension from \sqrt{d} to $\sqrt{\ln d}$ for Online Convex Optimization (OCO) over the probability simplex.

So, as we already saw analyzing AdaGrad, the shape of the domain is the important ingredient when we change from euclidean norms to other norms.

6.6 Example of OMD: p -norm Algorithms

Consider the distance generating function $\psi(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|_p^2$, for $1 < p \leq 2$ over $X = V = \mathbb{R}^d$. Let's remind the reader that the p -norm of a vector \mathbf{x} is defined as $(\sum_{i=1}^d |x_i|^p)^{\frac{1}{p}}$. We already proved that $\psi_V^*(\boldsymbol{\theta}) = \frac{1}{2} \|\boldsymbol{\theta}\|_q^2$, where $\frac{1}{p} + \frac{1}{q} = 1$, so that $q \geq 2$. Let's calculate the dual maps: $(\nabla \psi(\mathbf{x}))_j = \text{sign}(x_j) |x_j|^{p-1} \|\mathbf{x}\|_p^{2/p-1}$ and $(\nabla \psi_V^*(\boldsymbol{\theta}))_j = \text{sign}(\theta_j) |\theta_j|^{q-1} \|\boldsymbol{\theta}\|_q^{2/q-1}$. Hence, we can write the update rule as

$$\begin{aligned} \tilde{x}_{t+1,j} &= \text{sign}(x_{t,j}) |x_{t,j}|^{p-1} \|\mathbf{x}_t\|_p^{2/p-1} - \eta g_{t,j}, \quad j = 1, \dots, d, \\ x_{t+1,j} &= \text{sign}(\tilde{x}_{t+1,j}) |\tilde{x}_{t+1,j}|^{q-1} \|\tilde{\mathbf{x}}_{t+1}\|_q^{2/q-1}, \quad j = 1, \dots, d, \end{aligned}$$

where we broke the update in two steps to simplify the notation (and the implementation). Starting from $\mathbf{x}_1 = \mathbf{0}$, we have that

$$B_\psi(\mathbf{u}; \mathbf{x}_1) = \psi(\mathbf{u}) - \psi(\mathbf{x}_1) - \langle \nabla \psi(\mathbf{x}_1), \mathbf{u} - \mathbf{x}_1 \rangle = \psi(\mathbf{u}).$$

The last ingredient is the fact that $\psi(\mathbf{x})$ is $p-1$ strongly convex with respect to $\|\cdot\|_p$.

Lemma 6.16 ([Shalev-Shwartz, 2007, Lemma 17]). $\psi(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|_p^2$ is $(p-1)$ -strongly convex with respect to $\|\cdot\|_p$, for $1 \leq p \leq 2$.

Hence, the regret bound will be

$$\sum_{t=1}^T (\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{u})) \leq \frac{\|\mathbf{u}\|_p^2}{2\eta} + \frac{\eta}{2(p-1)} \sum_{t=1}^T \|\mathbf{g}_t\|_q^2.$$

Setting $p = 2$, we get the (unprojected) Online Subgradient Descent. However, we can set p to achieve a logarithmic dependency in the dimension d as in EG. Let's assume again that $\|\mathbf{g}_t\|_\infty \leq L_\infty$, so we have

$$\sum_{t=1}^T \|\mathbf{g}_t\|_q^2 \leq L_\infty^2 d^{2/q} T.$$

Also, note that $\|\mathbf{u}\|_p \leq \|\mathbf{u}\|_1$, so we have an upper bound to the regret of

$$\text{Regret}_T(\mathbf{u}) \leq \frac{\|\mathbf{u}\|_1^2}{2\eta} + \frac{L_\infty^2 d^{2/q} T \eta}{2(p-1)}, \forall \mathbf{u} \in \mathbb{R}^d.$$

Setting $\eta = \frac{\alpha\sqrt{p-1}}{L_\infty d^{1/q}\sqrt{T}}$, we get an upper bound to the regret of

$$\frac{1}{2} \left(\frac{\|\mathbf{u}\|_1^2}{\alpha} + \alpha \right) L_\infty \sqrt{T} \frac{d^{1/q}}{\sqrt{p-1}} = \frac{1}{2} \left(\frac{\|\mathbf{u}\|_1^2}{\alpha} + \alpha \right) L_\infty \sqrt{T} \sqrt{q-1} d^{1/q} \leq \frac{1}{2} \left(\frac{\|\mathbf{u}\|_1^2}{\alpha} + \alpha \right) L_\infty \sqrt{T} \sqrt{q} d^{1/q}.$$

Assuming $d \geq 3$, the choice of q that minimizes the last term is $q = 2 \ln d$ that makes the term $\sqrt{q} d^{1/q} = \sqrt{2e \ln d}$. Hence, we have regret bound of the order of $O(\sqrt{T \ln d})$.

So, the p -norm allows to interpolate from the behaviour of OSD to the one of EG. Note that here the set V is the entire space, however we could still set $V = \{\mathbf{x} \in \mathbb{R}^d : x_i \geq 0, \|\mathbf{x}\|_1 = 1\}$. While this would allow us to get the same asymptotic bound of EG, the update would not be in a closed form anymore.

6.7 An Application of Online Mirror Descent: Learning with Expert Advice

Let's introduce a particular Online Convex Optimization game called *Learning with Expert Advice*.

In this setting, we have d experts that gives us some advice on each round. In turn, in each round we have to decide which expert we want to follow. After we made our choice, the losses associated to each expert are revealed and we pay the loss associated to the expert we picked. The aim of the game is to minimize the losses we make compared to cumulative losses of the best expert. This is a general setting that allows to model many interesting cases. For example, we have a number of different online learning algorithms and we would like to close to the best among them.

Is this problem solvable? If we put ourselves in the adversarial setting, unfortunately it cannot be solved! Indeed, even with 2 experts, the adversary can force on us linear regret. Let's see how. In each round we have to pick expert 1 or expert 2. In each round, the adversary can decide that the expert we pick has loss 1 and the other one has loss 0. This means that the cumulative loss of the algorithm over T rounds is T . On the other hand, the best cumulative loss over expert 1 and 2 is less than $T/2$. This means that our regret, no matter what we do, can be as big as $T/2$.

The problem above is due to the fact that the adversary has too much power. One way to reduce its power is using *randomization*. We can allow the algorithm to be randomized *and* force the adversary to decide the losses at time t without knowing the outcome of the randomization of the algorithm at time t (but it can depend on the past randomization). This is enough to make the problem solvable. Moreover, it will also make the problem convex, allowing us to use any OCO algorithm on it.

First, let's write the problem in the original formulation. We set a discrete feasible set $V = \{\mathbf{e}_i\}_{i=1}^d$, where \mathbf{e}_i is the vector with all zeros but a 1 in the coordinate i . Our predictions and the competitor are from V . The losses are linear losses: $\ell_t(\mathbf{x}) = \langle \mathbf{g}_t, \mathbf{x}_t \rangle$, where we assume $0 \leq g_{t,i} \leq 1$, for $t = 1, \dots, T$ and $i = 1, \dots, d$. The regret is

$$\text{Regret}_T(\mathbf{e}_i) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{e}_i \rangle, \quad i = 1, \dots, d. \quad (6.8)$$

The only thing that makes this problem non-convex is the feasibility set, that is clearly a non-convex one.

Let's now see how the randomization makes this problem convex. Let's extend the feasible set to $V' = \{\mathbf{x} \in \mathbb{R}^d : x_i > 0, \|\mathbf{x}\|_1 = 1\}$. Note that $\mathbf{e}_i \in V'$. For this problem we can use an OCO algorithm to minimize the regret

$$\text{Regret}'_T(\mathbf{u}) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle, \forall \mathbf{u} \in V'.$$

Can we find a way to transform an upper bound to this regret to the one we care in (6.8)? One way is the following one: On each time step, construct a random variable i_t that is equal to i with probability $x_{t,i}$ for $i = 1, \dots, d$. Then, select the expert according to the outcome of i_t . Now, in expectation we have

$$\mathbb{E}[g_{t,i_t}] = \langle \mathbf{g}_t, \mathbf{x}_t \rangle,$$

and

$$\mathbb{E}[\text{Regret}_T(\mathbf{e}_i)] = \mathbb{E}[\text{Regret}'_T(\mathbf{e}_i)] = \mathbb{E} \left[\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{e}_i \rangle \right].$$

This means that we can minimize in expectation the non-convex regret with a randomized OCO algorithm. We can summarize this reasoning in Algorithm 6.3.

Algorithm 6.3 Learning with Expert Advice through Randomization

Require: $\mathbf{x}_1 \in \{\mathbf{x} \in \mathbb{R}^d : x_i > 0, \|\mathbf{x}\|_1 = 1\}$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Draw i_t according to $P(i_t = i) = x_{t,i}$
 - 3: Select expert i_t
 - 4: Observe all the experts' losses \mathbf{g}_t and pay the loss of the selected expert
 - 5: Update \mathbf{x}_t with an OCO algorithm with feasible set $\{\mathbf{x} \in \mathbb{R}^d : x_i > 0, \|\mathbf{x}\|_1 = 1\}$
 - 6: **end for**
-

For example, if we use EG as the OCO algorithm, setting $\mathbf{x}_1 = [1/d, \dots, 1/d]$, then we obtain the following update rule

$$x_{t+1,j} = \frac{x_{t,j} \exp(-\eta g_{t,j})}{\sum_{i=1}^d x_{t,i} \exp(-\eta g_{t,i})}, \quad j = 1, \dots, d$$

and the regret will be

$$\mathbb{E}[\text{Regret}_T(\mathbf{e}_i)] \leq \frac{\sqrt{2}}{2} L_\infty \sqrt{T \ln d}, \quad \forall \mathbf{e}_i.$$

It is worth stressing the importance of the result just obtained: We can design an algorithm that in expectation is close to the best expert in a set, *paying only a logarithmic penalty in the size of the set*.

In the future, we will see algorithms that achieve even the better regret guarantee of $O(\sqrt{T \cdot \text{KL}(\mathbf{u}; \mathbf{x}_1)})$, for any \mathbf{u} in the simplex. You should be able to convince yourself that no setting of η in EG allows to achieve such regret guarantee. Indeed, the algorithm will be based on a very different strategy.

6.8 History Bits

Mirror Descent (MD) was introduced by Nemirovsky and Yudin [1983] in the *batch* setting. The description of MD with Bregman divergence that I described here (with minor changes) was done by Beck and Teboulle [2003]. The minor changes are in decoupling the domain X of ψ from the feasibility set V . This allows to use functions ψ that do not satisfy the condition (6.4) but they satisfy (6.5). In the online setting, the mirror descent scheme was used for the first time by Warmuth and Jagota [1997].

Most of the online learning literature for OMD assumes ψ to be *Legendre* [see, e.g., Cesa-Bianchi and Lugosi, 2006] that corresponds to assuming (6.4). This condition allows to prove that $\nabla \psi_V^* = (\nabla \psi_V)^{-1}$. However, it turns

out that the Legendre condition is not necessary and we only need the function ψ to be differentiable on the predictions \mathbf{x}_t . In fact, we only need one of the two conditions in (6.4) or (6.5) to hold. Removing the Legendre assumption makes it easier to use OMD with different combinations of feasibility sets/Bregman divergences. So, I didn't introduce the concept of Legendre functions at all, relying instead on (a minor modification of) OMD as described by Beck and Teboulle [2003].

The concept of local norm was introduced in Abernethy et al. [2008].

The EG algorithm was introduced by Kivinen and Warmuth [1997], but not as a specific instantiation of OMD. Beck and Teboulle [2003] rediscover EG for the offline case as an example of Mirror Descent. Later, Cesa-Bianchi and Lugosi [2006] show that EG is just an instantiation of OMD. The p -norm algorithms for online prediction were originally introduced by Grove et al. [1997]. The trick to set $q = 2 \ln d$ is from Gentile and Littlestone [1999] (online learning) and apparently rediscovered in Ben-Tal et al. [2001] (optimization). The learning with expert setting was introduced in Littlestone and Warmuth [1994] and Vovk [1990]. The ideas in Algorithm 6.3 are based on the Multiplicative Weights algorithm [Littlestone and Warmuth, 1994] and the Hedge algorithm [Freund and Schapire, 1997]. By now, the literature on learning with expert is huge, with tons of variations over algorithms and settings.

6.9 Exercises

Problem 6.1. Prove that the ψ defined in Example 6.5 is 1-strongly convex w.r.t. the L_1 norm.

Problem 6.2. Derive a closed form update for OMD when using the ψ of Example 6.5 and $V = X$.

Problem 6.3. Prove the three-points equality for Bregman divergences in Lemma 6.6.

Problem 6.4. Let $A \in \mathbb{R}^{d \times d}$ a positive definite matrix. Define $\|\mathbf{x}\|_A^2 = \mathbf{x}^\top A \mathbf{x}$. Prove that $\frac{1}{2} \|\mathbf{x} - \mathbf{y}\|_A^2$ is the Bregman divergence $B_\psi(\mathbf{x}; \mathbf{y})$ associated with $\psi(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|_A^2$.

Problem 6.5. Find the conjugate function of $\psi(\mathbf{x}) = \sum_{i=1}^d x_i \ln x_i$ defined over $X = \{\mathbf{x} \in \mathbb{R}^d : x_i \geq 0, \|\mathbf{x}\|_1 = 1\}$.

Problem 6.6. We saw the Fenchel-Young inequality: $\langle \boldsymbol{\theta}, \mathbf{x} \rangle \leq f(\mathbf{x}) + f^*(\boldsymbol{\theta})$. Now, we want to show an equality, quantifying the gap in the inequality with a Bregman divergence term. Assume that f and f^* are differentiable, f strictly convex, and $\text{dom } f = \mathbb{R}^d$. Prove that

$$f(\mathbf{x}) + f^*(\boldsymbol{\theta}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle + B_f(\mathbf{x}; \nabla f^*(\boldsymbol{\theta})) .$$

Problem 6.7. In proof of Online Mirror Descent, we have the terms

$$-B_\psi(\mathbf{x}_{t+1}; \mathbf{x}_t) + \langle \eta_t \mathbf{g}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle .$$

Prove that they can be lower bounded by $B_\psi(\mathbf{x}_t; \mathbf{x}_{t+1})$.

Problem 6.8. Generalize the concept of strong convexity to Bregman functions, instead of norms, and prove a logarithmic regret guarantee for such functions using OMD.

Problem 6.9. Derive the EG update rule and regret bound in the case that the algorithm starts from an arbitrary vector \mathbf{x}_1 in the probability simplex.

Problem 6.10. In this problem, we will tackle Online Non-Convex Optimization. Assume that $V \subset \mathbb{R}^d$ is the feasible set and it is convex and bounded. The losses $\ell_t : \mathbb{R}^d \rightarrow [0, 1]$ are non-convex and 1-Lipschitz w.r.t. $\|\cdot\|_2$. Prove that there exists a randomized algorithm that achieves sublinear regret on this problem, assuming knowledge of the total number of rounds T . Hint: Aim for something like $\mathbb{E}[\text{Regret}_T(\mathbf{u})] \leq O(\sqrt{dT \ln T})$ and don't worry about efficiency of the algorithm.

Chapter 7

Follow-The-Regularized-Leader

Till now, we focused only on Online Subgradient Descent and its generalization, Online Mirror Descent, with a brief ad-hoc analysis of a Follow-The-Leader (FTL) analysis in the first chapter. In this chapter, we will extend FTL to a powerful and generic algorithm to do online convex optimization: **Follow-the-Regularized-Leader** (FTRL).

FTRL is a very intuitive algorithm: At each time step it will play the minimizer of the sum of the past losses *plus* a time-varying regularization. We will see that the regularization is needed to make the algorithm “more stable” with linear losses and avoid the jumping back and forth that we saw in Example 2.10.

7.1 Follow-the-Regularized-Leader

Algorithm 7.1 Follow-the-Regularized-Leader

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

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

As said above, in FTRL we output the minimizer of the regularized cumulative past losses. It should be clear that FTRL is not an algorithm, but rather a family of algorithms, in the same way as OMD is a family of algorithms.

Before analyzing the algorithm, let’s get some intuition on it. In OMD, we saw that the “state” of the algorithm is stored in the current iterate \mathbf{x}_t , in the sense that the next iterate \mathbf{x}_{t+1} depends on \mathbf{x}_t and the loss received at time t (the choice of the learning rate has only a little influence on the next iterate). Instead in FTRL, the next iterate \mathbf{x}_{t+1} depends on the entire history of losses received up to time t . This has an immediate consequence: In the case that V is bounded, OMD will only “remember” the last \mathbf{x}_t , and not the iterate before the projection. On the other hand, FTRL keeps in memory the entire history of the past, that in principle allows to recover the iterates before the projection in V .

This difference in behavior might make the reader think that FTRL is more computationally and memory expensive. And indeed it is! But, we will also see that there is a way to consider approximate losses that makes the algorithm as expensive as OMD, yet retaining strictly more information than OMD.

For FTRL, we prove a surprising result: an equality for the regret!

Lemma 7.1. *Let $V \subseteq \mathbb{R}^d$ be closed and non-empty. Denote by $F_t(\mathbf{x}) = \psi_t(\mathbf{x}) + \sum_{i=1}^{t-1} \ell_i(\mathbf{x})$. Assume that $\operatorname{argmin}_{\mathbf{x} \in V} F_t(\mathbf{x})$ is not empty and set $\mathbf{x}_t \in \operatorname{argmin}_{\mathbf{x} \in V} F_t(\mathbf{x})$. Then, for any \mathbf{u} , we have*

$$\sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) = \psi_{T+1}(\mathbf{u}) - \min_{\mathbf{x} \in V} \psi_1(\mathbf{x}) + \sum_{t=1}^T [F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t)] + F_{T+1}(\mathbf{x}_{T+1}) - F_{T+1}(\mathbf{u}).$$

Proof. Given that the terms $\ell_t(\mathbf{x}_t)$ are appearing at both side of the equality, we just have to verify that

$$-\sum_{t=1}^T \ell_t(\mathbf{u}) = \psi_{T+1}(\mathbf{u}) - \min_{\mathbf{x} \in V} \psi_1(\mathbf{x}) + \sum_{t=1}^T [F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1})] + F_{T+1}(\mathbf{x}_{T+1}) - F_{T+1}(\mathbf{u}).$$

Remembering that $F_1(\mathbf{x}_1) = \min_{\mathbf{x} \in V} \psi_1(\mathbf{x})$ and using the fact that the sum with F_t is telescopic, we have

$$-\sum_{t=1}^T \ell_t(\mathbf{u}) = \psi_{T+1}(\mathbf{u}) - F_1(\mathbf{x}_1) + F_1(\mathbf{x}_1) - F_{T+1}(\mathbf{x}_{T+1}) + F_{T+1}(\mathbf{x}_{T+1}) - F_{T+1}(\mathbf{u}) = \psi_{T+1}(\mathbf{u}) - F_{T+1}(\mathbf{u}),$$

that is true by the definition of F_{T+1} . \square

Remark 7.2. Note that we basically didn't assume anything on ℓ_t nor on ψ_t , the above equality holds even for non-convex losses and regularizers. Yet, solving the minimization problem at each step might be computationally infeasible.

Remark 7.3. Note that the left hand side of the equality in the theorem does not depend on ψ_{T+1} , so if needed we can set it to ψ_T .

Remark 7.4. The FTRL algorithm is invariant to any positive constant added to the regularizers, hence we can always state the regret guarantee with $\psi_t(\mathbf{u}) - \min_{\mathbf{x}} \psi_t(\mathbf{x})$ instead of $\psi_t(\mathbf{u})$. However, sometimes for clarity we will instead explicitly choose the regularizers such that their minimum is 0.

However, while surprising, the above equality is not yet a regret bound, because it is “implicit”. In fact, the losses are appearing on both sides of the equality.

Let's take a closer look at the equality. If $\mathbf{u} \in \text{dom } F_{t+1}$, we have that the sum of the last two terms on the r.h.s. is negative. On the other hand, the first two terms on the r.h.s. are similar to what we got in OMD. The interesting part is the sum of the terms $F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t)$. To give an intuition of what is going on, let's consider that case that the regularizer is constant over time, i.e., $\psi_t = \psi$. Hence, the terms in the sum can be rewritten as

$$F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t) = \psi(\mathbf{x}_t) + \sum_{i=1}^t \ell_i(\mathbf{x}_i) - \left(\psi(\mathbf{x}_{t+1}) + \sum_{i=1}^t \ell_i(\mathbf{x}_{t+1}) \right).$$

Hence, we are measuring the distance between the minimizer of the regularized losses (with two different regularizers) in two consecutive predictions of the algorithms. Roughly speaking, this term will be small if $\mathbf{x}_t \approx \mathbf{x}_{t+1}$ and the losses+regularization are “nice”. This should remind you exactly the OMD update, where we *constrain* \mathbf{x}_{t+1} to be close to \mathbf{x}_{t+1} . Instead, here the two predictions will be close one to the other if the minimizer of the regularized losses up to time t is close to the minimizer of the losses up to time $t+1$. So, like in OMD, the regularizer here will play the critical role of *stabilizing* the predictions, if the losses do not possess enough curvature.

To quantify this intuition, we need a property of strongly convex functions.

7.1.1 Convex Analysis Bits: Properties of Strongly Convex Functions

We will use the following lemma for strongly convex functions.

Lemma 7.5. Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ μ -strongly convex with respect to a norm $\|\cdot\|$. Then, for all $\mathbf{x}, \mathbf{y} \in \text{dom } f$, $\mathbf{g} \in \partial f(\mathbf{y})$, and $\mathbf{g}' \in \partial f(\mathbf{x})$, we have

$$f(\mathbf{x}) - f(\mathbf{y}) \leq \langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle + \frac{1}{2\mu} \|\mathbf{g} - \mathbf{g}'\|_*^2.$$

Proof. Define $\phi(\mathbf{z}) = f(\mathbf{z}) - \langle \mathbf{g}, \mathbf{z} \rangle$. Observe that $\mathbf{0} \in \partial\phi(\mathbf{y})$, hence \mathbf{y} is the minimizer of $\phi(\mathbf{z})$. Also, note that $\mathbf{g}' - \mathbf{g} \in \partial\phi(\mathbf{x})$. Hence, we can write

$$\begin{aligned}\phi(\mathbf{y}) &= \min_{\mathbf{z} \in \text{dom } \phi} \phi(\mathbf{z}) \\ &\geq \min_{\mathbf{z} \in \text{dom } \phi} \left(\phi(\mathbf{x}) + \langle \mathbf{g}' - \mathbf{g}, \mathbf{z} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{z} - \mathbf{x}\|^2 \right) \\ &\geq \inf_{\mathbf{z} \in \mathbb{R}^d} \left(\phi(\mathbf{x}) + \langle \mathbf{g}' - \mathbf{g}, \mathbf{z} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{z} - \mathbf{x}\|^2 \right) \\ &= \phi(\mathbf{x}) - \frac{1}{2\mu} \|\mathbf{g}' - \mathbf{g}\|_*^2,\end{aligned}$$

where the last step comes from the conjugate function of the squared norm, see Example 5.7. \square

Corollary 7.6. *Let $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ μ -strongly convex with respect to a norm $\|\cdot\|$. Let $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x})$. Then, for all $\mathbf{x} \in \text{dom } f$, and $\mathbf{g} \in \partial f(\mathbf{x})$, we have*

$$f(\mathbf{x}) - f(\mathbf{x}^*) \leq \frac{1}{2\mu} \|\mathbf{g}\|_*^2.$$

In words, the above lemma says that an upper bound to the suboptimality gap is proportional to the squared norm of the subgradient.

7.1.2 An Explicit Regret Bound for Strongly Convex Functions

We now state a Lemma quantifying the intuition on the “stability” of the predictions.

Lemma 7.7. *With the notation and assumptions of Lemma 7.1, assume that F_t is proper and λ_t -strongly convex w.r.t. $\|\cdot\|$, and ℓ_t proper and convex. Also, assume that $\partial\ell_t(\mathbf{x}_t)$ is non-empty. Then, we have*

$$F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t) \leq \frac{\|\mathbf{g}_t\|_*^2}{2\lambda_t} + \psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1}),$$

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

Proof. We have

$$\begin{aligned}F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t) &= (F_t(\mathbf{x}_t) + \ell_t(\mathbf{x}_t)) - (F_t(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_{t+1})) + \psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1}) \\ &\leq (F_t(\mathbf{x}_t) + \ell_t(\mathbf{x}_t)) - (F_t(\mathbf{x}_t^*) + \ell_t(\mathbf{x}_t^*)) + \psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1}) \\ &\leq \frac{\|\mathbf{g}_t'\|_*^2}{2\lambda_t} + \psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1}),\end{aligned}$$

where in the second inequality we used Corollary 7.6, the definition $\mathbf{x}_t^* := \operatorname{argmin}_{\mathbf{x} \in V} F_t(\mathbf{x}) + \ell_t(\mathbf{x})$, and $\mathbf{g}_t' \in \partial(F_t(\mathbf{x}_t) + \ell_t(\mathbf{x}_t))$. Observing that $\mathbf{x}_t = \operatorname{argmin}_{\mathbf{x} \in V} F_t(\mathbf{x})$, we have $\mathbf{0} \in \partial(F_t(\mathbf{x}_t) + \ell_t(\mathbf{x}_t))$ by Theorem 6.10. Hence, using Theorem 2.19, we can choose \mathbf{g}_t' such that we have $\mathbf{g}_t' \in \partial\ell_t(\mathbf{x}_t)$. \square

Let's see some immediate applications of FTRL.

Corollary 7.8. *Let ℓ_t a sequence of convex loss functions. Let $\psi : V \rightarrow \mathbb{R}$ a μ -strongly convex function w.r.t. $\|\cdot\|$. Set the sequence of regularizers as $\psi_t(\mathbf{x}) = \frac{1}{\eta_{t-1}}(\psi(\mathbf{x}) - \min_{\mathbf{z}} \psi(\mathbf{z}))$, where $\eta_{t+1} \leq \eta_t$, $t = 1, \dots, T$. Then, FTRL guarantees*

$$\sum_{t=1}^T \ell(\mathbf{x}_t) - \sum_{t=1}^T \ell(\mathbf{u}) \leq \frac{\psi(\mathbf{u}) - \min_{\mathbf{x} \in V} \psi(\mathbf{x})}{\eta_{T-1}} + \frac{1}{2\mu} \sum_{t=1}^T \eta_{t-1} \|\mathbf{g}_t\|_*^2,$$

for all $\mathbf{g}_t \in \partial\ell_t(\mathbf{x}_t)$. Moreover, if the functions ℓ_t are L -Lipschitz, setting $\eta_{t-1} = \frac{\alpha\sqrt{\mu}}{L\sqrt{t}}$ we get

$$\sum_{t=1}^T \ell(\mathbf{x}_t) - \sum_{t=1}^T \ell(\mathbf{u}) \leq \left(\frac{\psi(\mathbf{u}) - \min_{\mathbf{x}} \psi(\mathbf{x})}{\alpha} + \alpha \right) \frac{L\sqrt{T}}{\sqrt{\mu}}.$$

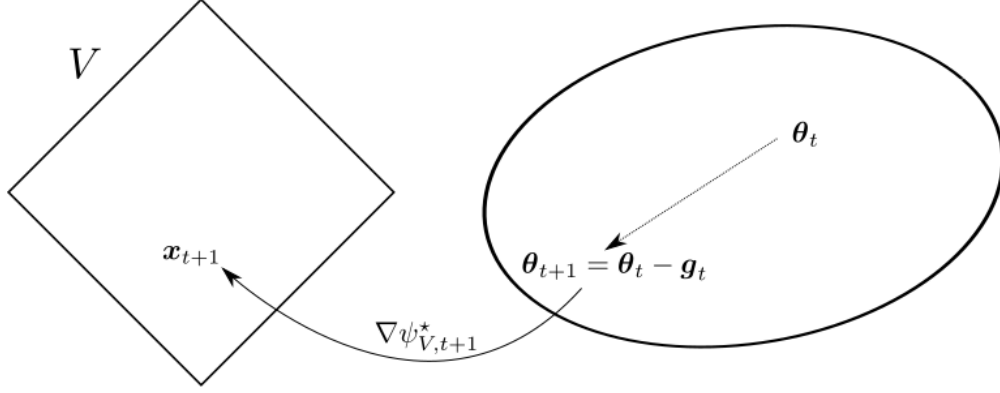


Figure 7.1: Dual mapping for FTRL with linear losses.

Proof. The corollary is immediate from Lemma 7.1, Lemma 7.7, and the observation that from the assumptions we have $\psi_t(\mathbf{x}) - \psi_{t+1}(\mathbf{x}) \leq 0, \forall \mathbf{x}$. We also set $\psi_{T+1} = \psi_T$, thanks to Remark 7.3. \square

This might look like the same regret guarantee of OMD, however here there is a very important difference: The last term contains a time-varying element (η_t) but the domain does not have to be bounded! Also, I used the regularizer $\frac{1}{\eta_{t-1}}\psi(\mathbf{x})$ and not $\frac{1}{\eta_t}\psi(\mathbf{x})$ to remind you another important difference: In OMD the learning rate η_t is chosen after receiving the subgradient \mathbf{g}_t while here you have to choose it before receiving it!

The another important difference is that here the update rule seems way more expensive than in OMD, because we need to solve an optimization problem at each step. However, it turns out we can use FTRL on *linearized losses* and obtain the same bound with the same computational complexity of OMD.

7.1.3 FTRL with Linearized Losses

If we consider that case in which the losses are linear, we have that the prediction of FTRL is

$$\mathbf{x}_{t+1} \in \operatorname{argmin}_{\mathbf{x} \in V} \psi_{t+1}(\mathbf{x}) + \sum_{i=1}^t \langle \mathbf{g}_i, \mathbf{x} \rangle = \operatorname{argmax}_{\mathbf{x} \in V} \left\langle -\sum_{i=1}^t \mathbf{g}_i, \mathbf{x} \right\rangle - \psi_{t+1}(\mathbf{x}).$$

Denote by $\psi_{V,t}(\mathbf{x}) = \psi_t(\mathbf{x}) + i_V(\mathbf{x})$. Now, if we assume $\psi_{V,t}$ to be proper, convex, and closed, using the Theorem 5.5, we have that $\mathbf{x}_{t+1} \in \partial \psi_{V,t+1}^*(-\sum_{i=1}^t \mathbf{g}_i)$. Moreover, if $\psi_{V,t+1}$ is strongly convex, we know that $\psi_{V,t+1}^*$ is differentiable and we get

$$\mathbf{x}_{t+1} = \nabla \psi_{V,t+1}^* \left(-\sum_{i=1}^t \mathbf{g}_i \right). \quad (7.1)$$

In turn, this update can be written in the following way

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \mathbf{g}_t, \\ \mathbf{x}_{t+1} &= \nabla \psi_{V,t+1}^*(\boldsymbol{\theta}_{t+1}). \end{aligned}$$

This corresponds to Figure 7.1.

Compare it to the mirror update of OMD, rewritten in a similar way:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \nabla \psi(\mathbf{x}_t) - \eta_t \mathbf{g}_t, \\ \mathbf{x}_{t+1} &= \nabla \psi_V^*(\boldsymbol{\theta}_{t+1}). \end{aligned}$$

They are very similar, but with important differences:

- In OMD, the state is kept in \mathbf{x}_t , so we need to transform it into a dual variable before making the update and then back to the primal variable.
- In FTRL with linear losses, the state is kept directly in the dual space, updated and then transformed in the primal variable. The primal variable is only used to predict, but not directly in the update.
- In OMD, the samples are weighted by the learning rates that is typically decreasing
- In FTRL with linear losses, all the subgradients have the same weight, but the regularizer is typically increasing over time.

Also, we will not loose anything in the bound! Indeed, we can run FTRL on the linearized losses $\tilde{\ell}_t(\mathbf{x}) = \langle \mathbf{g}_t, \mathbf{x} \rangle$, where $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$, guaranteeing exactly the same regret on the losses ℓ_t . The algorithm for such procedure is in Algorithm 7.2.

Algorithm 7.2 Follow-the-Regularized-Leader on Linearized Losses

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

- 1: **for** $t = 1$ **to** T **do**
- 2: Output $\mathbf{x}_t \in \operatorname{argmin}_{\mathbf{x} \in V} \psi_t(\mathbf{x}) + \sum_{i=1}^{t-1} \langle \mathbf{g}_i, \mathbf{x} \rangle$
- 3: Receive $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ and pay $\ell_t(\mathbf{x}_t)$
- 4: Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$
- 5: **end for**

In fact, using the definition of the subgradients and the assumptions of Corollary 7.8, we have

$$\operatorname{Regret}_T(\mathbf{u}) = \sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \sum_{t=1}^T (\tilde{\ell}_t(\mathbf{x}_t) - \tilde{\ell}_t(\mathbf{u})) \leq \frac{\psi(\mathbf{u}) - \min_{\mathbf{x} \in V} \psi(\mathbf{x})}{\eta_{T-1}} + \frac{1}{2\mu} \sum_{t=1}^T \eta_{t-1} \|\mathbf{g}_t\|_*^2, \forall \mathbf{u} \in V.$$

The only difference with respect to Corollary 7.8 is that here the \mathbf{g}_t are the specific ones we use in the algorithm, while in Corollary 7.8 the statement holds for any choice of the $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$.

In the next example, we can see the different behavior of FTRL and OMD.

Example 7.9. Consider $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 \leq 1\}$. With Online Subgradient Descent (OSD) with learning rate $\eta_t = \frac{1}{\sqrt{t}}$ and $\mathbf{x}_1 = \mathbf{0}$, the update is

$$\begin{aligned} \tilde{\mathbf{x}}_{t+1} &= \mathbf{x}_t - \frac{1}{\sqrt{t}} \mathbf{g}_t, \\ \mathbf{x}_{t+1} &= \tilde{\mathbf{x}}_{t+1} \min \left(\frac{1}{\|\tilde{\mathbf{x}}_{t+1}\|_2}, 1 \right). \end{aligned}$$

On the other hand in FTRL with linearized losses, we can use $\psi_t(\mathbf{x}) = \frac{\sqrt{t}}{2} \|\mathbf{x}\|_2^2 + i_V(\mathbf{x})$ and it is easy to verify that the update in (7.1) becomes

$$\begin{aligned} \tilde{\mathbf{x}}_{t+1} &= \frac{-\sum_{i=1}^t \mathbf{g}_i}{\sqrt{t}}, \\ \mathbf{x}_{t+1} &= \tilde{\mathbf{x}}_{t+1} \min \left(\frac{1}{\|\tilde{\mathbf{x}}_{t+1}\|_2}, 1 \right). \end{aligned}$$

While the regret guarantee would be the same for these two updates, from an intuitive point of view OMD seems to be loosing a lot of potential information due to the projection and the fact that we only memorize the projected iterate.

7.1.4 FTRL with Linearized Losses Can Be Equivalent to OMD

First, we see that even if FTRL and OMD seem very different, in certain cases they are equivalent. For example, consider that case that $V = X = \text{dom } \psi$. The output of OMD is

$$\mathbf{x}_{t+1} = \underset{\mathbf{x}}{\operatorname{argmin}} \langle \eta \mathbf{g}_t, \mathbf{x} \rangle + B_\psi(\mathbf{x}; \mathbf{x}_t) .$$

Assume that $\mathbf{x}_{t+1} \in \text{int dom } \psi$ for all $t = 1, \dots, T$. This implies that $\eta \mathbf{g}_t + \nabla \psi(\mathbf{x}_{t+1}) - \nabla \psi(\mathbf{x}_t) = \mathbf{0}$, that is $\nabla \psi(\mathbf{x}_{t+1}) = \nabla \psi(\mathbf{x}_t) - \eta \mathbf{g}_t$. Assuming $\mathbf{x}_1 = \min_{\mathbf{x} \in V} \psi(\mathbf{x})$, we have

$$\nabla \psi(\mathbf{x}_{t+1}) = -\eta \sum_{i=1}^t \mathbf{g}_i .$$

On the other hand, consider FTRL with linearized losses with regularizers $\psi_t = \frac{1}{\eta} \psi$, then

$$\mathbf{x}_{t+1} = \underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{\eta} \psi(\mathbf{x}) + \sum_{i=1}^t \langle \mathbf{g}_i, \mathbf{x} \rangle = \underset{\mathbf{x}}{\operatorname{argmin}} \psi(\mathbf{x}) + \eta \sum_{i=1}^t \langle \mathbf{g}_i, \mathbf{x} \rangle .$$

Assuming that $\mathbf{x}_{t+1} \in \text{int dom } \psi$, this implies that $\nabla \psi(\mathbf{x}_{t+1}) = -\eta \sum_{i=1}^t \mathbf{g}_i$. Further, assuming that $\nabla \psi$ is invertible, implies that the predictions of FTRL and OMD are the same.

This equivalence immediately gives us some intuition on the role of ψ in both algorithm: The same function is inducing the Bregman divergence, that is our similarity measure, and is the regularizer in FTRL. Moreover, the inverse of the growth rate of the regularizers in FTRL takes the role of the learning rate in OMD.

Example 7.10. Consider $\psi(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|_2^2$ and $V = \mathbb{R}^d$, then it satisfies the conditions above to have the predictions of OMD equal to the ones of FTRL.

7.2 Exponentiated Gradient with FTRL: No Need to know T

Let's see an example of an instantiation of FTRL with linearized losses to have the FTRL version of Exponentiated Gradient.

Let $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_1 = 1, x_i \geq 0\}$ and the sequence of loss functions $\ell_t : V \rightarrow \mathbb{R}$ be convex and L_∞ -Lipschitz w.r.t. the L_∞ -norm. Let $\psi : V \rightarrow \mathbb{R}^d$ defined as $\psi(\mathbf{x}) = \sum_{i=1}^d x_i \ln x_i$, where $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_1 = 1, x_i \geq 0\}$ and we define $0 \ln 0 = 0$. Set $\psi_t(\mathbf{x}) = \alpha L_\infty \sqrt{t} \psi(\mathbf{x})$, that is $\alpha L_\infty \sqrt{t}$ -strongly convex w.r.t. the L_1 norm, where $\alpha > 0$ is a parameter of the algorithm.

Given that the regularizers are strongly convex, we know that

$$\mathbf{x}_t = \nabla \psi_t^* \left(-\sum_{i=1}^{t-1} \mathbf{g}_i \right) .$$

We already saw that $\psi^*(\boldsymbol{\theta}) = \ln \left(\sum_{i=1}^d \exp(\theta_i) \right)$, that implies that $\psi_t^*(\boldsymbol{\theta}) = \alpha L_\infty \sqrt{t} \ln \left(\sum_{i=1}^d \exp \left(\frac{\theta_i}{\alpha L_\infty \sqrt{t}} \right) \right)$. So, we have that

$$x_{t,j} = \frac{\exp \left(-\frac{\sum_{k=1}^{t-1} g_{k,j}}{\alpha L_\infty \sqrt{t}} \right)}{\sum_{i=1}^d \exp \left(-\frac{\sum_{k=1}^{t-1} g_{k,i}}{\alpha L_\infty \sqrt{t}} \right)}, \quad j = 1, \dots, d .$$

Note that this is exactly the same update of EG based on OMD, but here we are effectively using time-varying learning rates.

We also get that the regret guarantee is

$$\begin{aligned} \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) &\leq L_\infty \sqrt{T} \left(\alpha \left(\sum_{i=1}^d u_i \ln u_i + \ln d \right) + \frac{1}{\alpha} \right) \\ &\leq L_\infty \sqrt{T} \left(\alpha \ln d + \frac{1}{\alpha} \right), \quad \forall \mathbf{u} \in V, \end{aligned}$$

where we used the fact that using $\psi_t = \alpha L_\infty \sqrt{t} \psi(\mathbf{x})$ and $\psi_t = \alpha L_\infty \sqrt{t} (\psi(\mathbf{x}) - \min_{\mathbf{x}} \psi(\mathbf{x}))$ are equivalent. Choosing $\alpha = \frac{1}{\sqrt{\ln d}}$. This regret guarantee is similar to the one we proved for OMD, but with an important difference: We do not have to know in advance the number of rounds T . In OMD a similar bound would be vacuous because it would depend on the $\max_{\mathbf{u}, \mathbf{x} \in V} B_\psi(\mathbf{u}; \mathbf{x})$ that is infinite.

7.3 AdaHedge*

In this section, we explain a variation of the EG/Hedge algorithm, called AdaHedge. The basic idea is to design an algorithm that is adaptive to the sum of the squared L_∞ norm of the losses, without any prior information on the range of the losses.

First, consider the case in which we use as constant regularizer the negative entropy $\psi_t(\mathbf{x}) = \lambda \sum_{i=1}^d x_i \ln x_i$, where $\lambda > 0$ will be determined in the following and V is the simplex in \mathbb{R}^d . Using FTRL with linear losses with this regularizer, we immediately obtain

$$\begin{aligned} \text{Regret}_T(\mathbf{u}) &\leq \lambda(\ln d + \sum_{i=1}^d u_i \ln u_i) + \sum_{t=1}^T (F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle) \\ &\leq \lambda \ln d + \sum_{t=1}^T (F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle), \end{aligned}$$

where we upper bounded the negative entropy of \mathbf{u} with 0. Using the strong convexity of the regularizer w.r.t. the L_1 norm and Lemma 7.7, we would further upper bound this as

$$\text{Regret}_T(\mathbf{u}) \leq \lambda \ln d + \sum_{t=1}^T \frac{\|\mathbf{g}_t\|_\infty^2}{2\lambda}.$$

This suggests that the optimal λ should be $\lambda = \sqrt{\frac{\sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2}{2 \ln d}}$. However, as we have seen in Section 4.2, this choice of any parameter of the algorithm is never feasible. Hence, exactly as we did in Section 4.2, we might think of using an online version of this choice

$$\psi_t(\mathbf{x}) = \lambda_t \sum_{i=1}^d x_i \ln x_i \quad \text{where} \quad \lambda_t = \frac{1}{\alpha} \sqrt{\sum_{i=1}^{t-1} \|\mathbf{g}_i\|_\infty^2}, \quad (7.2)$$

where $\alpha > 0$ is a constant that will be determined later. An important property of such choice is that it gives rise to an algorithm that is scale-free, that is its predictions \mathbf{x}_t are invariant from the scaling of the losses by any constant factor. This is easy to see because

$$x_{t,j} \propto \exp \left(-\frac{\alpha \sum_{i=1}^{t-1} g_{i,j}}{\sqrt{\sum_{i=1}^{t-1} \|\mathbf{g}_i\|_\infty^2}} \right), \quad \forall i = 1, \dots, d.$$

Note that this choice makes the regularizer non-decreasing over time and immediately gives us

$$\text{Regret}_T(\mathbf{u}) \leq \lambda_T \ln d + \sum_{t=1}^T \frac{\|\mathbf{g}_t\|_\infty^2}{2\lambda_t} = \frac{\ln d}{\alpha} \sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2} + \alpha \sum_{t=1}^T \frac{\|\mathbf{g}_t\|_\infty^2}{2\sqrt{\sum_{i=1}^{t-1} \|\mathbf{g}_i\|_\infty^2}}.$$

At this point, we might be tempted to use Lemma 4.13 to upper bound the sum in the upper bound, but unfortunately we cannot! Indeed, the denominator does not contain the term $\|\mathbf{g}_t\|_\infty^2$. We might add a constant to λ_t , but that would destroy the scale-freeness of the algorithm. However, it turns out that we can still prove our bound without any change

to the regularizer. The key observation is that we can bound the term $F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle$ in two different ways. One way is using Lemma 7.7. The other one is

$$\begin{aligned}
F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle &\leq F_t(\mathbf{x}_{t+1}) - F_{t+1}(\mathbf{x}_{t+1}) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle \\
&= \psi_t(\mathbf{x}_{t+1}) + \sum_i^{t-1} \langle \mathbf{g}_i, \mathbf{x}_{t+1} \rangle - \psi_{t+1}(\mathbf{x}_{t+1}) - \sum_{i=1}^t \langle \mathbf{g}_i, \mathbf{x}_{t+1} \rangle \\
&\leq -\langle \mathbf{g}_t, \mathbf{x}_{t+1} \rangle + \langle \mathbf{g}_t, \mathbf{x}_t \rangle \\
&\leq 2\|\mathbf{g}_t\|_\infty,
\end{aligned}$$

where we used the definition of \mathbf{x}_{t+1} and the fact that the regularizer is non-decreasing over time. So, we can now write

$$\begin{aligned}
\sum_{t=1}^T F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle &\leq \sum_{t=1}^T \min \left(\frac{\alpha \|\mathbf{g}_t\|_\infty^2}{2\sqrt{\sum_{i=1}^{t-1} \|\mathbf{g}_i\|_\infty^2}}, 2\|\mathbf{g}_t\|_\infty \right) \\
&= 2 \sum_{t=1}^T \sqrt{\min \left(\frac{\alpha^2 \|\mathbf{g}_t\|_\infty^4}{16 \sum_{i=1}^{t-1} \|\mathbf{g}_i\|_\infty^2}, \|\mathbf{g}_t\|_\infty^2 \right)} \\
&\leq 2 \sum_{t=1}^T \sqrt{\frac{2}{\frac{16 \sum_{i=1}^{t-1} \|\mathbf{g}_i\|_\infty^2}{\alpha^2 \|\mathbf{g}_t\|_\infty^4} + \frac{1}{\|\mathbf{g}_t\|_\infty^2}}} \\
&= 2 \sum_{t=1}^T \sqrt{2} \frac{\alpha \|\mathbf{g}_t\|_\infty^2}{\sqrt{\alpha^2 \|\mathbf{g}_t\|_\infty^2 + 16 \sum_{i=1}^{t-1} \|\mathbf{g}_i\|_\infty^2}},
\end{aligned}$$

where we used the fact that the minimum between two numbers is less than their harmonic mean. Assuming $\alpha > 4$ and using Lemma 4.13, we have

$$\sum_{t=1}^T F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle \leq \frac{\sqrt{2}}{2} \sum_{t=1}^T \frac{\alpha \|\mathbf{g}_t\|_\infty^2}{\sqrt{\sum_{i=1}^t \|\mathbf{g}_i\|_\infty^2}} \leq \sqrt{2 \sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2}$$

and

$$\text{Regret}_T(\mathbf{u}) \leq \left(\frac{\ln d}{\alpha} + \alpha\sqrt{2} \right) \sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2}. \quad (7.3)$$

The bound and the assumption on α suggest to set $\alpha = \min(4, 2^{-1/4} \sqrt{\ln d})$. To summarize, we obtained a scale-free algorithm with regret bound $O(\sqrt{\ln d \sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2})$.

We might consider ourselves happy, but there is a clear problem in the above algorithm: the choice of λ_t in the time-varying regularizer strictly depend on our upper bound. So, a loose bound will result in a poor choice of the regularization! In general, everytime we use a part of the proof in the design of an algorithm we cannot expect an exciting empirical performance, unless our upper bound was really tight. So, can we design a better regularizer? Well, we need a better upper bound!

Let's consider a generic regularizer $\psi_t(\mathbf{x}) = \lambda_t \psi(\mathbf{x})$ and its corresponding FTRL with linear losses regret upper bound

$$\text{Regret}_T(\mathbf{u}) \leq \lambda_T(\psi(\mathbf{u}) - \inf_{\mathbf{x} \in V} \psi(\mathbf{x})) + \sum_{t=1}^T (F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle),$$

where we assume λ_t to be non-decreasing in time.

Now, observe that the sum is unlikely to disappear for this kind of algorithms, so we could try to make the term $\lambda_T(\psi(\mathbf{u}) - \inf_{\mathbf{x} \in V} \psi(\mathbf{x}))$ of the same order of the sum. So, we would like to set λ_t of the same order of $\sum_{i=1}^t (F_i(\mathbf{x}_i) - F_{i+1}(\mathbf{x}_{i+1}) + \langle \mathbf{g}_i, \mathbf{x}_i \rangle)$. However, this approach would cause an annoying recurrence. So, using the fact that λ_t is non-decreasing, let's upper bound the terms in the sum just a little bit:

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

Now, we can set $\lambda_t = \frac{1}{\alpha^2} \sum_{i=1}^{t-1} \delta_i$ for $t \geq 2$, $\lambda_1 = 0$, and $\mathbf{x}_1 = \operatorname{argmin}_{\mathbf{x} \in V} \psi(\mathbf{x})$. This immediately implies that

$$\operatorname{Regret}_T(\mathbf{u}) \leq \left(\psi(\mathbf{u}) - \inf_{\mathbf{x} \in V} \psi(\mathbf{x}) + \alpha^2 \right) \lambda_{T+1}.$$

Setting ψ to be equal to the negative entropy, we get an algorithm known as AdaHedge.

With this choice of the regularizer, we can simplify a bit the expression of δ_t . For $t = 1$, we have $\delta_1 = \langle \mathbf{g}_1, \mathbf{x}_1 \rangle - \min_{\mathbf{x} \in V} \langle \mathbf{g}_1, \mathbf{x} \rangle$. Instead, for $t \geq 2$, using the properties of the Fenchel conjugates, we have that

$$\delta_t = \lambda_t \ln \frac{\sum_{j=1}^d \exp(\theta_{t+1,j}/\lambda_t)}{\sum_{j=1}^d \exp(\theta_{t,j}/\lambda_t)} + \langle \mathbf{g}_t, \mathbf{x}_t \rangle = \lambda_t \ln \left(\sum_{j=1}^d x_{t,j} \exp(-g_{t,j}/\lambda_t) \right) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle.$$

Overall, we get the pseudo-code of AdaHedge in Algorithm 7.3.

Algorithm 7.3 AdaHedge

Require: $\alpha > 0$

- 1: $\lambda_1 = 0$
 - 2: $\mathbf{x}_1 = [1/d, \dots, 1/d] \in \mathbb{R}^d$
 - 3: $\boldsymbol{\theta}_1 = \mathbf{0} \in \mathbb{R}^d$
 - 4: **for** $t = 1$ **to** T **do**
 - 5: Output \mathbf{x}_t
 - 6: Receive $\mathbf{g}_t \in \mathbb{R}^d$ and pay $\langle \mathbf{g}_t, \mathbf{x}_t \rangle$
 - 7: Update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{g}_t$
 - 8: Set $\delta_t = \begin{cases} \langle \mathbf{g}_1, \mathbf{x}_1 \rangle - \min_{j=1, \dots, d} g_{1,j}, & t = 1 \\ \lambda_t \ln \left(\sum_{j=1}^d x_{t,j} \exp(-g_{t,j}/\lambda_t) \right) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle & \text{otherwise} \end{cases}$
 - 9: Update $\lambda_{t+1} = \lambda_t + \frac{1}{\alpha^2} \delta_t$
 - 10: Update $x_{t+1,j} \propto \exp\left(\frac{\theta_{t+1,j}}{\lambda_{t+1}}\right), j = 1, \dots, d$
 - 11: **end for**
-

So, now we need an upper bound for λ_T . Observe that $\lambda_{t+1} = \lambda_t + \frac{1}{\alpha^2} \delta_t$. Moreover, as we have done before, we can upper bound δ_t in two different ways. In fact, from Lemma 7.7, we have $\delta_t \leq \frac{\|\mathbf{g}_t\|_\infty^2}{2\lambda_t}$ for $t \geq 2$. Also, denoting by $\tilde{\mathbf{x}}_{t+1} = \operatorname{argmin}_{\mathbf{x} \in V} \lambda_t\psi(\mathbf{x}) + \sum_{i=1}^t \langle \mathbf{g}_i, \mathbf{x} \rangle$, we have

$$\begin{aligned} \delta_t &= F_t(\mathbf{x}_t) - \min_{\mathbf{x} \in V} \left(\lambda_t\psi(\mathbf{x}) + \sum_{i=1}^t \langle \mathbf{g}_i, \mathbf{x} \rangle \right) + \langle \mathbf{g}_t, \mathbf{x}_t \rangle = F_t(\mathbf{x}_t) - \lambda_t\psi(\tilde{\mathbf{x}}_{t+1}) - \sum_{i=1}^t \langle \mathbf{g}_i, \tilde{\mathbf{x}}_{t+1} \rangle + \langle \mathbf{g}_t, \mathbf{x}_t \rangle \\ &\leq F_t(\tilde{\mathbf{x}}_{t+1}) - \lambda_t\psi(\tilde{\mathbf{x}}_{t+1}) - \sum_{i=1}^t \langle \mathbf{g}_i, \tilde{\mathbf{x}}_{t+1} \rangle + \langle \mathbf{g}_t, \mathbf{x}_t \rangle = -\langle \mathbf{g}_t, \tilde{\mathbf{x}}_{t+1} \rangle + \langle \mathbf{g}_t, \mathbf{x}_t \rangle \leq 2\|\mathbf{g}_t\|_\infty. \end{aligned}$$

Hence, we have

$$\begin{aligned}\lambda_1 &= 0, \\ \lambda_2 &\leq 2\alpha\|\mathbf{g}_1\|_\infty, \\ \lambda_{t+1} &= \lambda_t + \alpha\delta_t \leq \lambda_t + \frac{1}{\alpha^2} \min\left(2\|\mathbf{g}_t\|_\infty, \frac{\|\mathbf{g}_t\|_\infty^2}{2\lambda_t}\right), \quad \forall t \geq 2.\end{aligned}$$

We can solve this recurrence using the following Lemma, where $\Delta_t = \lambda_t$ and $a_t = \|\mathbf{g}_t\|_\infty$.

Lemma 7.11. *Let $\{a_t\}_{t=1}^\infty$ be any sequence of non-negative real numbers. Suppose that $\{\Delta_t\}_{t=0}^\infty$ is a sequence of non-negative real numbers satisfying*

$$\Delta_0 = 0 \quad \text{and} \quad \Delta_t \leq \Delta_{t-1} + \min\left\{ba_t, c\frac{a_t^2}{2\Delta_{t-1}}\right\} \quad \text{for any } t \geq 1.$$

Then, for any $T \geq 0$, $\Delta_T \leq \sqrt{(b^2 + c) \sum_{t=1}^T a_t^2}$.

Proof. Observe that

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

We bound each term in the sum separately. The left term of the minimum inequality in the definition of Δ_t gives $(\Delta_t - \Delta_{t-1})^2 \leq b^2 a_t^2$, while the right term gives $2(\Delta_t - \Delta_{t-1})\Delta_{t-1} \leq c a_t^2$. So, we conclude $\Delta_T^2 \leq (b^2 + c) \sum_{t=1}^T a_t^2$. \square

So, overall we got

$$\text{Regret}_T(\mathbf{u}) \leq \left(\psi(\mathbf{u}) - \inf_{\mathbf{x} \in V} \psi(\mathbf{x}) + \alpha^2\right) \lambda_{T+1} \leq \left(\frac{\psi(\mathbf{u}) - \inf_{\mathbf{x} \in V} \psi(\mathbf{x})}{\alpha^2} + 1\right) \sqrt{(4 + \alpha^2) \sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2},$$

and setting $\alpha = \sqrt{\ln d}$, we have

$$\text{Regret}_T(\mathbf{u}) \leq 2\sqrt{(4 + \ln d) \sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2}.$$

Note that this is roughly the same regret in (7.3), but the very important difference is that this new regret bound depends on the much tighter quantity λ_{T+1} , that we upper bounded with $O(\sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2})$, but in general will be much smaller than that. For example, λ_{T+1} can be upper bounded using the tighter local norms, see Section 10.1.1. Instead, in the first solution, the regret will always be dominated by the term $\sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2}$ because we explicitly use it in the regularizer.

There is an important lesson to be learned from AdaHedge: the regret is not the full story and algorithms with the same worst-case guarantee can exhibit vastly different empirical behaviours. Unfortunately, this message is rarely heard and there is a part of the community that focuses too much on the worst-case guarantee rather than on the empirical performance. Even worse, sometimes people favor algorithms with a “more elegant analysis” completely ignoring the likely worse empirical performance.

7.4 Composite Losses

Let’s now see a variant of the linearization of the losses: *partial linearization of composite losses*.

Suppose that the losses we receive are composed by two terms: one convex function changing over time and another part is fixed and known. These losses are called *composite*. For example, we might have $\ell_t(\mathbf{x}) = \tilde{\ell}_t(\mathbf{x}) +$

$\lambda\|\mathbf{x}\|_1$. Using the linearization, we might just take the subgradient of ℓ_t . However, in this particular case, we might lose the ability of the L_1 norm to produce sparse solutions.

There is a better way to deal with these kind of losses: Move the constant part of the loss inside the regularization term. In this way, that part will not be linearized but used exactly in the argmin of the update. Assuming that the argmin is still easily computable, you can always expect better performance from this approach. In particular, in the case of adding an L_1 norm to the losses, you will be predicting in each step with the solution of an L_1 regularized optimization problem.

Practically speaking, in the example above, we will define $\psi_t(\mathbf{x}) = L\sqrt{t}(\psi(\mathbf{x}) - \min_{\mathbf{y}} \psi(\mathbf{y})) + \lambda t\|\mathbf{x}\|_1$, where we assume ψ to be 1-strongly convex and the losses $\tilde{\ell}_t$ be L -Lipschitz. Note that we use at time t a term $\lambda t\|\mathbf{x}\|_1$ because we anticipate the next term in the next round. Given that $\psi_t(\mathbf{x}) + \sum_{i=1}^{t-1} \ell_i(\mathbf{x})$ is $L\sqrt{t}$ -strongly convex, using Lemma 7.7, we have

$$\begin{aligned}
& \sum_{t=1}^T \tilde{\ell}_t(\mathbf{x}_t) - \sum_{t=1}^T \tilde{\ell}_t(\mathbf{u}) \\
& \leq \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \\
& \leq L\sqrt{T} \left(\psi(\mathbf{u}) - \min_{\mathbf{x}} \psi(\mathbf{x}) \right) + \lambda T\|\mathbf{u}\|_1 - \min_{\mathbf{x}} (\lambda\|\mathbf{x}\|_1 + \psi(\mathbf{x}) - \min_{\mathbf{y}} \psi(\mathbf{y})) + \frac{1}{2} \sum_{t=1}^T \frac{\|\mathbf{g}_t\|_*^2}{L\sqrt{t}} - \lambda \sum_{t=1}^{T-1} \|\mathbf{x}_{t+1}\|_1 \\
& \leq L\sqrt{T} \left(1 + \psi(\mathbf{u}) - \min_{\mathbf{x}} \psi(\mathbf{x}) \right) + \lambda T\|\mathbf{u}\|_1 - \lambda\|\mathbf{x}_1\|_1 - \lambda \sum_{t=1}^{T-1} \|\mathbf{x}_{t+1}\|_1 \\
& = L\sqrt{T} \left(1 + \psi(\mathbf{u}) - \min_{\mathbf{x}} \psi(\mathbf{x}) \right) + \lambda T\|\mathbf{u}\|_1 - \lambda \sum_{t=1}^T \|\mathbf{x}_t\|_1,
\end{aligned}$$

where $\mathbf{g}_t \in \partial \tilde{\ell}_t(\mathbf{x}_t)$. Reordering the terms, we have

$$\sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) = \sum_{t=1}^T (\lambda\|\mathbf{x}_t\|_1 + \tilde{\ell}_t(\mathbf{x}_t)) - \sum_{t=1}^T (\lambda\|\mathbf{u}\|_1 + \tilde{\ell}_t(\mathbf{u})) \leq L\sqrt{T} \left(\psi(\mathbf{u}) - \min_{\mathbf{x}} \psi(\mathbf{x}) + 1 \right).$$

Example 7.12. Let's also take a look at the update rule in that case that $\psi(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2$ and we get composite losses with the L_1 norm. We have

$$\mathbf{x}_t = \operatorname{argmin}_{\mathbf{x}} \frac{L\sqrt{t}}{2}\|\mathbf{x}\|_2^2 + \lambda t\|\mathbf{x}\|_1 + \sum_{i=1}^{t-1} \langle \mathbf{g}_i, \mathbf{x} \rangle.$$

We can solve this problem observing that the minimization decomposes over each coordinate of \mathbf{x} . Denote by $\boldsymbol{\theta}_t = \sum_{i=1}^{t-1} \mathbf{g}_i$. Hence, we know from first-order optimality condition that $x_{t,i}$ is the solution for the coordinate i iff there exists $v_i \in \partial|x_{t,i}|$ such that

$$L\sqrt{t}x_{t,i} + \lambda tv_i + \theta_{t,i} = 0$$

Consider the 3 different cases:

- $|\theta_{t,i}| \leq \lambda t$, then $x_{t,i} = 0$ and $v_i = -\frac{\theta_{t,i}}{\lambda t}$.
- $\theta_{t,i} > \lambda t$, then $x_{t,i} = -\frac{\theta_{t,i} - \lambda t}{L\sqrt{t}}$ and $v_i = -1$.
- $\theta_{t,i} < -\lambda t$, then $x_{t,i} = -\frac{\theta_{t,i} + \lambda t}{L\sqrt{t}}$ and $v_i = 1$.

So, overall we have

$$x_{t,i} = -\frac{\operatorname{sign}(\theta_{t,i}) \max(|\theta_{t,i}| - \lambda t, 0)}{L\sqrt{t}}.$$

Observe as this update will produce sparse solutions, while just taking the subgradient of the L_1 norm would have never produced sparse predictions.

Remark 7.13 (Proximal operators). *In the example above, we calculated something like*

$$\operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{x}\|_1 - \langle \mathbf{v}, \mathbf{x} \rangle + \frac{1}{2} \|\mathbf{x}\|_2^2 = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{x}\|_1 - \langle \mathbf{v}, \mathbf{x} \rangle + \frac{1}{2} \|\mathbf{x}\|_2^2 + \frac{1}{2} \|\mathbf{v}\|_2^2 = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{x}\|_1 + \frac{1}{2} \|\mathbf{x} - \mathbf{v}\|_2^2.$$

This operation is known in the optimization literature as Proximal Operator of the L_1 norm. In general, a proximal operator of a convex, proper, and closed function $f : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ is defined as

$$\operatorname{Prox}_f(\mathbf{v}) = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) + \frac{1}{2} \|\mathbf{x} - \mathbf{v}\|_2^2.$$

Proximal operators are used in optimization in the same way as we used it: They allow to minimize the entire function rather a linear approximation of it. Also, proximal operators generalizes the concept of Euclidean projection. Indeed, $\operatorname{Prox}_{i_V}(\mathbf{v}) = \Pi_V(\mathbf{v})$.

7.5 FTRL with Strongly Convex Functions

Let's now go back to the FTRL regret bound and let's see if you can strengthen it in the case that the regularizer is *proximal*, that is it satisfies that $\mathbf{x}_t \in \operatorname{argmin}_{\mathbf{x}} \psi_{t+1}(\mathbf{x}) - \psi_t(\mathbf{x})$.

Lemma 7.14. *Denote by $F_t(\mathbf{x}) = \psi_t(\mathbf{x}) + \sum_{i=1}^{t-1} \ell_i(\mathbf{x})$. Assume that $\operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} F_t(\mathbf{x})$ is not empty and set $\mathbf{x}_t \in \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} F_t(\mathbf{x})$. Also, assume that F_t is λ_t -strongly convex w.r.t. $\|\cdot\|$ and ℓ_t convex, and the regularizer is such that $\mathbf{x}_t \in \operatorname{argmin}_{\mathbf{x}} \psi_{t+1}(\mathbf{x}) - \psi_t(\mathbf{x})$. Also, assume that $\partial \ell_t(\mathbf{x}_t)$ is non-empty. Then, we have*

$$F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t) \leq \frac{\|\mathbf{g}_t\|_*^2}{2\lambda_{t+1}} + \psi_t(\mathbf{x}_t) - \psi_{t+1}(\mathbf{x}_t), \quad \forall \mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t).$$

Proof. We have

$$\begin{aligned} & F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t) \\ &= (F_t(\mathbf{x}_t) + \ell_t(\mathbf{x}_t) + \psi_{t+1}(\mathbf{x}_t) - \psi_t(\mathbf{x}_t)) - F_{t+1}(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_t) + \psi_t(\mathbf{x}_t) \\ &= F_{t+1}(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_t) + \psi_t(\mathbf{x}_t) \\ &\leq \frac{\|\mathbf{g}'_t\|_*^2}{2\lambda_{t+1}} - \psi_{t+1}(\mathbf{x}_t) + \psi_t(\mathbf{x}_t), \end{aligned}$$

where in the second inequality we used Corollary 7.6, the fact that $\mathbf{x}_{t+1} = \operatorname{argmin}_{\mathbf{x}} F_{t+1}(\mathbf{x})$, and $\mathbf{g}'_t \in \partial F_{t+1}(\mathbf{x}_t)$. Observing that from the proximal property, we have that $\mathbf{x}_t = \operatorname{argmin}_{\mathbf{x}} F_t(\mathbf{x}) + \psi_{t+1}(\mathbf{x}) - \psi_t(\mathbf{x})$, $\mathbf{0} \in \partial(F_t(\mathbf{x}_t) + \psi_{t+1}(\mathbf{x}_t) - \psi_t(\mathbf{x}_t))$. Hence, using the theorem of the subdifferential of sum of functions, and remembering that $F_{t+1}(\mathbf{x}) = F_t(\mathbf{x}) + \ell_t(\mathbf{x}) + \psi_{t+1}(\mathbf{x}) - \psi_t(\mathbf{x})$, we can choose \mathbf{g}'_t such that we have $\mathbf{g}'_t \in \partial \ell_t(\mathbf{x}_t)$. \square

Remark 7.15. *Note that a constant regularizer is proximal because any point is the minimizer of the zero function. On the other hand, a constant regularizer makes the two Lemmas the same, unless the loss functions contribute to the total strong convexity.*

We will now use the above lemma to prove a logarithmic regret bound for strongly convex losses.

Corollary 7.16. *Let ℓ_t be μ_t strongly convex w.r.t. $\|\cdot\|$, for $t = 1, \dots, T$. Set the sequence of regularizers to zero. Then, FTRL guarantees a regret of*

$$\sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \frac{1}{2} \sum_{t=1}^T \frac{\|\mathbf{g}_t\|_*^2}{\sum_{i=1}^t \mu_i}, \quad \forall \mathbf{u} \in \mathbb{R}^d, \quad \forall \mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t).$$

The above regret guarantee is the same of OMD over strongly convex losses, but here we do not need to know the strong convexity of the losses. In fact, we just need to output the minimizer over the past losses. However, as we noticed last time, this might be undesirable because now each update is an optimization problem.

Hence, we can again use the idea of replacing the losses with an easy *surrogate*. In the Lipschitz case, it made sense to use linear losses. However, here you can do better and use *quadratic* losses, because the losses are strongly convex. So, we can run FTRL on the quadratic losses $\tilde{\ell}_t(\mathbf{x}) = \langle \mathbf{g}_t, \mathbf{x} \rangle + \frac{\mu_t}{2} \|\mathbf{x} - \mathbf{x}_t\|^2$, where $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$. The algorithm would be the following one:

Algorithm 7.4 Follow-the-Regularized-Leader on “Quadratized” Losses

Require: A sequence of regularizers $\psi_1, \dots, \psi_T : \mathbb{R}^d \rightarrow (-\infty, +\infty]$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Output $\mathbf{x}_t \in \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} \psi_t(\mathbf{x}) + \sum_{i=1}^{t-1} (\langle \mathbf{g}_i, \mathbf{x} \rangle + \frac{\mu_i}{2} \|\mathbf{x} - \mathbf{x}_i\|^2)$
 - 3: Receive $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ and pay $\ell_t(\mathbf{x}_t)$
 - 4: Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$
 - 5: Calculate the strong convexity μ_t of ℓ_t
 - 6: **end for**
-

To see why this is a good idea, consider the case that the losses are strongly convex w.r.t. the L_2 norm. The update now becomes:

$$\mathbf{x}_t = \operatorname{argmin}_{\mathbf{x}} \sum_{i=1}^{t-1} \left(\langle \mathbf{g}_i, \mathbf{x} \rangle + \frac{\mu_i}{2} \|\mathbf{x} - \mathbf{x}_i\|^2 \right) = \frac{\sum_{i=1}^{t-1} (\mu_i \mathbf{x}_i - \mathbf{g}_i)}{\sum_{i=1}^{t-1} \mu_i}. \quad (7.4)$$

Moreover, we will get exactly the same regret bound as in Corollary 7.16, with the only difference that here the guarantee holds for a specific choice of the \mathbf{g}_t rather than for any subgradient in $\partial \ell_t(\mathbf{x}_t)$.

Example 7.17. Going back to the example in the first chapter, where $V = [0, 1]$ and $\ell_t(\mathbf{x}) = (x - y_t)^2$ are strongly convex, we now see immediately that FTRL without a regularizer, that is Follow the Leader, gives logarithmic regret. Note that in this case the losses were defined only over V , so that the minimization is carried over V .

7.6 Online Newton Step

Last time, we saw that the notion of strong convexity allows us to build quadratic surrogate loss functions, on which FTRL has smaller regret. Can we find a more general notion of strong convexity that allows to get a small regret for a larger class of functions? We can start from strong convexity and try to generalize it. So, instead of asking that the function is strongly convex w.r.t. a norm, we might be happy requiring strong convexity holds in a particular point w.r.t. a norm that depends on the points itself.

In particular, we can require that for each loss ℓ_t and for all $\mathbf{x} \in \operatorname{dom} \ell_t$ the following holds

$$\forall \mathbf{x} \in V, \mathbf{g} \in \partial \ell_t(\mathbf{x}), \exists A_t \in \mathbb{R}^{d \times d} : A_t \succeq 0, \ell_t(\mathbf{y}) \geq \ell_t(\mathbf{x}) + \langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle + \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|_{A_t}^2, \forall \mathbf{y} \in V,$$

where $\|\mathbf{x}\|_{A_t}$ is defined as $\sqrt{\mathbf{x}^\top A_t \mathbf{x}}$. Note that this is a weaker property than strong convexity because A_t depends on \mathbf{x} . On the other hand, in the definition of strong convexity we want the last term to be the same norm (or Bregman divergence in the more general formulation) everywhere in the space.

The rationale of this new definition is that it still allows us to build surrogate loss functions, but without requiring to have strong convexity over the entire space. Hence, we can think to use FTRL on the surrogate losses

$$\tilde{\ell}_t(\mathbf{x}) = \langle \mathbf{g}_t, \mathbf{x} \rangle$$

and the proximal regularizers $\psi_t(\mathbf{x}) = \frac{\lambda}{2} \|\mathbf{x}\|^2 + \frac{1}{2} \sum_{i=1}^{t-1} \|\mathbf{x}_i - \mathbf{x}\|_{A_i}^2$, where $\lambda > 0$. We will denote by $S_t = \lambda I_d + \sum_{i=1}^t A_i$.

Remark 7.18. Note that $\|\mathbf{x}\|_{S_t} := \sqrt{\mathbf{x}^\top S_t \mathbf{x}}$ is a norm because S_t is Positive Definite (PD) and $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top S_t \mathbf{x}$ is 1-strongly convex w.r.t. $\|\cdot\|_{S_t}$ defined as $\|\mathbf{x}\|_{S_t} = \sqrt{\mathbf{x}^\top S_t \mathbf{x}}$ (because the Hessian is S_t and $\mathbf{x}^\top \nabla^2 f(\mathbf{y}) \mathbf{x} = \|\mathbf{x}\|_{A_t}^2$). Also, the dual norm of $\|\cdot\|_{S_t}$ is $\|\cdot\|_{S_t^{-1}}$.

From the above remark, we have that the regularizer $\psi_t(\mathbf{x})$ is 1-strongly convex w.r.t $\|\cdot\|_{S_{t-1}}$. Hence, using the FTRL regret equality in Lemma 7.1 and Lemma 7.14 for proximal regularizers, we immediately get the following guarantee

$$\begin{aligned} \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle &= \sum_{t=1}^T \tilde{\ell}_t(\mathbf{x}_t) - \sum_{t=1}^T \tilde{\ell}_t(\mathbf{u}) \\ &\leq \psi_{T+1}(\mathbf{u}) + \frac{1}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_{S_t^{-1}}^2 + \sum_{t=1}^T (\psi_t(\mathbf{x}_t) - \psi_{t+1}(\mathbf{x}_t)) \\ &= \frac{\lambda}{2} \|\mathbf{u}\|^2 + \frac{1}{2} \sum_{t=1}^T \|\mathbf{x}_t - \mathbf{u}\|_{A_t}^2 + \frac{1}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_{S_t^{-1}}^2. \end{aligned}$$

So, reordering the terms we have

$$\sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \sum_{t=1}^T \left(\langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle - \frac{1}{2} \|\mathbf{u} - \mathbf{x}_t\|_{A_t}^2 \right) \leq \frac{\lambda}{2} \|\mathbf{u}\|^2 + \frac{1}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_{S_t^{-1}}^2. \quad (7.5)$$

Note how the proof and the algorithm mirror what we did for FTRL with strongly convex losses in Section 7.5.

Remark 7.19. It is possible to generalize our Lemma of FTRL for proximal regularizers to hold in this generalized notion of strong convexity. This would allow to get exactly the same bound running FTRL over the original losses with regularizer $\psi(\mathbf{x}) = \frac{\lambda}{2} \|\mathbf{x}\|_2^2$.

Let's now see a practical instantiation of this idea. Consider the case that the sequence of loss functions we receive satisfy

$$\ell_t(\mathbf{x}) - \ell_t(\mathbf{u}) \leq \langle \mathbf{g}, \mathbf{x} - \mathbf{u} \rangle - \frac{\mu}{2} (\langle \mathbf{g}, \mathbf{x} - \mathbf{u} \rangle)^2, \quad \forall \mathbf{x}, \mathbf{u} \in V, \mathbf{g} \in \partial \ell_t(\mathbf{x}). \quad (7.6)$$

In words, we assume to have a class of functions that can be upper bounded by a quadratic that depends on the current subgradient. In particular, these functions possess some curvature only in the direction of (any of) the subgradients. Denoting by $A_t = \mu \mathbf{g}_t \mathbf{g}_t^\top$, we can use the above idea using

$$\psi_t(\mathbf{x}) = \frac{\lambda}{2} \|\mathbf{x}\|^2 + \frac{1}{2} \sum_{i=1}^{t-1} \|\mathbf{x}_i - \mathbf{x}\|_{A_i}^2 = \frac{\lambda}{2} \|\mathbf{x}\|^2 + \frac{\mu}{2} \sum_{i=1}^{t-1} (\langle \mathbf{g}_i, \mathbf{x} - \mathbf{x}_i \rangle)^2.$$

Hence, the update rule would be

$$\mathbf{x}_t = \operatorname{argmin}_{\mathbf{x} \in V} \sum_{i=1}^{t-1} \langle \mathbf{g}_i, \mathbf{x} \rangle + \frac{\lambda}{2} \|\mathbf{x}\|_2^2 + \frac{\mu}{2} \sum_{i=1}^{t-1} (\langle \mathbf{g}_i, \mathbf{x} - \mathbf{x}_i \rangle)^2.$$

We obtain the following algorithm, called Online Newton Step (ONS).

Denoting by $S_t = \lambda I_d + \sum_{i=1}^t A_i$ and using (7.5), we have

$$\sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \frac{\lambda}{2} \|\mathbf{u}\|_2^2 + \frac{1}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_{S_t^{-1}}^2.$$

To bound the last term, we will use the following Lemma.

Algorithm 7.5 Online Newton Step

Require: $V \subset \mathbb{R}^d$ closed non-empty convex set, $\lambda, \mu > 0$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Set $\mathbf{x}_t = \operatorname{argmin}_{\mathbf{x} \in V} \sum_{i=1}^{t-1} \langle \mathbf{g}_i, \mathbf{x} \rangle + \frac{\lambda}{2} \|\mathbf{x}\|_2^2 + \frac{\mu}{2} \sum_{i=1}^{t-1} (\langle \mathbf{g}_i, \mathbf{x} - \mathbf{x}_i \rangle)^2$
 - 3: Receive ℓ_t and pay $\ell_t(\mathbf{x}_t)$
 - 4: Set $\mathbf{g}_t \in \partial \ell(\mathbf{x}_t)$
 - 5: **end for**
-

Lemma 7.20 ([Cesa-Bianchi and Lugosi, 2006, Lemma 11.11 and Theorem 11.7]). *Let $\mathbf{z}_1, \dots, \mathbf{z}_T$ a sequence of vectors in \mathbb{R}^d and $\lambda > 0$. Define $H_t = \lambda I_d + \sum_{i=1}^t \mathbf{z}_i \mathbf{z}_i^\top$. Then, the following holds*

$$\sum_{t=1}^T \mathbf{z}_t^\top H_t^{-1} \mathbf{z}_t \leq \sum_{i=1}^d \ln \left(1 + \frac{\lambda_i}{\lambda} \right),$$

where $\lambda_1, \dots, \lambda_d$ are the eigenvalues of $H_T - \lambda I_d$.

Putting all together and assuming $\|\mathbf{g}_t\|_2 \leq L$ and (7.6) holds for the losses, then ONS satisfies the following regret

$$\sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \frac{\lambda}{2} \|\mathbf{u}\|_2^2 + \frac{1}{2\mu} \sum_{i=1}^d \ln \left(1 + \frac{\lambda_i}{\lambda} \right) \leq \frac{\lambda}{2} \|\mathbf{u}\|_2^2 + \frac{d}{2\mu} \ln \left(1 + \frac{\mu T L^2}{d\lambda} \right),$$

where in the second inequality we used the inequality of arithmetic and geometric means, $(\prod_{i=1}^d x_i)^{1/d} \leq \frac{1}{d} \sum_{i=1}^d x_i$, and the fact that $\sum_{i=1}^d \lambda_i \leq \mu T L^2$.

Hence, if the losses satisfy (7.6), we can guarantee a logarithmic regret. However, differently from the strongly convex case, here the complexity of the update is at least quadratic in the number of dimensions. Moreover, the regret also depends linearly on the number of dimensions.

Remark 7.21. *Despite the name, the ONS algorithm should not be confused with the Newton algorithm. They are similar in spirit because they both construct quadratic approximation to the function, but the Newton algorithm uses the exact Hessian while the ONS uses an approximation that works only for a restricted class of functions. In this view, the ONS algorithm is more similar to Quasi-Newton methods. However, the best lens to understand the ONS is still through generalized concept of strong convexity.*

Let's now see an example of functions that satisfy (7.6).

Example 7.22 (Exp-Concave Losses). *Defining $X \subseteq \mathbb{R}^d$ convex, we say that a function $f : X \rightarrow \mathbb{R}$ is α -exp-concave if $\exp(-\alpha f(\mathbf{x}))$ is concave.*

Choose $\beta \leq \frac{\alpha}{2}$ such that $|\beta \langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle| \leq \frac{1}{2}$ for all $\mathbf{x}, \mathbf{y} \in X$ and $\mathbf{g} \in \partial f(\mathbf{x})$. Note that we need a bounded domain for β to exist. Then, this class of functions satisfy the property (7.6). In fact, given that f is α -exp-concave then it is also 2β -exp-concave. Hence, from the definition we have

$$\exp(-2\beta f(\mathbf{y})) \leq \exp(-2\beta f(\mathbf{x})) - 2\beta \exp(-2\beta f(\mathbf{x})) \langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle,$$

that is

$$\exp(-2\beta f(\mathbf{y}) + 2\beta f(\mathbf{x})) \leq 1 - 2\beta \langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle,$$

that implies

$$f(\mathbf{x}) - f(\mathbf{y}) \leq \frac{1}{2\beta} \ln(1 + 2\beta \langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle) \leq \langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle - \frac{\beta}{2} (\langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle)^2,$$

where we used the elementary inequality $\ln(1+x) \leq x - x^2/4$, for $|x| \leq 1$.

Example 7.23. *Let $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq U\}$. The logistic loss of a linear predictor $\ell(\mathbf{x}) = \ln(1 + \exp(-\langle \mathbf{z}, \mathbf{x} \rangle))$, where $\|\mathbf{z}\|_2 \leq 1$ is $\exp(-2U)/2$ -exp-concave.*

7.7 Online Regression: Vovk-Azoury-Warmuth Forecaster

Let's now consider the specific case that $\ell_t(\mathbf{x}) = \frac{1}{2}(\langle \mathbf{z}_t, \mathbf{x} \rangle - y_t)^2$ and $V = \mathbb{R}^d$, that is the one of *unconstrained online linear regression with square loss*. These losses are not strongly convex w.r.t. \mathbf{x} , but they are exp-concave when the domain is bounded. We could use the ONS algorithm, but it would not work in the unbounded case. Another possibility would be to run FTRL, but that losses are not strongly convex and we would get only a $O(\sqrt{T})$ regret.

It turns out we can still get a logarithmic regret, if we make an additional assumption! We will assume to have access to \mathbf{z}_t before predicting \mathbf{x}_t . Note that this is a mild assumptions in most of the interesting applications. Then, the algorithm will just be *FTRL over the past losses plus the loss on the received \mathbf{z}_t hallucinating a label of 0*. We will call this algorithm Vovk-Azoury-Warmuth forecaster, from the names of the inventors. The details are in Algorithm 7.6.

Algorithm 7.6 Vovk-Azoury-Warmuth Forecaster

Require: $\lambda > 0$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Receive $\mathbf{z}_t \in \mathbb{R}^d$
 - 3: Set $\mathbf{x}_t = \operatorname{argmin}_{\mathbf{x}} \frac{\lambda}{2} \|\mathbf{x}\|_2^2 + \frac{1}{2} \sum_{i=1}^{t-1} (\langle \mathbf{z}_i, \mathbf{x} \rangle - y_i)^2 + \frac{1}{2} (\langle \mathbf{z}_t, \mathbf{x} \rangle)^2$
 - 4: Receive $y_t \in \mathbb{R}$ and pay $\ell(\mathbf{x}_t) = \frac{1}{2} (\langle \mathbf{z}_t, \mathbf{x}_t \rangle - y_t)^2$
 - 5: **end for**
-

As we did for composite losses, we look closely to the loss functions, to see if there are terms that we might move inside the regularizer. The motivation would be the same as in the composite losses case: the bound will depends only on the subgradients of the part of the losses that are outside of the regularizer.

So, observe that

$$\ell_t(\mathbf{x}) = \frac{1}{2} (\langle \mathbf{z}_t, \mathbf{x} \rangle)^2 - y_t \langle \mathbf{z}_t, \mathbf{x} \rangle + \frac{1}{2} y_t^2.$$

From the above, we see that we could think to move the terms $\frac{1}{2} (\langle \mathbf{z}_t, \mathbf{x} \rangle)^2$ in the regularizer and leaving the linear terms in the loss: $\tilde{\ell}_t(\mathbf{x}) = -y_t \langle \mathbf{z}_t, \mathbf{x} \rangle$. Hence, we will use

$$\psi_t(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \left(\sum_{i=1}^t \mathbf{z}_i \mathbf{z}_i^\top \right) \mathbf{x} + \frac{\lambda}{2} \|\mathbf{x}\|_2^2.$$

Note that the regularizer at time t contains the \mathbf{z}_t that is revealed to the algorithm before it makes its prediction. For simplicity of notation, denote by $S_t = \lambda I_d + \sum_{i=1}^t \mathbf{z}_i \mathbf{z}_i^\top$.

Using such procedure, the prediction can be written in a closed form:

$$\begin{aligned} \mathbf{x}_t &= \operatorname{argmin}_{\mathbf{x}} \frac{\lambda}{2} \|\mathbf{x}\|_2^2 + \frac{1}{2} \sum_{i=1}^{t-1} (\langle \mathbf{z}_i, \mathbf{x} \rangle - y_i)^2 + \frac{1}{2} (\langle \mathbf{z}_t, \mathbf{x} \rangle)^2 \\ &= \left(\lambda I_d + \sum_{i=1}^t \mathbf{z}_i \mathbf{z}_i^\top \right)^{-1} \sum_{i=1}^{t-1} y_i \mathbf{z}_i. \end{aligned}$$

Hence, using the regret we proved for FTRL with strongly convex regularizers and $\psi_{T+1} = \psi_T$, we get the following guarantee

$$\begin{aligned} \sum_{t=1}^T (\tilde{\ell}_t(\mathbf{x}_t) - \tilde{\ell}_t(\mathbf{u})) &= \sum_{t=1}^T (-y_t \langle \mathbf{z}_t, \mathbf{x}_t \rangle + y_t \langle \mathbf{z}_t, \mathbf{u} \rangle) \\ &\leq \psi_T(\mathbf{u}) - \min_{\mathbf{x}} \psi_T(\mathbf{x}) + \frac{1}{2} \sum_{t=1}^T y_t^2 \mathbf{z}_t^\top S_t^{-1} \mathbf{z}_t + \sum_{t=1}^{T-1} (\psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1})) \\ &= \frac{1}{2} \mathbf{u}^\top S_T \mathbf{u} + \frac{1}{2} \sum_{t=1}^T y_t^2 \mathbf{z}_t^\top S_t^{-1} \mathbf{z}_t - \frac{1}{2} \sum_{t=1}^{T-1} (\langle \mathbf{z}_{t+1}, \mathbf{x}_{t+1} \rangle)^2. \end{aligned}$$

Noting that $\mathbf{x}_1 = \mathbf{0}$ and reordering the terms, we have

$$\begin{aligned} \sum_{t=1}^T \frac{1}{2} (\langle \mathbf{z}_t, \mathbf{x}_t \rangle - y_t)^2 - \sum_{t=1}^T \frac{1}{2} (\langle \mathbf{z}_t, \mathbf{u} \rangle - y_t)^2 &= \frac{1}{2} \sum_{t=1}^T (\langle \mathbf{z}_t, \mathbf{x}_t \rangle)^2 + \sum_{t=1}^T (-y_t \langle \mathbf{z}_t, \mathbf{x}_t \rangle + y_t \langle \mathbf{z}_t, \mathbf{u} \rangle) - \frac{1}{2} \sum_{t=1}^T (\langle \mathbf{z}_t, \mathbf{u} \rangle)^2 \\ &\leq \frac{\lambda}{2} \|\mathbf{u}\|_2^2 + \frac{1}{2} \sum_{t=1}^T y_t^2 \mathbf{z}_t^\top S_t^{-1} \mathbf{z}_t. \end{aligned}$$

Remark 7.24. Note that, differently from the ONS algorithm, the regularizers here are not proximal. Yet, we get S_t^{-1} in the bound because the current sample \mathbf{z}_t is used in the regularizer. Without the knowledge of \mathbf{z}_t the last term would be $\frac{1}{2} \sum_{t=1}^T y_t^2 \mathbf{z}_t^\top S_{t-1}^{-1} \mathbf{z}_t$ that cannot be controlled without additional assumptions on \mathbf{z}_t .

So, using again Lemma 7.20 and assuming $|y_t| \leq Y$, $t = 1, \dots, T$, we have

$$\sum_{t=1}^T \frac{1}{2} (\langle \mathbf{z}_t, \mathbf{x}_t \rangle - y_t)^2 - \sum_{t=1}^T \frac{1}{2} \sum_{i=1}^T (\langle \mathbf{z}_t, \mathbf{u} \rangle - y_t)^2 \leq \frac{\lambda}{2} \|\mathbf{u}\|_2^2 + \frac{Y^2}{2} \sum_{i=1}^d \ln \left(1 + \frac{\lambda_i}{\lambda} \right),$$

where $\lambda_1, \dots, \lambda_d$ are the eigenvalues of $\sum_{i=1}^T \mathbf{z}_i \mathbf{z}_i^\top$.

If we assume that $\|\mathbf{z}_t\|_2 \leq R$, we can reason as we did for the similar term in ONS, to have

$$\sum_{i=1}^d \ln \left(1 + \frac{\lambda_i}{\lambda} \right) \leq d \ln \left(1 + \frac{R^2 T}{\lambda d} \right).$$

Putting all together, we have the following theorem.

Theorem 7.25. Assume $\|\mathbf{z}_t\|_2 \leq R$ and $|y_t| \leq Y$ for $t = 1, \dots, T$. Then, using the prediction strategy in Algorithm 7.6, we have

$$\sum_{t=1}^T \frac{1}{2} (\langle \mathbf{z}_t, \mathbf{x}_t \rangle - y_t)^2 - \sum_{t=1}^T \sum_{i=1}^T \frac{1}{2} (\langle \mathbf{z}_t, \mathbf{u} \rangle - y_t)^2 \leq \frac{\lambda}{2} \|\mathbf{u}\|_2^2 + \frac{dY^2}{2} \ln \left(1 + \frac{R^2 T}{\lambda d} \right).$$

Remark 7.26. It is possible to show that the regret of the Vovk-Azoury-Warmuth forecaster is optimal up to multiplicative factors [Cesa-Bianchi and Lugosi, 2006, Theorem 11.9].

7.8 Optimistic FTRL

Throughout this book, we considered the adversarial model as our model of the environment. This allowed us to design algorithm that work in this setting, as well as in other more benign settings. However, the world is never completely adversarial. So, we might be tempted to model the environment in some way, but that would leave our algorithm vulnerable to attacks. An alternative, is to consider the data as generated by some *predictable process plus adversarial noise*. In this view, it might be beneficial to try to model the predictable part, without compromising the robustness to the adversarial noise.

In this section, we will explore this possibility through a particular version of Follow-The-Regularized-Leader (FTRL), where we *predict* the next loss. In very intuitive terms, if our predicted loss is correct, we can expect the regret to decrease. However, if our prediction is wrong we still want to recover the worst case guarantee. Such algorithm is called **Optimistic FTRL**.

The core idea of Optimistic FTRL is to predict the next loss and use it in the update rule, as summarized in Algorithm 7.7. Note that for the sake of the analysis, it does not matter how the prediction is generated. It can be even generated by another online learning procedure!

Let's see why this is a good idea. Remember that FTRL simply predicts with the minimizer of the previous losses plus a time-varying regularizer. Let's assume for a moment that instead we have the gift of predicting the future, so we

Algorithm 7.7 Optimistic Follow-the-Regularized-Leader

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

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

do know the next loss ahead of time. Then, we could predict with its minimizer and suffer a negative regret. However, probably our foresight abilities are not so powerful, so our prediction of the next loss might be inaccurate. In this case, a better idea might be just to add our predicted loss to the previous ones and minimize the regularized sum. We would expect the regret guarantee to improve if our prediction of the future loss is precise. At the same time, if the prediction is wrong, we expect its influence to be limited, given that we use it together with all the past losses.

All these intuitions can be formalized in the following Theorem.

Theorem 7.27. *With the notation in Algorithm 7.7, let $V \subseteq \mathbb{R}^d$ be convex, closed, and non-empty. Denote by $F_t(\mathbf{x}) = \psi_t(\mathbf{x}) + \sum_{i=1}^{t-1} \ell_i(\mathbf{x})$. Assume for $t = 1, \dots, T$ that F_t is proper and λ_t -strongly convex w.r.t. $\|\cdot\|$, ℓ_t and $\tilde{\ell}_t$ proper and convex, and $\operatorname{int} \operatorname{dom} F_t \cap V \neq \{\}$. Also, assume that $\partial \ell_t(\mathbf{x}_t)$ and $\partial \tilde{\ell}_t(\mathbf{x}_t)$ are non-empty. Then, there exists $\tilde{\mathbf{g}}_t \in \partial \tilde{\ell}_t(\mathbf{x}_t)$ for $t = 1, \dots, T$, such that we have*

$$\begin{aligned} \sum_{t=1}^T \ell(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) &\leq \psi_{T+1}(\mathbf{u}) - \psi_1(\mathbf{x}_1) + \sum_{t=1}^T \left[\langle \mathbf{g}_t - \tilde{\mathbf{g}}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 + \psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1}) \right] \\ &\leq \psi_{T+1}(\mathbf{u}) - \psi_1(\mathbf{x}_1) + \sum_{t=1}^T \left[\frac{\|\mathbf{g}_t - \tilde{\mathbf{g}}_t\|_*^2}{2\lambda_t} + \psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1}) \right], \end{aligned}$$

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

Proof. We can interpret the Optimistic-FTRL as FTRL with a regularizer $\tilde{\psi}_t(\mathbf{x}) = \psi_t(\mathbf{x}) + \ell_t(\mathbf{x})$. Also, note that $\tilde{\ell}_{T+1}(\mathbf{x})$ has no influence on the algorithm, so we can set it to the null function.

Hence, from the regret equality for FTRL, we immediately get

$$\begin{aligned} \sum_{t=1}^T \ell(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) &\leq \tilde{\ell}_{T+1}(\mathbf{u}) + \psi_{T+1}(\mathbf{u}) - \min_{\mathbf{x} \in V} (\tilde{\ell}_1(\mathbf{x}) + \psi_1(\mathbf{x})) + \sum_{t=1}^T [F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t) + \tilde{\ell}_t(\mathbf{x}_t) - \tilde{\ell}_{t+1}(\mathbf{x}_{t+1})] \\ &= \psi_{T+1}(\mathbf{u}) - \psi_1(\mathbf{x}_1) + \sum_{t=1}^T [F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t)]. \end{aligned}$$

Now focus on the terms $F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t)$. Observe that $F_t(\mathbf{x}) + \ell_t(\mathbf{x}) + i_V(\mathbf{x})$ is λ_t -strongly convex w.r.t. $\|\cdot\|$, hence we have

$$\begin{aligned} F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t) &= (F_t(\mathbf{x}_t) + \ell_t(\mathbf{x}_t)) - (F_t(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_{t+1})) + \psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1}) \\ &\leq \langle \mathbf{g}'_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 + \psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1}), \end{aligned}$$

where $\mathbf{g}'_t \in \partial(F_t(\mathbf{x}_t) + \ell_t(\mathbf{x}_t) + i_V(\mathbf{x}_t))$. Observing that $\mathbf{x}_t = \operatorname{argmin}_{\mathbf{x} \in V} F_t(\mathbf{x}) + \tilde{\ell}_t(\mathbf{x})$, we have $\mathbf{0} \in \partial(F_t(\mathbf{x}_t) + \tilde{\ell}_t(\mathbf{x}_t) + i_V(\mathbf{x}_t))$. Hence, given that our assumptions guarantee that the subdifferential of the sum is equal to the sum

of the subdifferentials, there exists $\tilde{\mathbf{g}}_t \in \partial \tilde{\ell}_t(\mathbf{x}_t)$ such that $\mathbf{g}'_t = \mathbf{g}_t - \tilde{\mathbf{g}}_t$. So, we have

$$F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t) \leq \langle \mathbf{g}_t - \tilde{\mathbf{g}}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 + \psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1}).$$

By the definition of dual norms, we also have that

$$\langle \mathbf{g}_t - \tilde{\mathbf{g}}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \leq \|\mathbf{g}_t - \tilde{\mathbf{g}}_t\|_* \|\mathbf{x}_t - \mathbf{x}_{t+1}\| - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \leq \frac{\lambda_t}{2} \|\mathbf{g}_t - \tilde{\mathbf{g}}_t\|_*^2. \quad \square$$

Let's take a look at the second bound in the theorem. Compared to the similar bound for FTRL, we now have the terms $\|\mathbf{g}_t - \tilde{\mathbf{g}}_t\|_*^2$ instead of the ones $\|\mathbf{g}_t\|_*^2$. So, if the prediction of the next loss is good, that term can become smaller and possibly even zero! On the other hand, if the predictions are bad, for Lipschitz losses we only lose a constant factor. Overall, in the best case we can gain a lot, in the worst case we do not lose that much.

Despite the simplicity of the algorithm and its analysis, there are many applications of this principle. We will only describe a couple of them. Recently, this idea was used even to recover the Nesterov's acceleration algorithm and to prove faster convergence in repeated games.

7.8.1 Regret that Depends on the Variance of the Subgradients

Consider of running Optimistic-FTRL on the linearized losses $\ell_t(\mathbf{x}) = \langle \mathbf{g}_t, \mathbf{x} \rangle$. We can gain something out of the Optimistic-FTRL compared to plain FTRL if we are able to predict the next \mathbf{g}_t . A simple possibility is to predict the average of the past values, $\tilde{\mathbf{g}}_t = \frac{1}{t-1} \sum_{i=1}^{t-1} \mathbf{g}_i$. Indeed, from the first chapter, we know that such strategy is itself an online learning procedure! In particular, it corresponds to a Follow-The-Leader algorithm on the losses $\ell_t(\mathbf{x}) = \|\mathbf{x} - \mathbf{g}_t\|_2^2$. Hence, from the strong convexity of this losses, we know that

$$\sum_{t=1}^T \|\bar{\mathbf{x}}_t - \mathbf{g}_t\|_2^2 - \sum_{t=1}^T \|\mathbf{u} - \mathbf{g}_t\|_2^2 \leq 4(1 + \ln T), \quad \forall \mathbf{u} \in R^d.$$

This implies

$$\sum_{t=1}^T \|\bar{\mathbf{x}}_t - \mathbf{g}_t\|_2^2 \leq 4(1 + \ln T) + \min_{\mathbf{u}} \sum_{t=1}^T \|\mathbf{u} - \mathbf{g}_t\|_2^2.$$

It is immediate to see that the minimizer is $\mathbf{u} = \frac{1}{T} \sum_{t=1}^T \mathbf{g}_t$, that results in T times the empirical variance of the subgradients. Plugging it in the Optimistic-FTRL regret, with $\psi_t = \psi$, we have

$$\sum_{t=1}^T \ell(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \psi(\mathbf{u}) - \psi(\mathbf{x}_1) + 4 + 4 \ln T + \min_{\mathbf{g}} \sum_{t=1}^T \frac{\|\mathbf{g}_t - \mathbf{g}\|_*^2}{2\lambda}.$$

Remark 7.28. *Instead of using the mean of the past subgradients, we could use any other strategy or even a mix of different strategies. For example, assuming the subgradients bounded, we could use an algorithm to solve the Learning with Expert problem, where each expert is a strategy. Then, we would obtain a bound that depends on the predictions of the best strategy, plus the regret of the expert algorithm.*

7.8.2 Online Convex Optimization with Gradual Variations

In this section, we consider the case that the losses we receive have small variations over time. We will show that in this case it is possible to get constant regret in the case that the losses are equal.

In this case, the simple strategy we can use to predict the next subgradient is to use the previous one, that is $\tilde{\ell}_t(\mathbf{x}) = \langle \mathbf{g}_{t-1}, \mathbf{x} \rangle$ for $t \geq 2$ and $\tilde{\ell}_1(\mathbf{x}) = 0$.

Corollary 7.29. Under the assumptions of Theorem 7.27, define $\tilde{\ell}_t(\mathbf{x}) = \langle \mathbf{g}_{t-1}, \mathbf{x} \rangle$ for $t \geq 2$ and $\tilde{\ell}_1(\mathbf{x}) = 0$. Set $\psi_t(\mathbf{x}) = \lambda_t \psi(\mathbf{x})$ where ψ is 1-strongly convex w.r.t. $\|\cdot\|$ and λ_t satisfies $\lambda_t \lambda_{t-1} \geq 8M^2$ for $t = 2, \dots, T$, where M is the smoothness constant of the losses ℓ_t . Then, $\forall \mathbf{u} \in V$, we have

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq \lambda_t \psi(\mathbf{u}) - \lambda_1 \psi(\mathbf{x}_1) + \frac{1}{\lambda_1} \|\nabla \ell_1(\mathbf{x}_1)\|_*^2 + \sum_{t=2}^T \frac{2}{\lambda_t} \|\nabla \ell_t(\mathbf{x}_{t-1}) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2.$$

Moreover, assuming $\|\nabla \ell_t(\mathbf{x})\|_* \leq L$ for all $\mathbf{x} \in V$, setting $\lambda_t = \sqrt{\max(8M^2, 4L^2) + \sum_{i=2}^{t-1} \|\nabla \ell_i(\mathbf{x}_{i-1}) - \nabla \ell_{i-1}(\mathbf{x}_{i-1})\|_*^2}$, we have

$$\text{Regret}_T(\mathbf{u}) \leq \left(\psi(\mathbf{u}) - \min_{\mathbf{x}} \psi(\mathbf{x}) + 4 \right) \sqrt{\max(8M^2, 4L^2) + \sum_{t=2}^T \|\nabla \ell_t(\mathbf{x}_{t-1}) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2} + \frac{1}{4}.$$

Proof. From the Optimistic-FTRL bound with a fixed regularizer, we immediately get

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq \lambda_T \psi(\mathbf{u}) - \lambda_1 \psi(\mathbf{x}_1) + \sum_{t=1}^T \left[\langle \mathbf{g}_t - \tilde{\mathbf{g}}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \right].$$

Now, consider the case that the losses ℓ_t are M -smooth. So, for any $\alpha_t > 0$, we have

$$\begin{aligned} \langle \mathbf{g}_t - \tilde{\mathbf{g}}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 &= \langle \nabla \ell_t(\mathbf{x}_t) - \nabla \ell_{t-1}(\mathbf{x}_{t-1}), \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \\ &\leq \frac{\alpha_t}{2} \|\nabla \ell_t(\mathbf{x}_t) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2 + \frac{1}{2\alpha_t} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2. \end{aligned}$$

Focusing on the first term, for $t = 2, \dots, T$, we have

$$\begin{aligned} \frac{\alpha_t}{2} \|\nabla \ell_t(\mathbf{x}_t) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2 &= \frac{\alpha_t}{2} \|\nabla \ell_t(\mathbf{x}_t) - \nabla \ell_{t-1}(\mathbf{x}_{t-1}) - \nabla \ell_t(\mathbf{x}_{t-1}) + \nabla \ell_t(\mathbf{x}_{t-1})\|_*^2 \\ &\leq \alpha_t \|\nabla \ell_t(\mathbf{x}_t) - \nabla \ell_t(\mathbf{x}_{t-1})\|_*^2 + \alpha_t \|\nabla \ell_t(\mathbf{x}_{t-1}) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2 \\ &\leq \alpha_t M^2 \|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2 + \alpha_t \|\nabla \ell_t(\mathbf{x}_{t-1}) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2. \end{aligned}$$

Choose $\alpha_t = \frac{2}{\lambda_t}$. We have for $t = 2, \dots, T$

$$\begin{aligned} \langle \mathbf{g}_t - \tilde{\mathbf{g}}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 &\leq \frac{2M^2}{\lambda_t} \|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2 + \frac{2}{\lambda_t} \|\nabla \ell_t(\mathbf{x}_{t-1}) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2 - \frac{\lambda_t}{4} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2. \end{aligned}$$

For $t = 1$, we have

$$\langle \mathbf{g}_t - \tilde{\mathbf{g}}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \leq \frac{1}{\lambda_t} \|\nabla \ell_t(\mathbf{x}_t) - \tilde{\mathbf{g}}_t\|_*^2 - \frac{\lambda_t}{4} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2.$$

Now observe the assumption λ_t implies $\frac{2M^2}{\lambda_t} \leq \frac{\lambda_{t-1}}{4}$ for $t = 2, \dots, T$. So, summing for $t = 1, \dots, T$, we have

$$\sum_{t=1}^T \left(\langle \mathbf{g}_t - \tilde{\mathbf{g}}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle - \frac{\lambda_t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \right) \leq \frac{1}{\lambda_1} \|\nabla \ell_1(\mathbf{x}_1)\|_*^2 + \sum_{t=2}^T \frac{2}{\lambda_t} \|\nabla \ell_t(\mathbf{x}_{t-1}) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2.$$

Putting all together, we have the first stated bound.

The second one is obtained observing that

$$\begin{aligned}
\sum_{t=2}^T \frac{\|\nabla \ell_t(\mathbf{x}_{t-1}) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2}{\lambda_t} &= \sum_{t=2}^T \frac{\|\nabla \ell_t(\mathbf{x}_{t-1}) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2}{\sqrt{\max(8M^2, 4L^2) + \sum_{i=2}^{t-1} \|\nabla \ell_i(\mathbf{x}_{i-1}) - \nabla \ell_{i-1}(\mathbf{x}_{i-1})\|_*^2}} \\
&\leq \sum_{t=2}^T \frac{\|\nabla \ell_t(\mathbf{x}_{t-1}) - \nabla \ell_{t-1}(\mathbf{x}_{t-1})\|_*^2}{\sqrt{\sum_{i=2}^t \|\nabla \ell_i(\mathbf{x}_{i-1}) - \nabla \ell_{i-1}(\mathbf{x}_{i-1})\|_*^2}} \\
&\leq 2 \sqrt{\sum_{i=2}^T \|\nabla \ell_i(\mathbf{x}_{i-1}) - \nabla \ell_{i-1}(\mathbf{x}_{i-1})\|_*^2}. \quad \square
\end{aligned}$$

Note that if the losses are all the same, the regret becomes a constant! This is not surprising, because the prediction of the next loss is a linear approximation of the previous loss. Indeed, looking back at the proof, the key idea is to use the smoothness to argue that, if even the past subgradient was taken in a different point than the current one, it is still a good prediction of the current subgradient.

Remark 7.30. *Note that the assumption of smoothness is necessary. Indeed, passing always the same function and using online-to-batch conversion, would result in a convergence rate of $O(1/T)$ for a Lipschitz function, that is impossible.*

7.9 History Bits

Follow the Regularized Leader was introduced in Abernethy et al. [2008] where at each step the prediction is computed as the minimizer of a regularization term plus the sum of losses on all past rounds. However, the key ideas of FTRL, and in particular its analysis through the dual, were planted by Shai Shalev-Shwartz and Yoram Singer way before [Shalev-Shwartz and Singer, 2006][Shalev-Shwartz and Singer, 2007b]. Later, the PhD thesis of Shai Shalev-Shwartz [Shalev-Shwartz, 2007] contained the most precise dual analysis of FTRL, but he called it “online mirror descent” because the name FTRL was only invented later! Later, I contributed to the confusion naming a general analysis of FTRL with time-varying regularizer and linear losses “generalized online mirror descent” [Orabona et al., 2015]. So, now I am trying to set the record straight :)

Later to all this, the optimization community starts using FTRL with linear losses as well, calling it Dual Averaging [Nesterov, 2009]. Note that this paper by Nesterov is actually from 2005¹, and in the paper he claims that these ideas are from 2001-2002, but he decided not to publish them for a while because he was convinced that “the importance of black-box approach in Convex Optimization will be irreversibly vanishing, and, finally, this approach will be completely replaced by other ones based on a clever use of problems structure (interior-point methods, smoothing, etc.).” Hence, Nesterov is for sure the the first inventor of offline FTRL with linear losses.

The same ideas were then translated to online learning and stochastic optimization in Xiao [2010], essentially rediscovering the framework of Shalev-Shwartz and rebranding it Regularized Dual Averaging (RDA). Finally, McMahan [2017] gives the elegant equality result that I presented here (with minor improvements) that holds for general loss functions and regularizers. A similar equality, setting $\mathbf{u} = \mathbf{x}_{T+1}$ and specialized only to linear losses, was also proven in Abernethy et al. [2014], essentially matching the inequality in Orabona et al. [2015]. Note that the dual interpretation of FTRL comes out naturally for linear losses, but Lemma 7.1 underlines the fact that the algorithm is actually more general.

Another source of confusion stems from the fact that some people differentiate among a “lazy” and “greedy” version of OMD. In reality, as proved in McMahan [2017], the lazy algorithm is just FTRL with linearized losses and the greedy one is just OMD. The notation “lazy online mirror descent” was introduced in Zinkevich [2004], where he basically introduced for the first time FTRL with linearized losses in the special case of $\psi_t(\mathbf{x}) = \frac{\eta}{2} \|\mathbf{x}\|_2^2$.

The use of FTRL with the regularizer in (7.2) was proposed in Orabona and Pál [2015], I presented a simpler version of their proof that does not require Fenchel conjugates. The AdaHedge algorithm was introduced in

¹https://papers.ssrn.com/sol3/papers.cfm?abstract_id=912637

van Erven et al. [2011] and refined in de Rooij et al. [2014]. The analysis reported here is from Orabona and Pál [2015], that generalized AdaHedge to arbitrary regularizers in AdaFTRL. Additional properties of AdaHedge for the stochastic case were proven in de Rooij et al. [2014].

The first analysis of FTRL with composite losses is in Xiao [2010]. The analysis presented here using the negative terms $\psi_t(\mathbf{x}_{t+1}) - \psi_{t+1}(\mathbf{x}_{t+1})$ to easily prove regret bounds for FTRL for composite losses is from Orabona et al. [2015].

The first proof of FTRL for strongly convex losses was in Shalev-Shwartz and Singer [2007a] (even if they do not call it FTRL).

There is an interesting bit about FTRL-Proximal [McMahan, 2011]: FTRL-Proximal is an instantiation of FTRL that uses a particular proximal regularizer. It became very famous in internet companies when Google disclosed in a very influential paper that they were using FTRL-Proximal to train the classifier for click prediction [McMahan et al., 2013]. This generated even more confusion because many people started conflating the term FTRL-Proximal (a specific algorithm) with FTRL (an entire family of algorithms). Unfortunately, this confusion is still going on in these days.

The Online Newton Step algorithm was introduced in Hazan et al. [2006] and it is described to the particular case that the loss functions are exp-concave. Exp-concave functions were studied in Kivinen and Warmuth [1999]. Here, I described a slight generalization for any sequence of functions that satisfy (7.6), that in my view it better shows the parallel between FTRL over strongly convex functions and ONS. Note that Hazan et al. [2006] also describes a variant of ONS based on OMD, but I find its analysis less interesting from a didactic point of view. The proof presented here through the properties of proximal regularizers might be new, I am not sure.

The Vovk-Azoury-Warmuth forecaster was introduced independently by Azoury and Warmuth [2001] and Vovk [2001]. The proof presented here is from Orabona et al. [2015].

The Optimistic Online Mirror Descent algorithm was proposed by Chiang et al. [2012] and extended in Rakhlin and Sridharan [2013] to use arbitrary “hallucinated” losses. The Optimistic FTRL version was proposed in Rakhlin and Sridharan [2013] and rediscovered in Steinhardt and Liang [2014], even if it was called Online Mirror Descent for the misnaming problem we already explained. The proof of Theorem 7.27 I present here is new.

Corollary 7.29 was proved by Chiang et al. [2012] for Optimistic OMD and presented in a similar form in Joulani et al. [2017] for Optimistic FTRL, but for bounded domains.

7.10 Exercises

Problem 7.1. *Prove that the update of FTRL with linearized loss in Example 7.9 is correct.*

Problem 7.2. *Consider Follow-the-Regularized-Leader (FTRL) with losses $\ell_t(\mathbf{x})$ and regularizers $\psi_t(\mathbf{x})$. Assume that $\psi_{t+1} \geq \psi_t$ for $t = 1, \dots, T-1$. Denote by $F_t(\mathbf{x}) := \psi_t(\mathbf{x}) + \sum_{i=1}^{t-1} \ell_i(\mathbf{x})$ and show that*

$$F_t(\mathbf{x}_t) - F_{t+1}(\mathbf{x}_{t+1}) + \ell_t(\mathbf{x}_t) \leq \ell_t(\mathbf{x}_t) - \ell_t(\mathbf{x}_{t+1}).$$

Problem 7.3. *Find a way to have L^* bounds for smooth losses with linearized FTRL: Do you need an additional assumption compared to what we did for OSD?*

Problem 7.4. *Prove that the update in (7.4) is equivalent to the one of OSD with $\mathbf{x}_1 = 0$ and learning rate $\eta_t = \frac{1}{\sum_{i=1}^t \mu_i}$.*

Problem 7.5. *Prove the statement in Example 7.23.*

Problem 7.6. *Prove that the losses $\ell_t(\mathbf{x}) = \frac{1}{2}(\langle \mathbf{z}_t, \mathbf{x} \rangle - y_t)^2$, where $\|\mathbf{z}_t\|_2 \leq 1$, $|y_t| \leq 1$, and $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 \leq 1\}$, are exp-concave and find the exp-concavity constant.*

Chapter 8

Online Linear Classification

In this chapter, we will consider the problem of *online linear classification*. We consider the following setting:

- At each time step we receive a sample \mathbf{z}_t
- We output a prediction of the binary label $y_t \in \{-1, 1\}$ of \mathbf{z}_t
- We receive the true label and we see if we did a mistake or not
- We update our online classifier

The aim of the online algorithm is to minimize the number of mistakes it does compared to some best fixed classifier.

We will focus on linear classifiers, that predicts with the sign of the inner product between a vector \mathbf{x}_t and the input features \mathbf{z}_t . Hence, $\tilde{y}_t = \text{sign}(\langle \mathbf{z}_t, \mathbf{x}_t \rangle)$. This problem can be written again as a regret minimization problem:

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}),$$

where $\ell_t(\mathbf{x}) = \mathbf{1}[\text{sign}(\langle \mathbf{z}_t, \mathbf{x} \rangle) \neq y_t]$. It should be clear that these losses are non-convex. Hence, we need an alternative way to deal with them. In the following, we will see two possible approaches to this problem.

8.1 Online Randomized Classifier

As we did for the Learning with Expert Advice framework, we might think to convexify the losses using randomization. Hence, on each round we can predict a number in $q_t \in [-1, 1]$ and output the label $+1$ according with probability $\frac{1+q_t}{2}$ and the label -1 with probability $\frac{1-q_t}{2}$. So, define the random variable

$$\tilde{y}(p) = \begin{cases} +1, & \text{with probability } p, \\ -1, & \text{with probability } 1 - p. \end{cases}$$

Now observe that $\mathbb{E}_{\tilde{y}_t} [\tilde{y}(\frac{1+q_t}{2}) \neq y_t] = \frac{1}{2}|q_t - y_t|$. If we consider linear predictors, we can think to have $q_t = \langle \mathbf{z}_t, \mathbf{x}_t \rangle$ and similarly for the competitor $q'_t = \langle \mathbf{z}_t, \mathbf{u} \rangle$. Constraining both the algorithm and the competitor to the space of vectors where $|\langle \mathbf{z}_t, \mathbf{x} \rangle| \leq 1$ for $t = 1, \dots, T$, we can write

$$\mathbb{E} \left[\sum_{t=1}^T \mathbf{1}[\tilde{y}(\langle \mathbf{z}_t, \mathbf{x}_t \rangle) \neq y_t] - \sum_{t=1}^T \mathbf{1}[\tilde{y}(\langle \mathbf{z}_t, \mathbf{u} \rangle) \neq y_t] \right] = \mathbb{E} \left[\sum_{t=1}^T \frac{1}{2} |\langle \mathbf{z}_t, \mathbf{x}_t \rangle - y_t| - \sum_{t=1}^T \frac{1}{2} |\langle \mathbf{z}_t, \mathbf{u} \rangle - y_t| \right].$$

Hence, the surrogate convex loss becomes $\tilde{\ell}_t(\mathbf{x}) = \frac{1}{2}|\langle \mathbf{z}_t, \mathbf{x} \rangle - y_t|$ and the feasible set is any convex set where we have the property $|\langle \mathbf{z}_t, \mathbf{x} \rangle| \leq 1$ for $t = 1, \dots, T$.

Given that this problem is convex, assuming \mathbf{z}_t to be bounded w.r.t. some norm, we can use almost any of the algorithms we have seen till now, from Online Mirror Descent to Follow-The-Regularized-Leader. All of them would result in $O(\sqrt{T})$ regret upper bounds, assuming that \mathbf{z}_t are bounded in some norm. The only caveat is to restrict $\langle \mathbf{z}_t, \mathbf{x}_t \rangle$ in $[-1, 1]$. One way to do it might be to consider assuming $\|\mathbf{z}_t\|_* \leq R$ and choose the feasible set $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq \frac{1}{R}\}$.

Putting all together, for example, we can have the following strategy using FTRL with regularizers $\psi_t(\mathbf{x}) = \frac{\sqrt{2t}}{4} \|\mathbf{x}\|_2^2$.

Algorithm 8.1 Randomized Online Linear Classifier through FTRL

Require: $R > 0$ such that $\|\mathbf{z}_t\|_2 \leq R$ for $t = 1, \dots, T$

- 1: Set $\boldsymbol{\theta}_1 = \mathbf{0} \in \mathbb{R}^d$
 - 2: **for** $t = 1$ **to** T **do**
 - 3: $\mathbf{x}_t = \eta_t \boldsymbol{\theta}_t \min(\frac{1}{R\eta_t \|\boldsymbol{\theta}_t\|_2}, 1)$
 - 4: Receive $\mathbf{z}_t \in \mathbb{R}^d$
 - 5: Predict $\tilde{y}_t = \begin{cases} 1, & \text{with probability } \frac{\langle \mathbf{z}_t, \mathbf{x}_t \rangle + 1}{2} \\ -1, & \text{otherwise} \end{cases}$
 - 6: Receive y_t and pay $\mathbf{1}[y_t \neq \tilde{y}_t]$
 - 7: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{1}{2} \text{sign}(\langle \mathbf{z}_t, \mathbf{x}_t \rangle - y_t) \mathbf{z}_t$ where $\text{sign}(0) := 0$
 - 8: **end for**
-

Theorem 8.1. Let $(\mathbf{z}_t, y_t)_{t=1}^T$ an arbitrary sequence of samples/labels couples where $(\mathbf{z}_t, y_t) \in X \times \{-1, 1\}$ and $X \subset \mathbb{R}^d$. Assume $\|\mathbf{z}_t\|_2 \leq R$, for $t = 1, \dots, T$. Then, running the Randomized Online Linear Classifier algorithm with $\eta_t = \frac{\sqrt{2}}{\sqrt{t}}$ where $\alpha > 0$, for any $\mathbf{u} \in \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 \leq \frac{1}{R}\}$ we have the following guarantee

$$\mathbb{E} \left[\sum_{t=1}^T \mathbf{1}[\tilde{y}(\langle \mathbf{z}_t, \mathbf{x}_t \rangle) \neq y_t] \right] - \mathbb{E} \left[\sum_{t=1}^T \mathbf{1}[\tilde{y}(\langle \mathbf{z}_t, \mathbf{u} \rangle) \neq y_t] \right] \leq \sqrt{2T}.$$

Proof. The proof is straightforward from the FTRL regret bound with the chosen increasing regularizer. □

8.2 The Perceptron Algorithm

The above strategy has the shortcoming of restricting the feasible vectors in a possibly very small set. In turn, this could make the performance of the competitor low. In turn, the performance of the online algorithm is only close to the one of the competitor.

Another way to deal with the non-convexity is to compare the number of mistakes that the algorithm does with a convex cumulative loss of the competitor. That is, we can try to prove a weaker regret guarantee:

$$\sum_{t=1}^T \mathbf{1}[y_t \neq \tilde{y}_t] - \sum_{t=1}^T \ell(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t) = O(\sqrt{T}). \quad (8.1)$$

In particular, the convex loss we consider is *powers* of the **Hinge Loss**: $\ell_q(\tilde{y}, y) = \max(1 - \tilde{y}y, 0)^q$. The hinge loss is a convex upper bound to the 0/1 loss and it achieves the value of zero when the sign of the prediction is correct *and* the magnitude of the inner product is big enough. Moreover, taking powers of it, we get a family of functions that trade-offs the loss for the wrongly classified samples with the one for the correctly classified samples but with a value of $|\langle \mathbf{z}_t, \mathbf{x} \rangle| \leq 1$, see Figure 8.1.

The oldest algorithm we have to minimize the modified regret in (8.1) is the **Perceptron** algorithm, in Algorithm 8.2.

The Perceptron algorithm updates the current prediction \mathbf{x}_t moving in the direction of the current sample multiplied by its label. Let's see why this is a good idea. Assume that $y_t = 1$ and the algorithm made a mistake. Then, the updated

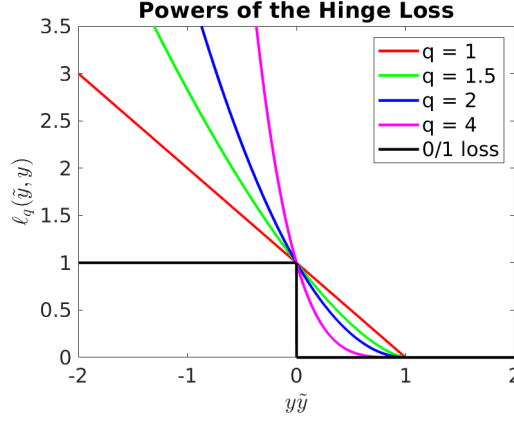


Figure 8.1: Powers of the Hinge Loss.

Algorithm 8.2 Perceptron Algorithm

- 1: Set $\mathbf{x}_1 = \mathbf{0} \in \mathbb{R}^d$
 - 2: **for** $t = 1$ **to** T **do**
 - 3: Receive $\mathbf{z}_t \in \mathbb{R}^d$
 - 4: Predict $\tilde{y}_t = \text{sign}(\langle \mathbf{z}_t, \mathbf{x}_t \rangle)$
 - 5: Receive y_t and pay $\mathbf{1}[y_t \neq \tilde{y}_t]$
 - 6: $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{1}[y_t \neq \tilde{y}_t]y_t \mathbf{z}_t$
 - 7: **end for**
-

prediction \mathbf{x}_{t+1} would predict a more positive number on the same sample \mathbf{z}_t . In fact, we have

$$\langle \mathbf{z}_t, \mathbf{x}_{t+1} \rangle = \langle \mathbf{z}_t, \mathbf{x}_t + y_t \mathbf{z}_t \rangle = \langle \mathbf{z}_t, \mathbf{x}_t \rangle + \|\mathbf{z}_t\|_2^2.$$

In the same way, if $y_t = -1$ and the algorithm made a mistake, the update would result in a more negative prediction on the same sample.

For the Perceptron algorithm we can prove the following guarantee.

Theorem 8.2. *Let $(\mathbf{z}_t, y_t)_{t=1}^T$ an arbitrary sequence of samples/labels couples where $(\mathbf{z}_t, y_t) \in X \times \{-1, 1\}$ and $X \subset \mathbb{R}^d$. Assume $\|\mathbf{z}_t\|_2 \leq R$, for $t = 1, \dots, T$. Then, running the Perceptron algorithm we have the following guarantee*

$$\sum_{t=1}^T \mathbf{1}[y_t \neq \tilde{y}_t] - \sum_{t=1}^T \ell_q(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t) \leq \frac{q^2 R^2 \|\mathbf{u}\|_2^2}{2} + q \|\mathbf{u}\| \sqrt{\frac{q^2 R^2 \|\mathbf{u}\|_2^2}{4} + \sum_{t=1}^T \ell_q(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t)}, \quad \forall \mathbf{u} \in \mathbb{R}^d, q \geq 1.$$

Before proving the theorem, let's take a look to its meaning. If there exists a $\mathbf{u} \in \mathbb{R}^d$ such that $\sum_{t=1}^T \ell_q(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t) = 0$, then the Perceptron algorithm makes a *finite* number of mistakes upper bounded by $R^2 \|\mathbf{u}\|_2^2$. In case that are many \mathbf{u} that achieves $\sum_{t=1}^T \ell_q(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t) = 0$ we have that the finite number of mistakes is bounded the norm of the smallest \mathbf{u} among them. What is the meaning of this quantity?

Remember that a hyperplane represented by its normal vector \mathbf{u} divides the space in two half spaces: one with the points \mathbf{z} that give a positive value for the inner product $\langle \mathbf{z}, \mathbf{u} \rangle$ and other one where the same inner product is negative. Now, we have that the distance of a sample \mathbf{z}_t from the hyperplane whose normal is \mathbf{u} is

$$\frac{|\langle \mathbf{z}_t, \mathbf{u} \rangle|}{\|\mathbf{u}\|_2}.$$

Also, given that we are considering a \mathbf{u} that gives cumulative hinge loss zero, we have that that quantity is at least $\frac{1}{\|\mathbf{u}\|_2}$. So, the norm of the minimal \mathbf{u} that has cumulative hinge loss equal to zero is inversely proportional to the

minimum distance between the points and the separating hyperplane. This distance is called the **margin** of the samples $(\mathbf{z}_t, y_t)_{t=1}^T$. So, if the margin is small, the Perceptron algorithm can do more mistakes than when the margin is big.

If the problem cannot be linearly separable, the Perceptron satisfies a regret of $O(\sqrt{L^*})$, where L^* is the loss of the competitor. Moreover, we measure the competitor with a *family of loss functions* and compete with the best \mathbf{u} measured with the best loss. This adaptivity is achieved through two basic ingredients:

- *The Perceptron is independent of scaling of the update by a hypothetical learning rate η* , in the sense that the mistakes it does are independent of the scaling. That is, we could update with $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \mathbf{1}[y_t \neq \tilde{y}_t] y_t \mathbf{z}_t$ and have the same mistakes and updates because they only depend on the sign of \tilde{y}_t . Hence, we can think as it is always using the best possible learning rate η .
- The weakened definition of regret allows to consider a family of loss functions, because *the Perceptron is not using any of them in the update*.

Let's now prove the regret guarantee. For the proof, we will need the two following technical lemmas.

Lemma 8.3. [Cucker and Zhou, 2007, Lemma 10.17] Let $p, q > 1$ be such that $\frac{1}{p} + \frac{1}{q} = 1$. Then

$$ab \leq \frac{1}{q} a^q + \frac{1}{p} b^p, \forall a, b.$$

Lemma 8.4. Let $a, c > 0$, $b \geq 0$, and $x \geq 0$ such that $x - a\sqrt{x} \leq c$. Then, $x \leq c + \frac{a^2}{2} + a\sqrt{\frac{a^2}{4} + c}$.

Proof. Let $y = \sqrt{x}$, then we have $y^2 - ay - c \leq 0$. Solving for y we have $y \leq \frac{a + \sqrt{a^2 + 4c}}{2}$. Hence, $x = y^2 \leq \frac{a^2 + 2a\sqrt{a^2 + 4c} + 4c}{4} = \frac{a^2}{2} + \frac{a}{2}\sqrt{a^2 + 4c} + c$. \square

Proof of Theorem 8.2. Denote by the total number of the mistakes of the Perceptron algorithm by $M = \sum_{t=1}^T \mathbf{1}[y_t \neq \tilde{y}_t]$.

First, note that the Perceptron algorithm can be thought as running Online Subgradient Descent (OSD) with a fixed stepsize η over the losses $\tilde{\ell}_t(\mathbf{x}) = -\langle \mathbf{1}[y_t \neq \tilde{y}_t] y_t \mathbf{z}_t, \mathbf{x} \rangle$ over $V = \mathbb{R}^d$. Indeed, OSD over such losses would update

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \mathbf{1}[y_t \neq \tilde{y}_t] y_t \mathbf{z}_t. \quad (8.2)$$

Now, as said above, η does not affect in any way the sign of the predictions, hence the Perceptron algorithm could be run with (8.2) and its predictions would be exactly the same. Hence, we have

$$\sum_{t=1}^T -\langle \mathbf{1}[y_t \neq \tilde{y}_t] y_t \mathbf{z}_t, \mathbf{x}_t \rangle + \sum_{t=1}^T \langle \mathbf{1}[y_t \neq \tilde{y}_t] y_t \mathbf{z}_t, \mathbf{u} \rangle \leq \frac{\|\mathbf{u}\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^T \mathbf{1}[y_t \neq \tilde{y}_t] \|\mathbf{z}_t\|_2^2, \forall \eta > 0.$$

Given that this inequality holds for any η , we can choose the ones that minimizes the r.h.s., to have

$$\sum_{t=1}^T -\langle \mathbf{1}[y_t \neq \tilde{y}_t] y_t \mathbf{z}_t, \mathbf{x}_t \rangle + \sum_{t=1}^T \langle \mathbf{1}[y_t \neq \tilde{y}_t] y_t \mathbf{z}_t, \mathbf{u} \rangle \leq \|\mathbf{u}\| \sqrt{\sum_{t=1}^T \mathbf{1}[y_t \neq \tilde{y}_t] \|\mathbf{z}_t\|_2^2} \leq \|\mathbf{u}\| R \sqrt{\sum_{t=1}^T \mathbf{1}[y_t \neq \tilde{y}_t]}. \quad (8.3)$$

Note that $-\mathbf{1}[y_t \neq \tilde{y}_t] y_t \langle \mathbf{z}_t, \mathbf{x}_t \rangle \geq 0$. Also, we have

$$\langle y_t \mathbf{z}_t, \mathbf{u} \rangle = 1 - (1 - \langle y_t \mathbf{z}_t, \mathbf{u} \rangle) \geq 1 - \max(1 - \langle y_t \mathbf{z}_t, \mathbf{u} \rangle, 0) = 1 - \ell(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t).$$

So, denoting by $\tau_t = \mathbf{1}[y_t \neq \tilde{y}_t]$, we can rewrite (8.3) as

$$\begin{aligned} \sum_{t=1}^T \tau_t &\leq \|\mathbf{u}\| R \sqrt{\sum_{t=1}^T \tau_t} + \sum_{t=1}^T \tau_t \ell(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t) \\ &\leq \|\mathbf{u}\| R \sqrt{\sum_{t=1}^T \tau_t} + \left(\sum_{t=1}^T \tau_t^p \right)^{1/p} \left(\sum_{t=1}^T \ell(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t)^q \right)^{1/q} \\ &= \|\mathbf{u}\| R \sqrt{\sum_{t=1}^T \tau_t} + \left(\sum_{t=1}^T \tau_t \right)^{1/p} \left(\sum_{t=1}^T \ell(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t)^q \right)^{1/q}, \end{aligned}$$

where we used Holder's inequality and $\frac{1}{p} + \frac{1}{q} = 1$.

Given that $M = \sum_{t=1}^T \tau_t$ and denoting by $L_q = \sum_{t=1}^T \ell(\langle \mathbf{z}_t, \mathbf{u} \rangle, y_t)$, we have

$$M \leq \|\mathbf{u}\| R \sqrt{M} + M^{1/p} L_q^{1/q}.$$

Let's now consider two cases. For $q = 1$, we can use Lemma 8.4 and have the stated bound. Instead, for $q > 1$, using Lemma 8.3 we have

$$M \leq \|\mathbf{u}\| R \sqrt{M} + M^{1/p} L_q^{1/q} \leq \|\mathbf{u}\| R \sqrt{M} + \frac{1}{p} M + \frac{1}{q} L_q,$$

that implies

$$M \left(1 - \frac{1}{p} \right) \leq \|\mathbf{u}\| R \sqrt{M} + \frac{1}{q} L_q.$$

Using the fact that $1 - \frac{1}{p} = \frac{1}{q}$, we have

$$M \leq q \|\mathbf{u}\| R \sqrt{M} + L_q.$$

Finally, using Lemma 8.4, we have the stated bound. □

8.3 History Bits

The Perceptron was proposed by Rosenblatt [1958]. The proof of convergence in the non-separable case for $q = 1$ is by Gentile [2003] and for $q = 2$ is from Freund and Schapire [1999]. The proof presented here is based on the one in Beygelzimer et al. [2017].

8.4 Exercises

Problem 8.1. *in the Perceptron mistake upper bound we let the competitor have any norm and we measure the loss of the competitor with the hinge loss $\ell^{\text{hinge}}(\hat{y}, y) = \max(\gamma - \hat{y}y, 0)$, where $\hat{y} \in \mathbb{R}$ and $y \in \{-1, 1\}$. Instead, we can equivalently constrain the competitor to have norm equal to 1 and measure the loss with the **hinge loss at margin gamma**: $\ell_\gamma^{\text{hinge}}(\hat{y}, y) = \frac{1}{\gamma} \max(\gamma - \hat{y}y, 0)$.*

Prove that $\ell_\gamma^{\text{hinge}}(\hat{y}, y)$ is still an upper bound on the zero-one loss over the prediction given by $\text{sign}(\hat{y})$.

Chapter 9

Parameter-free Online Linear Optimization

In the previous sections, we have shown that Online Mirror Descent (OMD) and Follow-The-Regularized-Leader (FTRL) achieves a regret of $O(\sqrt{T})$ for convex Lipschitz losses. We have also shown that for bounded domains these bounds are optimal up to constant multiplicative factors. However, in the unbounded case the bounds we get are suboptimal w.r.t. the dependency on the competitor. More in particular, let's consider an example with Online Subgradient Descent with $V = \mathbb{R}^d$ over 1-Lipschitz losses and learning rate $\eta = \frac{\alpha}{\sqrt{T}}$. We get the following regret guarantee

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \frac{\|\mathbf{u}\|_2^2}{2\eta} + \frac{\eta T}{2} = \frac{1}{2}\sqrt{T} \left(\frac{\|\mathbf{u}\|_2^2}{\alpha} + \alpha \right).$$

So, in order to get the best possible guarantee, we should know $\|\mathbf{u}\|_2$ and set $\alpha = \|\mathbf{u}\|_2$. As we said, this strategy does not work for a couple of reasons: i) we do not know $\|\mathbf{u}\|_2$ ii) if we guessed any value of $\|\mathbf{u}\|_2$ the adversary could easily change the losses to make that value completely wrong.

Far from being a technicality, this is an important issue as shown in the next example.

Example 9.1. Consider that we want to use OSD with online-to-batch conversion to minimize a function that is 1-Lipschitz. The convergence rate will be $O((\frac{\|\mathbf{u}^*\|_2^2}{\alpha} + \alpha)\sqrt{T})$ using a learning rate of $\eta = \frac{\alpha}{\sqrt{T}}$. Consider the case that $\|\mathbf{u}^*\|_2 = 100$, specifying $\alpha = 1$ will result in a convergence rate 100 times slower than specifying the optimal choice in hindsight $\alpha = 100$. Note that this is a real effect not an artifact of the proof. Indeed, it is intuitive that the optimal learning rate should be proportional to the distance between the initial point that algorithm picks and the optimal solution.

If we could tune the learning rate in the optimal way, we would get a regret of

$$\text{Regret}_T(\mathbf{u}) \leq \|\mathbf{u}\|_2 \sqrt{T}.$$

However, this is also impossible, because we proved a lower bound that says that the regret must be $\Omega(\|\mathbf{u}\|_2 \sqrt{T \ln(\|\mathbf{u}\|_2 + 1)})$.

In the following, we will show that it is possible to reduce any Online Convex Optimization (OCO) game to betting on a non-stochastic coin. This will allow us to use a radically different way to design OCO algorithms that will enjoy the optimal regret and will not require any parameter (e.g. learning rates, regularization weights) to be tuned. We call these kind of algorithms *parameter-free*.

9.1 Coin-Betting Game

Imagine the following repeated game:

- Set the initial Wealth to ϵ : $\text{Wealth}_0 = \epsilon$.
- In each round $t = 1, \dots, T$

- You bet $|x_t|$ money on side of the coin equal to $\text{sign}(x_t)$; you cannot bet more money than what you currently have.
- The adversary reveals the outcome of the coin $c_t \in \{-1, 1\}$.
- You gain money $x_t c_t$, that is $\text{Wealth}_t = \text{Wealth}_{t-1} + c_t x_t = \epsilon + \sum_{i=1}^t c_i x_i$.

Given that we cannot borrow money, we can codify the bets x_t as $\beta_t \text{Wealth}_{t-1}$, with $\beta_t \in [-1, 1]$. So, $|\beta_t|$ is the fraction of money to bet and $\text{sign}(\beta_t)$ the side of the coin on which we bet.

The aim of the game is to make as much money as possible. As usual, given the adversarial nature of the game, we cannot hope to always win money. Instead, we try to gain as much money as the strategy that bets a fixed amount of money $\beta^* \in [-1, 1]$ for the entire game.

Note that

$$\text{Wealth}_t = \text{Wealth}_{t-1} + c_t x_t = \text{Wealth}_{t-1} + \beta_t \text{Wealth}_{t-1} c_t = \text{Wealth}_{t-1} (1 + \beta_t c_t) = \epsilon \prod_{i=1}^{t-1} (1 + \beta_i c_i).$$

So, given the multiplicative nature of the wealth, it is also useful to take the logarithm of the ratio of the wealth of the algorithm and wealth of the optimal betting fraction. Hence, we want to minimize the following regret

$$\begin{aligned} \ln \max_{\beta \in [-1, 1]} \epsilon \prod_{t=1}^T (1 + \beta c_t) - \ln \text{Wealth}_T &= \ln \max_{\beta \in [-1, 1]} \epsilon \prod_{t=1}^T (1 + \beta c_t) - \ln \left(\epsilon \prod_{t=1}^T (1 + \beta_t c_t) \right) \\ &= \max_{\beta \in [-1, 1]} \sum_{t=1}^T \ln(1 + \beta c_t) - \sum_{t=1}^T \ln(1 + \beta_t c_t). \end{aligned}$$

In words, this is nothing else than the regret of an OCO game where the losses are $\ell_t(x) = -\ln(1 + x c_t)$ and $V = [-1, 1]$. We can also extend a bit the formulation allowing “continuous coins”, where $c_t \in [-1, 1]$ rather than in $\{-1, 1\}$.

Remark 9.2. *Note that the constraint to bet a fraction between -1 and 1 is not strictly necessary. We could allow the algorithm to bet more money than what it currently has, lending it some money in each round. However, the restriction makes the analysis easier because it allows the transformation above into an OCO problem, using the non-negativity of $1 + \beta_t c_t$.*

We could just use OMD or FTRL, taking special care of the non-Lipschitzness of the functions, but it turns out that there exists a better strategy specifically for this problem. There exists a very simple strategy to solve the coin-betting game above, that is called **Krichevsky-Trofimov (KT) bettor**. It simply says that on each time step t you bet $\beta_t = \frac{\sum_{i=1}^{t-1} c_i}{t}$. So, the algorithm is the following one.

Algorithm 9.1 Krichevsky-Trofimov bettor

Require: Initial money $\text{Wealth}_0 = \epsilon > 0$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Calculate the betting fraction $\beta_t = \frac{\sum_{i=1}^{t-1} c_i}{t}$
 - 3: Bet $x_t = \beta_t \text{Wealth}_{t-1}$, that is $|x_t|$ money on the side $\text{sign}(x_t) = \text{sign}(\beta_t)$
 - 4: Receive the coin outcome $c_t \in [-1, 1]$
 - 5: Win/lose $x_t c_t$, that is $\text{Wealth}_t = \text{Wealth}_{t-1} + c_t x_t$
 - 6: **end for**
-

For it, we can prove the following theorem.

Theorem 9.3 ([Cesa-Bianchi and Lugosi, 2006, Theorem 9.4]). *Let $c_t \in \{-1, 1\}$ for $t = 1, \dots, T$. Then, the KT bettor in Algorithm 9.1 guarantees*

$$\ln \text{Wealth}_T \geq \ln \max_{\beta \in [-1, 1]} \prod_{t=1}^T (1 + \beta c_t) - \frac{1}{2} \ln T - K,$$

where K is a universal constant.

Note that if the outcomes of the coin are skewed towards one side, the optimal betting fraction will gain an exponential amount of money, as proved in the next Lemma.

Lemma 9.4. *Let $g_t \in \{-1, 1\}, t = 1, \dots, T$. Then, we have*

$$\max_{\beta \in [-1, 1]} \exp \left(\sum_{t=1}^T \ln(1 - \beta g_t) \right) \geq \exp \left(\sum_{t=1}^T \frac{\left(\sum_{t=1}^T g_t \right)^2}{4T} \right).$$

Proof.

$$\begin{aligned} \max_{\beta \in [-1, 1]} \exp \left(\sum_{t=1}^T \ln(1 - \beta g_t) \right) &\geq \max_{\beta \in [-1/2, 1/2]} \exp \left(\sum_{t=1}^T \ln(1 - \beta g_t) \right) \geq \max_{\beta \in [-1/2, 1/2]} \exp \left(\beta \sum_{t=1}^T g_t - \beta^2 \sum_{t=1}^T g_t^2 \right) \\ &= \max_{\beta \in [-1/2, 1/2]} \exp \left(\beta \sum_{t=1}^T g_t - \beta^2 T \right) = \exp \left(\frac{\left(\sum_{t=1}^T g_t \right)^2}{4T} \right). \end{aligned}$$

where we used the elementary inequality $\ln(1 + x) \geq x - x^2$ for $x \in [-1/2, 1/2]$. \square

Hence, KT guarantees an exponential amount of money, paying only a \sqrt{T} penalty. It is possible to prove that the guarantee above for the KT algorithm is optimal to constant additive factors. Moreover, observe that the KT strategy does not require any parameter to be set: no learning rates, nor regularizer. That is, KT is *parameter-free*.

Also, we can extend the guarantee of the KT algorithm to the case in which the coin are “continuous, that is $c_t \in [-1, 1]$. We have the following Theorem.

Theorem 9.5 ([Orabona and Pal, 2016, Lemma 14]). *Let $c_t \in [-1, 1]$ for $t = 1, \dots, T$. Then, the KT bettor in Algorithm 9.1 guarantees*

$$\ln \text{Wealth}_T \geq \sum_{t=1}^T \frac{\left(\sum_{t=1}^T c_t \right)^2}{4T} - \frac{1}{2} \ln T - K,$$

where K is a universal constant.

So, we have introduced the coin-betting game, extended it to continuous coins and presented a simple and optimal parameter-free strategy. In the next Section, we show *how to use the KT bettor as a parameter-free 1-d OCO algorithm!*

9.2 Parameter-free 1d OCO through Coin-Betting

So, Theorem 9.3 tells us that we can win almost as much money as a strategy betting the optimal fixed fraction of money at each step. We only pay a logarithmic price in the log wealth, that corresponds to a $\frac{1}{\sqrt{T}}$ term in the actual wealth.

Now, let’s see why this problem is interesting in OCO. It turns out that *solving the coin-betting game with continuous coins is equivalent to solving a 1-dimensional unconstrained online linear optimization problem*. That is, a coin-betting algorithm is equivalent to design an online learning algorithm that produces a sequences of $x_t \in \mathbb{R}$ that minimize the 1-dimensional regret with linear losses:

$$\text{Regret}_T(u) := \sum_{t=1}^T g_t x_t - \sum_{t=1}^T g_t u,$$

where the g_t are adversarial and bounded. Without loss of generality, we will assume $g_t \in [-1, 1]$. Also, remembering that OCO games can be reduced to Online Linear Optimization (OLO) games, such reduction would effectively

reduces OCO to coin-betting! Moreover, through online-to-batch conversion, any stochastic 1-d problem could be reduced to a coin-betting game!

The key theorem that allows the conversion between OLO and coin-betting is the following one.

Theorem 9.6. *Let $\phi : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ be a proper closed convex function and let $\phi^* : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ be its Fenchel conjugate. An algorithm that generates $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T \in \mathbb{R}^d$ guarantees*

$$\forall \mathbf{g}_1, \dots, \mathbf{g}_T \in \mathbb{R}^d, \quad \epsilon - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle \geq \phi \left(- \sum_{t=1}^T \mathbf{g}_t \right)$$

where $\epsilon \in \mathbb{R}$, if and only if it guarantees

$$\forall \mathbf{u} \in \mathbb{R}^d, \quad \underbrace{\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle}_{\text{Regret}_T(\mathbf{u})} \leq \phi^*(\mathbf{u}) + \epsilon.$$

Proof. Let's prove the left to right implication.

$$\begin{aligned} \text{Regret}_T(\mathbf{u}) &= \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle - \phi \left(- \sum_{t=1}^T \mathbf{g}_t \right) + \epsilon \\ &\leq \sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \langle \boldsymbol{\theta}, \mathbf{u} \rangle - \phi(\boldsymbol{\theta}) + \epsilon = \phi^*(\mathbf{u}) + \epsilon. \end{aligned}$$

For the other implication, we have

$$\begin{aligned} - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle &= - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \\ &= \sup_{\mathbf{u} \in \mathbb{R}^d} - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \\ &\geq \sup_{\mathbf{u} \in \mathbb{R}^d} - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle - \phi^*(\mathbf{u}) - \epsilon = \phi \left(- \sum_{t=1}^T \mathbf{g}_t \right) - \epsilon. \quad \square \end{aligned}$$

To make sense of the above theorem, assume that we are considering a 1-d problem and $g_t \in [-1, 1]$. Then, guaranteeing a lower bound to

$$\epsilon - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle$$

can be done through a betting strategy that bets x_t money on the coins $c_t = -g_t$. So, the theorem implies that *proving a reward lower bound for the wealth in a coin-betting game implies a regret upper bound for the corresponding 1-dimensional OLO game*. However, proving a reward lower bound is easier because it doesn't depend on the competitor \mathbf{u} . Indeed, not knowing the norm of the competitor is exactly the reason why tuning the learning rates in OMD is hard!

This consideration immediately gives us the conversion between 1-d OLO and coin-betting: **the outcome of the coin is the negative of the subgradient of the losses on the current prediction**. Indeed, setting $c_t = -g_t$, we have that a coin-betting algorithm that bets x_t would give us

$$\text{Wealth}_T = \epsilon + \sum_{t=1}^T x_t c_t = \epsilon - \sum_{t=1}^T x_t g_t.$$

So, a lower bound on the wealth corresponds to a lower bound that can be used in Theorem 9.6. To obtain a regret guarantee, we only need to calculate the Fenchel conjugate of the reward function, assuming it can be expressed as a function of $\sum_{t=1}^T c_t$.

The last step is to reduce 1-d OCO to 1-d OLO. But, this is an easy step that we have done many times. Indeed, we have

$$\text{Regret}_T(u) = \sum_{t=1}^T \ell_t(x_t) - \sum_{t=1}^T \ell_t(u) \leq \sum_{t=1}^T x_t g_t - \sum_{t=1}^T g_t u,$$

where $g_t \in \partial \ell_t(x_t)$.

So, to summarize, the Fenchel conjugate of the wealth lower bound for the coin-betting game becomes the regret guarantee for the OCO game. In the next section, we specialize all these considerations to the KT algorithm.

9.2.1 KT as a 1d Online Convex Optimization Algorithm

Here, we want to use the considerations in the above section to use KT as a parameter-free 1-d OCO algorithm. First, let's see what such algorithm looks like. KT bets $x_t = \beta_t \text{Wealth}_{t-1}$, starting with ϵ money. Now, set $c_t = -g_t$ where $g_t \in \partial \ell_t(x_t)$ and assume the losses ℓ_t 1-Lipschitz. So, we get

$$x_t = -\frac{\sum_{i=1}^{t-1} g_i}{t} \left(\epsilon - \sum_{i=1}^{t-1} g_i x_i \right).$$

The pseudo-code is in Algorithm 9.2.

Algorithm 9.2 Krichevsky-Trofimov OCO

Require: $\epsilon > 0$ (any number between 1 and \sqrt{T})

- 1: **for** $t = 1$ **to** T **do**
 - 2: Predict $x_t = -\frac{\sum_{i=1}^{t-1} g_i}{t} \left(\epsilon - \sum_{i=1}^{t-1} g_i x_i \right)$
 - 3: Receive loss ℓ_t and pay $\ell_t(x_t)$
 - 4: Set $g_t \in \partial \ell_t(x_t)$
 - 5: **end for**
-

Let's now see what kind of regret we get. From Theorem 9.5, we have that the KT bettor guarantees the following lower bound on the wealth when used with $c_t = -g_t$:

$$\epsilon - \sum_{t=1}^T x_t g_t \geq \frac{\epsilon}{K\sqrt{T}} \exp \left(\frac{(\sum_{t=1}^T g_t)^2}{4T} \right).$$

So, we found the function ϕ , we just need ϕ^* or an upper bound to it, that can be found with the following Lemma.

Lemma 9.7. Define $f(x) = \beta \exp \frac{x^2}{2\alpha}$, for $\alpha, \beta > 0$, $x \geq 0$. Then

$$f^*(y) = |y| \sqrt{\alpha W \left(\frac{\alpha y^2}{\beta^2} \right)} - \beta \exp \left(\frac{W \left(\frac{\alpha y^2}{\beta^2} \right)}{2} \right) \leq |y| \sqrt{\alpha W \left(\frac{\alpha y^2}{\beta^2} \right)} - \beta \leq |y| \sqrt{\alpha \ln \left(1 + \frac{\alpha y^2}{\beta^2} \right)} - \beta.$$

where $W(x)$ is the Lambert function, i.e. $W : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ defined as to satisfy $x = W(x) \exp(W(x))$.

Proof. From the definition of Fenchel dual, we have

$$f^*(y) = \max_x x y - f(x) = \max_x x y - \beta \exp \frac{x^2}{2\alpha} \leq x^* y - \beta,$$

where $x^* = \operatorname{argmax}_x x y - f(x)$. We now use the fact that x^* satisfies $y = f'(x^*)$, to have $x^* = \operatorname{sign}(y) \sqrt{\alpha W \left(\frac{\alpha y^2}{\beta^2} \right)}$, where $W(\cdot)$ is the Lambert function. Using Lemma A.2 in the Appendix, we obtain the stated bound. \square

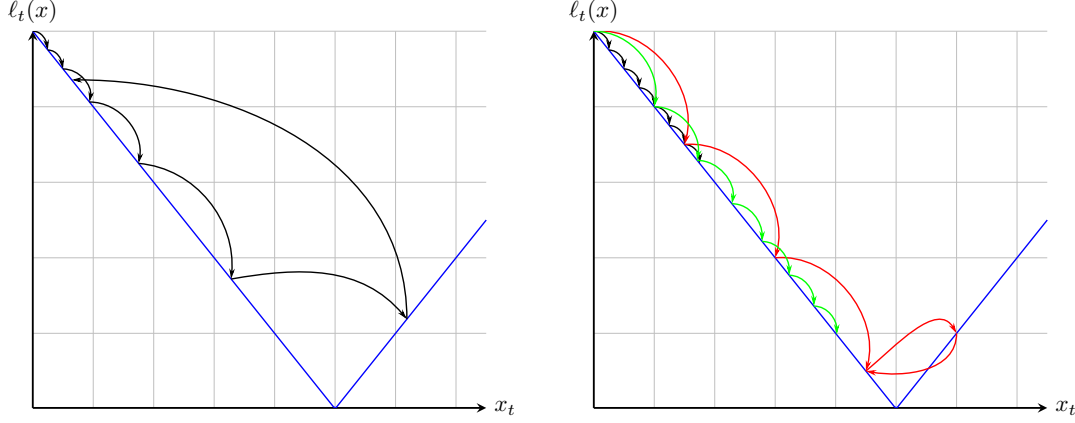


Figure 9.1: Behaviour of KT (left) and online gradient descent with various learning rates and same number of steps (right) when $\ell_t(x) = |x - 10|$.

So, the regret guarantee of KT used a 1d OLO algorithm is upper bounded by

$$\text{Regret}_T(u) = \sum_{t=1}^T \ell_t(x_t) - \sum_{t=1}^T \ell_t(u) \leq |u| \sqrt{4T \ln \left(\frac{\sqrt{2}|u|KT}{\epsilon} + 1 \right)} + \epsilon, \quad \forall u \in \mathbb{R},$$

where the only assumption was that the first derivatives (or sub-derivatives) of ℓ_t are bounded in absolute value by 1. Also, it is important to note that any setting of ϵ in $[1, \sqrt{T}]$ would not change the asymptotic rate.

To better appreciate this regret, compare this bound to the one of OMD with learning rate $\eta = \frac{\alpha}{\sqrt{T}}$:

$$\text{Regret}_T(u) = \sum_{t=1}^T \ell_t(x_t) - \sum_{t=1}^T \ell_t(u) \leq \frac{1}{2} \left(\frac{u^2}{\alpha} + \alpha \right) \sqrt{T}, \quad \forall u \in \mathbb{R}.$$

Hence, the coin-betting approach allows to get almost the optimal bound, without having to guess the correct learning rate! The price that we pay for this parameter-freeness is the log factor, that is optimal from our lower bound.

It is interesting also to look at what the algorithm would do on an easy problem, where $\ell_t(x) = |x - 10|$. In Figure 9.1, we show the different predictions that the KT algorithm and online subgradient descent (OSD) would do. Note how the convergence rate of OSD critically depends on the learning rate: too big will not give convergence and too small will make slow down the convergence. On the other hand, KT will go *exponentially fast* towards the minimum and then it will automatically backtrack. This exponential growth effectively works like a line search procedure that allows to get the optimal regret without tuning learning rates. Later in the iterations, KT will oscillate around the minimum, *automatically shrinking its steps, without any parameter to tune*. Of course, this is a simplified example. In a truly OCO game, the losses are different at each time step and the intuition behind the algorithm becomes more difficult. Yet, the optimality of the regret assures us that the KT strategy is the right strategy.

Next time, we will see that we can also reduce OCO in \mathbb{R}^d and learning with experts to coin-betting games.

9.3 Coordinate-wise Parameter-free OCO

We have already seen that it is always possible to decompose an OCO problem over the coordinate and use a different 1-dimensional Online Linear Optimization algorithm on each coordinate. In particular, we saw that

$$\begin{aligned} \text{Regret}_T(\mathbf{u}) &= \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \\ &= \sum_{t=1}^T \sum_{i=1}^d g_{t,i}(x_{t,i} - u_i) = \sum_{i=1}^d \sum_{t=1}^T g_{t,i}(x_{t,i} - u_i), \end{aligned}$$

where the $\sum_{t=1}^T g_{t,i}(x_{t,i} - u_i)$ is exactly the regret w.r.t. the linear losses constructed by the coordinate i of the subgradient.

Hence, if we have a 1-dimensional OLO algorithm, we can d copies of it, each one fed with the coordinate i of the subgradient. In particular, we might think to use the KT algorithm over each coordinate. The pseudo-code of this procedure is in Algorithm 9.3.

Algorithm 9.3 OCO with Coordinate-Wise Krichevsky-Trofimov

Require: $\epsilon > 0$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Output \mathbf{x}_t whose coordinates are $x_{t,i} = -\frac{\sum_{j=1}^{t-1} g_{j,i}}{t} \left(\epsilon - \sum_{j=1}^{t-1} g_{j,i} x_{j,i} \right)$ for $i = 1, \dots, d$
 - 3: Receive loss ℓ_t and pay $\ell_t(\mathbf{x}_t)$
 - 4: Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$
 - 5: **end for**
-

The regret bound we get is immediate: We just have to sum the regret over the coordinates.

Theorem 9.8. *With the notation in Algorithm 9.3, assume that $\|\mathbf{g}_t\|_\infty \leq 1$. Then, $\forall \mathbf{u} \in \mathbb{R}^d$, the following regret bounds hold*

$$\sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) \leq \sum_{i=1}^d |u_i| \sqrt{4T \ln \left(1 + \frac{\sqrt{2} u_i^2 K T}{\epsilon} \right)} + d\epsilon \leq \|\mathbf{u}\|_1 \sqrt{4T \ln \left(1 + \frac{\sqrt{2} \|\mathbf{u}\|_\infty^2 K T}{\epsilon} \right)} + d\epsilon,$$

where K is a universal constant.

Note that the Theorem above suggests that in high dimensional settings ϵ should be proportional to $\frac{1}{d}$.

9.4 Parameter-free in Any Norm

The above reductions works only with in a finite dimensional space. Moreover, it gives a dependency on the competitor w.r.t. the L_1 norm that might be undesirable. So, here we present another simple reduction from 1-dimensional OCO to infinite dimensions.

This reduction requires an unconstrained OCO algorithm for the 1-dimensional case and an algorithm for learning in d -dimensional (or infinite dimensional) balls. For the 1-dimensional learner, we could use the KT algorithm, while for learning in d -dimensional balls we can use, for example, Online Mirror Descent (OMD). Given these two learners, we decompose the problem of learning a vector \mathbf{x}_t in the problem of learning a *direction* and a *magnitude*. The regret of this procedure turns out to be just the sum of the regret of the two learners.

We can formalize this idea in the following Theorem.

Algorithm 9.4 Learning Magnitude and Direction Separately

Require: 1d Online learning algorithm \mathcal{A}_{1d} , Online learning algorithm \mathcal{A}_B with feasible set equal to the unit ball

$B \subset \mathbb{R}^d$ w.r.t. $\|\cdot\|$

- 1: **for** $t = 1$ **to** T **do**
 - 2: Get point $z_t \in \mathbb{R}$ from \mathcal{A}_{1d}
 - 3: Get point $\tilde{\mathbf{x}}_t \in B$ from \mathcal{A}_B
 - 4: Play $\mathbf{x}_t = z_t \tilde{\mathbf{x}}_t \in \mathbb{R}^d$
 - 5: Receive $\ell_t : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ and pay $\ell_t(\mathbf{x}_t)$
 - 6: Set $\mathbf{g}_t \in \partial \ell_t(\mathbf{x}_t)$
 - 7: Set $s_t = \langle \mathbf{g}_t, \tilde{\mathbf{x}}_t \rangle$
 - 8: Send $\ell_t^{\mathcal{A}_{1d}}(x) = s_t x$ as the t -th linear loss to \mathcal{A}_{1d}
 - 9: Send $\ell_t^{\mathcal{A}_B}(\mathbf{x}) = \langle \mathbf{g}_t, \mathbf{x} \rangle$ as the t -th linear loss to \mathcal{A}_B
 - 10: **end for**
-

Theorem 9.9. Denote by $\text{Regret}_T^{\mathcal{A}_B}(\mathbf{u})$ the linear regret of algorithm \mathcal{A}_B for any \mathbf{u} in the unit ball w.r.t a norm $\|\cdot\|$, and $\text{Regret}_T^{\mathcal{A}_{1d}}(u)$ the linear regret of algorithm \mathcal{A}_{1d} for any competitor $u \in \mathbb{R}$. Then, for any $\mathbf{u} \in \mathbb{R}^d$, Algorithm 9.4 guarantees regret

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle = \text{Regret}_T^{\mathcal{A}_{1d}}(\|\mathbf{u}\|) + \|\mathbf{u}\| \text{Regret}_T^{\mathcal{A}_B} \left(\frac{\mathbf{u}}{\|\mathbf{u}\|} \right).$$

Further, the subgradients s_t sent to \mathcal{A}_{1d} satisfy $|s_t| \leq \|\mathbf{g}_t\|_*$.

Proof. First, observe that $|s_t| \leq \|\mathbf{g}_t\|_* \|\tilde{\mathbf{x}}_t\| \leq \|\mathbf{g}_t\|_*$ since $\|\tilde{\mathbf{x}}_t\| \leq 1$ for all t . Now, compute:

$$\begin{aligned} \text{Regret}_T(\mathbf{u}) &= \sum_{t=1}^T \ell_t(\mathbf{x}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle = \sum_{t=1}^T \langle \mathbf{g}_t, z_t \tilde{\mathbf{x}}_t \rangle - \langle \mathbf{g}_t, \mathbf{u} \rangle \\ &= \underbrace{\sum_{t=1}^T (\langle \mathbf{g}_t, \tilde{\mathbf{x}}_t \rangle z_t - \langle \mathbf{g}_t, \tilde{\mathbf{x}}_t \rangle \|\mathbf{u}\|)}_{\text{linear regret of } \mathcal{A}_{1d} \text{ at } \|\mathbf{u}\| \in \mathbb{R}} + \sum_{t=1}^T (\langle \mathbf{g}_t, \tilde{\mathbf{x}}_t \rangle \|\mathbf{u}\| - \langle \mathbf{g}_t, \mathbf{u} \rangle) \\ &= \text{Regret}_T^{\mathcal{A}_{1d}}(\|\mathbf{u}\|) + \sum_{t=1}^T (\langle \mathbf{g}_t, \tilde{\mathbf{x}}_t \rangle \|\mathbf{u}\| - \langle \mathbf{g}_t, \mathbf{u} \rangle) \\ &= \text{Regret}_T^{\mathcal{A}_{1d}}(\|\mathbf{u}\|) + \|\mathbf{u}\| \sum_{t=1}^T \left(\langle \mathbf{g}_t, \tilde{\mathbf{x}}_t \rangle - \left\langle \mathbf{g}_t, \frac{\mathbf{u}}{\|\mathbf{u}\|} \right\rangle \right) \\ &= \text{Regret}_T^{\mathcal{A}_{1d}}(\|\mathbf{u}\|) + \|\mathbf{u}\| \text{Regret}_T^{\mathcal{A}_B} \left(\frac{\mathbf{u}}{\|\mathbf{u}\|} \right). \quad \square \end{aligned}$$

Remark 9.10. Note that the direction vector is not constrained to have norm equal to 1, yet this does not seem to affect the regret equality.

We can instantiate the above theorem using the KT betting algorithm for the 1d learner and OMD for the direction learner. Observe that in order to have the coin outcomes in $[-1, 1]$, we need to divide the losses by the Lipschitz constant. We obtain the following examples.

Example 9.11. Let \mathcal{A}_B be OSD with $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 \leq 1\}$ and learning rate $\eta_t = \frac{\sqrt{2D}}{2\sqrt{t}}$. Let \mathcal{A}_{1d} the KT algorithm for 1-dimensional OCO with $\epsilon = 1$. Assume the loss functions are 1-Lipschitz w.r.t. the $\|\cdot\|_2$. Then, using the construction in Algorithm 9.4, we have

$$\text{Regret}_T(\mathbf{u}) \leq O \left(\left(\|\mathbf{u}\|_2 \sqrt{\ln(\|\mathbf{u}\|_2 T + 1)} + \|\mathbf{u}\|_2 \right) \sqrt{T} + 1 \right), \quad \forall \mathbf{u} \in \mathbb{R}^d.$$

Using an online-to-batch conversion, this algorithm is a stochastic gradient descent procedure without learning rates to tune.

To better appreciate this kind of guarantee, let's take a look at the one of FTRL (OSD can be used in unbounded domains only with constant learning rates). With the regularizer $\psi_t(\mathbf{x}) = \frac{\sqrt{t}}{2\alpha} \|\mathbf{x}\|_2$ and 1-Lipschitz losses we get a regret of

$$\text{Regret}_T(\mathbf{u}) \leq \sqrt{T} \left(\frac{\|\mathbf{u}\|_2^2}{2\alpha} + \alpha \right).$$

So, to get the right dependency on $\|\mathbf{u}\|_2$ we need to tune α , but we saw this is impossible. On the other hand, the regret in Example 9.11 suffers from a logarithmic factor, that is the price to pay not to have to tune parameters.

In the same way, we can even have a parameter-free regret bound for L_p norms.

Example 9.12. Let \mathcal{A}_B be OMD with $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_p \leq 1\}$ and learning rate $\eta_t = \frac{\sqrt{2(p-1)D}}{2\sqrt{t}}$. Let \mathcal{A}_{1d} the KT algorithm for 1-dimensional OCO with $\epsilon = 1$. Assume the loss functions are 1-Lipschitz w.r.t. the $\|\cdot\|_q$. Then, using the construction in Algorithm 9.4, we have

$$\text{Regret}_T(\mathbf{u}) \leq O \left(\left(\|\mathbf{u}\|_p \sqrt{\ln(\|\mathbf{u}\|_p T + 1)} + \frac{\|\mathbf{u}\|_p}{\sqrt{p-1}} \right) \sqrt{T} + 1 \right), \forall \mathbf{u} \in \mathbb{R}^d.$$

If we want to measure the competitor w.r.t the L_1 norm, we have to use the same method we saw for OMD: Set $q = 2 \ln d$ and p such that $1/p + 1/q = 1$. Now, assuming that $\|\mathbf{g}_t\|_\infty \leq 1$, we have that $\|\mathbf{g}_t\|_q \leq d^{1/q}$. Hence, we have to divide all the losses by $d^{1/q}$ and, for all $\mathbf{u} \in \mathbb{R}^d$, we obtain

$$\begin{aligned} d^{-1/q} \sum_{t=1}^T (\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{u})) &\leq O \left(\left(\|\mathbf{u}\|_p \sqrt{\ln(\|\mathbf{u}\|_p T + 1)} + \|\mathbf{u}\|_p \sqrt{q-1} \right) \sqrt{T} + 1 \right) \\ &\leq O \left(\left(\|\mathbf{u}\|_1 \sqrt{\ln(\|\mathbf{u}\|_1 T + 1)} + \|\mathbf{u}\|_1 \sqrt{\ln d} \right) \sqrt{T} + 1 \right). \end{aligned}$$

Note that the regret against $\mathbf{u} = \mathbf{0}$ of the parameter-free construction is *constant*. It is important to understand that there is nothing special in the origin: We could translate the prediction by any offset and get a guarantee that treats the offset as the point with constant regret. This is shown in the next Proposition.

Proposition 9.13. Let \mathcal{A} an OLO algorithm that predicts \mathbf{x}_t and guarantees linear regret $\text{Regret}_T^{OLO}(\mathbf{u})$ for any $\mathbf{u} \in \mathbb{R}^d$. We have that the regret of the predictions $\hat{\mathbf{x}}_t = \mathbf{x}_t + \mathbf{x}_0$ for OCO is

$$\sum_{t=1}^T \ell_t(\hat{\mathbf{x}}_t) - \sum_{t=1}^T \ell_t(\mathbf{u}) \leq \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t + \mathbf{x}_0 - \mathbf{u} \rangle = \text{Regret}_T^{OLO}(\mathbf{u} - \mathbf{x}_0).$$

9.5 Combining Online Convex Optimization Algorithms

Finally, we now show a useful application of the parameter-free OCO algorithms property to have a constant regret against $\mathbf{u} = \mathbf{0}$.

Theorem 9.14. Let \mathcal{A}_1 and \mathcal{A}_2 two OLO algorithms that produces the predictions $\mathbf{x}_{t,1}$ and $\mathbf{x}_{t,2}$ respectively. Then, predicting with $\mathbf{x}_t = \mathbf{x}_{t,1} + \mathbf{x}_{t,2}$, we have for any $\mathbf{u} \in \mathbb{R}^d$

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle = \min_{\mathbf{u}=\mathbf{u}_1+\mathbf{u}_2} \text{Regret}_T^{\mathcal{A}_1}(\mathbf{u}_1) + \text{Regret}_T^{\mathcal{A}_2}(\mathbf{u}_2).$$

Moreover, if both algorithm guarantee a constant regret of ϵ against $\mathbf{u} = \mathbf{0}$, we have for any $\mathbf{u} \in \mathbb{R}^d$

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq \epsilon + \min(\text{Regret}_T^{\mathcal{A}_1}(\mathbf{u}), \text{Regret}_T^{\mathcal{A}_2}(\mathbf{u})).$$

Proof. Set $\mathbf{u}_1 + \mathbf{u}_2 = \mathbf{u}$. Then,

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_{t,1} \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u}_1 \rangle + \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_{t,2} \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u}_2 \rangle. \quad \square$$

In words, the above theorem allows us to combine online learning algorithm. If the algorithms we combine have constant regret against the null competitor, then we always get the best of the two guarantees.

Example 9.15. We can combine two parameter-free OCO algorithms, one that gives a bound that depends on the L_2 norm of the competitor and subgradients and another one specialized to the L_1/L_∞ norm of competitor/subgradients. The above theorem assures us that we will also get the best guarantee between the two, paying only a constant factor in the regret.

Of course, upper bounding the OCO regret with the linear regret, the above theorem also upper bounds the OCO regret.

9.6 Reduction to Learning with Experts

First, remember that the regret we got from OMD (and similarly for FTRL) is

$$\text{Regret}_T(\mathbf{u}) \leq O\left(\frac{KL(\mathbf{u}; \boldsymbol{\pi})}{\eta} + \eta T\right),$$

where $\boldsymbol{\pi}$ is the prior distribution on the experts and $KL(\cdot; \cdot)$ is the KL-divergence. As we reasoned in the OCO case, in order to set the learning rate we should know the value of $KL(\mathbf{u}; \boldsymbol{\pi})$. If we could set η to $\sqrt{\frac{KL(\mathbf{u}; \boldsymbol{\pi})}{T}}$, we would obtain a regret of $\sqrt{T KL(\mathbf{u}; \boldsymbol{\pi})}$. However, given the adversarial nature of the game, this is impossible. So, as we did in the OCO case, we will show that even this problem can be reduced to betting on a coin, obtaining optimal guarantees with a parameter-free algorithm.

Remark 9.16. There exists a different notion of regret for learning with experts. Order the cumulative losses of all actions from lowest to highest and define the ϵ -quantile regret to be the difference between the cumulative loss of the learner and the $\lceil \epsilon d \rceil$ -th element in the sorted list. In formulas

$$\text{Regret}_T(\epsilon) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{p}_t \rangle - \sum_{t=1}^T g_{t, i_\epsilon},$$

where $\epsilon \in [1/d, 1]$ and i_ϵ is the $\lceil \epsilon d \rceil$ -th best performing action.

This definition makes sense when the number of actions is very large and there are many of them that are close to the optimal one. Note the task of guaranteeing a small regret becomes easier with increasing ϵ . So, we would like to design an algorithm whose regret is inversely proportional to ϵ and it also does not depend on d in any way.

We now show that a regret that depends on the KL divergence between a generic competitor \mathbf{u} and $\boldsymbol{\pi}$ implies such regret guarantee. Define \mathbf{u}_ϵ as the vector that has the coordinates corresponding to the smallest $\lceil \epsilon d \rceil$ experts equal to $\frac{1}{\lceil \epsilon d \rceil}$ and 0 in the other coordinates. Also, assume to have an algorithm that guarantees an upper bound $\text{Regret}_T(\mathbf{u})$ equal to $F_T(KL(\mathbf{u}; \boldsymbol{\pi}))$ for any sequence of losses $\mathbf{g}_t \in [0, 1]^d$, where F is a non-decreasing function. Then, setting $\boldsymbol{\pi} = [1/d, \dots, 1/d]$, we have

$$\text{Regret}_T(\epsilon) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{p}_t \rangle - \sum_{t=1}^T g_{t, i_\epsilon} \leq \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{p}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u}_\epsilon \rangle \leq F_T(KL(\mathbf{u}_\epsilon; \boldsymbol{\pi})) = F_T\left(\ln \frac{d}{\lceil \epsilon d \rceil}\right) \leq F_T\left(\ln \frac{1}{\epsilon}\right),$$

where in the first inequality we used the fact that the average of a set of numbers is smaller than the biggest number in the set.

First, let's introduce some notation. Let $d \geq 2$ be the number of experts and Δ^d be the d -dimensional probability simplex. Let $\pi = [\pi_1, \pi_2, \dots, \pi_d] \in \Delta^d$ be any *prior* distribution. Let \mathcal{A} be a coin-betting algorithm. We will instantiate d copies of \mathcal{A} .

Consider any round t . Let $x_{t,i} \in \mathbb{R}$ be the bet of the i -th copy of \mathcal{A} . The LEA algorithm computes $\hat{\mathbf{p}}_t = [\hat{p}_{t,1}, \hat{p}_{t,2}, \dots, \hat{p}_{t,d}] \in \mathbb{R}_+^d$ as

$$\hat{p}_{t,i} = \pi_i \cdot \max(x_{t,i}, 0). \quad (9.1)$$

Then, the LEA algorithm predicts $\mathbf{p}_t = [p_{t,1}, p_{t,2}, \dots, p_{t,d}] \in \Delta^d$ as

$$\mathbf{p}_t = \begin{cases} \frac{\hat{\mathbf{p}}_t}{\|\hat{\mathbf{p}}_t\|_1}, & \text{if } \hat{\mathbf{p}}_t \neq \mathbf{0}, \\ \pi, & \text{otherwise.} \end{cases} \quad (9.2)$$

Then, the algorithm receives the reward vector $\mathbf{g}_t = [g_{t,1}, g_{t,2}, \dots, g_{t,d}] \in [0, 1]^d$. Finally, it feeds the reward to each copy of \mathcal{A} . The reward for the i -th copy of \mathcal{A} is $c_{t,i} \in [-1, 1]$ defined as

$$c_{t,i} = \begin{cases} \langle \mathbf{g}_t, \mathbf{p}_t \rangle - g_{t,i} & \text{if } x_{t,i} > 0, \\ \max(\langle \mathbf{g}_t, \mathbf{p}_t \rangle - g_{t,i}, 0) & \text{if } x_{t,i} \leq 0. \end{cases} \quad (9.3)$$

The construction above defines a LEA algorithm defined by the predictions \mathbf{p}_t , based on the algorithm \mathcal{A} . We can prove the following regret bound for it.

Theorem 9.17 (Regret Bound for Experts). *Let \mathcal{A} be a coin-betting algorithm that guarantees a wealth after t rounds with initial money equal to 1 of $\exp(f_t(\sum_{i=1}^t c'_i))$ for any sequence of continuous coin outcomes $c'_1, \dots, c'_t \in [-1, 1]$. Then, the regret of the LEA algorithm with prior $\pi \in \Delta^d$ that predicts at each round with \mathbf{p}_t in (9.2) satisfies*

$$\forall T \geq 0, \quad \forall \mathbf{u} \in \Delta^d, \quad \text{Regret}_T(\mathbf{u}) = \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{p}_t - \mathbf{u} \rangle \leq h(KL(\mathbf{u}; \pi)),$$

for any $h : \mathbb{R} \rightarrow \mathbb{R}$ concave and non-decreasing such that $x \leq h(f_T(x))$.

Proof. We first prove that $\sum_{i=1}^d \pi_i c_{t,i} x_{t,i} \leq 0$. Indeed,

$$\begin{aligned} \sum_{i=1}^d \pi_i c_{t,i} x_{t,i} &= \sum_{i: \pi_i x_{t,i} > 0} \pi_i \max(x_{t,i}, 0) (\langle \mathbf{g}_t, \mathbf{p}_t \rangle - g_{t,i}) + \sum_{i: \pi_i x_{t,i} \leq 0} \pi_i x_{t,i} \max(\langle \mathbf{g}_t, \mathbf{p}_t \rangle - g_{t,i}, 0) \\ &= \|\hat{\mathbf{p}}_t\|_1 \sum_{i=1}^d p_{t,i} (\langle \mathbf{g}_t, \mathbf{p}_t \rangle - g_{t,i}) + \sum_{i: \pi_i x_{t,i} \leq 0} \pi_i x_{t,i} \max(\langle \mathbf{g}_t, \mathbf{p}_t \rangle - g_{t,i}, 0) \\ &= 0 + \sum_{i: \pi_i x_{t,i} \leq 0} \pi_i x_{t,i} \max(\langle \mathbf{g}_t, \mathbf{p}_t \rangle - g_{t,i}, 0) \leq 0. \end{aligned}$$

The first equality follows from definition of $c_{t,i}$. To see the second equality, consider two cases: If $\pi_i x_{t,i} \leq 0$ for all i then $\|\hat{\mathbf{p}}_t\|_1 = 0$ and therefore both $\|\hat{\mathbf{p}}_t\|_1 \sum_{i=1}^d p_{t,i} (g_{t,i} - \langle \mathbf{g}_t, \mathbf{p}_t \rangle)$ and $\sum_{i: \pi_i x_{t,i} > 0} \pi_i \max(w_{t,i}, 0) (g_{t,i} - \langle \mathbf{g}_t, \mathbf{p}_t \rangle)$ are trivially zero. If $\|\hat{\mathbf{p}}_t\|_1 > 0$ then $\pi_i \max(x_{t,i}, 0) = \hat{p}_{t,i} = \|\hat{\mathbf{p}}_t\|_1 p_{t,i}$ for all i .

From the assumption on \mathcal{A} , we have for any sequence $\{c'_t\}_{t=1}^T$ such that $c'_t \in [-1, 1]$ that

$$\text{Wealth}_T = 1 + \sum_{t=1}^T c'_t x_t \geq \exp \left(f_T \left(\sum_{t=1}^T c'_t \right) \right). \quad (9.4)$$

So, inequality $\sum_{i=1}^d \pi_i c_{t,i} x_{t,i} \leq 0$ and (9.4) imply

$$\sum_{i=1}^d \pi_i \exp \left(f_T \left(\sum_{t=1}^T c_{t,i} \right) \right) \leq 1 + \sum_{i=1}^d \pi_i \sum_{t=1}^T c_{t,i} x_{t,i} \leq 1. \quad (9.5)$$

Now, for any competitor $\mathbf{u} \in \Delta^d$,

$$\begin{aligned}
\text{Regret}_T(\mathbf{u}) &= \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{p}_t - \mathbf{u} \rangle \\
&= \sum_{t=1}^T \sum_{i=1}^d u_i (\langle \mathbf{g}_t, \mathbf{p}_t \rangle - g_{t,i}) \\
&\leq \sum_{t=1}^T \sum_{i=1}^N u_i c_{t,i} \quad (\text{by definition of } c_{t,i}) \\
&\leq \sum_{i=1}^N u_i h \left(f_T \left(\sum_{t=1}^T c_{t,i} \right) \right) \quad (\text{definition of the } h(x)) \\
&\leq h \left[\sum_{i=1}^N u_i f_T \left(\sum_{t=1}^T c_{t,i} \right) \right] \quad (\text{by concavity of } h \text{ and Jensen inequality}) \\
&\leq h \left[KL(\mathbf{u}; \boldsymbol{\pi}) + \ln \left(\sum_{i=1}^N \pi_i \exp \left(f_T \left(\sum_{t=1}^T c_{t,i} \right) \right) \right) \right] \quad (\text{Fenchel-Young inequality}) \\
&\leq h(KL(\mathbf{u}; \boldsymbol{\pi})) \quad (\text{by (9.5)}) . \quad \square
\end{aligned}$$

Now, we could think to use the KT bettor with this theorem. However, we would obtain a sub-optimal regret guarantee. In fact, remembering the lower bound on the wealth of KT and setting $h(x) = \sqrt{4T(x + \frac{1}{2} \ln T + K)}$ where K is a universal constant, we have

$$\text{Regret}_T(\mathbf{u}) \leq O(\sqrt{T(KL(\mathbf{u}; \boldsymbol{\pi}) + \ln T)}) .$$

We might think that the $\ln T$ is the price we have to pay to adapt to the unknown competitor \mathbf{u} . However, it turns out it can be removed. In the next section, we see how to change the KT strategy to obtain the optimal guarantee.

9.6.1 A Betting Strategy that Loses at Most a Constant Fraction of Money

In the reduction before, if we use the KT betting strategy we would have a $\ln T$ term under the square root. It turns out that we can avoid that term if we know the number of rounds beforehand. Then, in case T is unknown we can just use a doubling trick, paying only a constant multiplicative factor in the regret.

The logarithmic term in the regret comes from the fact that the lower bound on the wealth is

$$O \left(\frac{1}{\sqrt{T}} \exp \left(\frac{(\sum_{t=1}^T c_t)^2}{4T} \right) \right) .$$

Note that in the case in which the number of heads in the sequence is equal to the number of tails, so that $\sum_{t=1}^T c_t = 0$, the guaranteed wealth becomes proportional to $\frac{1}{\sqrt{T}}$. So, for T that goes to infinity the bettor will lose all of its money.

Instead, we need a more conservative strategy that guarantees

$$O \left(\alpha \exp \left(\frac{(\sum_{t=1}^T c_t)^2}{4T} \right) \right) ,$$

for α small enough and independent of T . In this case, the betting strategy has to pace its betting, possibly with the knowledge of the duration of the game, so that even in the case that the number of heads is equal to the number of tails it will only lose a fraction of its money. At the same time, it will still gain an exponential amount of money when the coin outcomes are biased towards one side.

We will prove that this is possible, designing a new betting strategy.

Denote by $S_t = \sum_{i=1}^t c_i$ for $t = 1, \dots, T$ and define

$$F_t(x) = \epsilon \exp \left(\frac{x^2}{2(t+T)} - \sum_{i=1}^t \frac{1}{2(i+T)} \right), \quad (9.6)$$

$$\beta_t = \frac{F_t(S_{t-1} + 1) - F_t(S_{t-1} - 1)}{F_t(S_{t-1} + 1) + F_t(S_{t-1} - 1)}. \quad (9.7)$$

Note that if

$$(1 + \beta_t c_t) F_{t-1} \left(\sum_{i=1}^{t-1} c_i \right) \geq F_t \left(\sum_{i=1}^t c_i \right) \quad (9.8)$$

then, by induction, $\text{Wealth}_T \geq F_T \left(\sum_{t=1}^T c_t \right)$. In fact, we have

$$\text{Wealth}_t = (1 + \beta_t c_t) \text{Wealth}_{t-1} \geq (1 + \beta_t c_t) F_{t-1} \left(\sum_{i=1}^{t-1} c_i \right) \geq F_t \left(\sum_{i=1}^t c_i \right).$$

Hence, we have to prove that (9.8) is true in order to guarantee a minimum wealth of our betting strategy.

First, given that $\ln(1 + \beta_t c) - \frac{(x+c)^2}{2(t+T)}$ is a concave function of c , we have

$$\min_{c \in [-1, 1]} \ln(1 + \beta_t c) - \frac{(x+c)^2}{2(t+T)} = \min \left(\ln(1 + \beta_t) - \frac{(x+1)^2}{2(t+T)}, \ln(1 - \beta_t) - \frac{(x-1)^2}{2(t+T)} \right).$$

Also, our choice of β_t makes the two quantities above equal with $x = S_{t-1}$, that is

$$\ln(1 + \beta_t) - \ln F_t(S_{t-1} + 1) = \ln(1 - \beta_t) - \ln F_t(S_{t-1} - 1)$$

For other choices of β_t , the two alternatives would be different and the minimum one could always be the one picked by the adversary. Instead, making the two choices worst outcomes equivalent, we minimize the damage of the adversarial choice of the outcomes of the coin. So, we have that

$$\begin{aligned} \ln(1 + \beta_t c_t) - \ln F_t(S_t) &= \ln(1 + \beta_t c_t) + F_t(S_{t-1} + c_t) \\ &\geq \min_{c \in [-1, 1]} \ln(1 + \beta_t c) + \ln F_t(S_{t-1} + c) \\ &= -\ln \frac{F_t(S_{t-1} + 1) + F_t(S_{t-1} - 1)}{2} \\ &= -\ln \left[\exp \left(\frac{S_{t-1}^2 + 1}{2(t+T)} - \sum_{i=1}^t \frac{1}{2(i+T)} \right) \frac{1}{2} \left(\exp \left(\frac{S_{t-1}}{t+T} \right) + \exp \left(\frac{-S_{t-1}}{t+T} \right) \right) \right] \\ &= -\frac{S_{t-1}^2 + 1}{2(t+T)} - \ln \cosh \frac{S_{t-1}}{t+T} + \sum_{i=1}^t \frac{1}{2(i+T)} \\ &\geq -\frac{S_{t-1}^2}{2(t+T)} - \frac{S_{t-1}^2}{2(t+T)^2} + \sum_{i=1}^{t-1} \frac{1}{2(i+T)} \\ &\geq -\frac{S_{t-1}^2}{2(t+T)} - \frac{S_{t-1}^2}{2(t+T)(t-1+T)} + \sum_{i=1}^{t-1} \frac{1}{2(i+T)} \\ &= -\frac{S_{t-1}^2}{2(t-1+T)} + \sum_{i=1}^{t-1} \frac{1}{2(i+T)} \\ &= -\ln F_{t-1}(S_{t-1}), \end{aligned}$$

where in the second equality we used the definition of β_t and in the second inequality we used the fact that $\ln \cosh(x) \leq \frac{x^2}{2}, \forall x$.

Hence, given that (9.8) is true, this strategy guarantees

$$\text{Wealth}_T \geq \exp \left(\frac{\left(\sum_{t=1}^T c_t \right)^2}{4T} - \sum_{t=1}^T \frac{1}{2(i+T)} \right) \geq \exp \left(\frac{\left(\sum_{t=1}^T c_t \right)^2}{4T} - \frac{1}{2} \ln 2 \right) = \frac{\sqrt{2}}{2} \exp \left(\frac{\left(\sum_{t=1}^T c_t \right)^2}{2T} \right).$$

We can now use this betting strategy in the expert reduction in Theorem 9.17, setting $h(x) = \sqrt{4T(x + \frac{1}{2} \ln 2)}$, to have

$$\text{Regret}_T(\mathbf{u}) \leq \sqrt{4T \left(KL(\mathbf{u}; \boldsymbol{\pi}) + \frac{1}{2} \ln 2 \right)}. \quad (9.9)$$

Note that this betting strategy could also be used in the OCO reduction. Given that we removed the logarithmic term in the exponent, in the 1-dimensional case, we would obtain a regret of

$$\text{Regret}_T(u) \leq O \left(|u| \sqrt{T \ln \left(\frac{|u| \sqrt{T}}{\epsilon} + 1 \right)} + \epsilon \right),$$

where we gained in the \sqrt{T} term inside the logarithmic, instead of the T term of the KT algorithm. This implies that now we can set ϵ to \sqrt{T} and obtain an asymptotic rate of $O(\sqrt{T})$ rather than $O(\sqrt{T \ln T})$.

9.7 History Bits

The keyword “parameter-free” has been introduced in Chaudhuri et al. [2009] for a similar strategy for the learning with expert problem. It is now used as an umbrella term for all online algorithms that guarantee the optimal regret uniformly over the competitor class. The first algorithm for 1-d parameter-free OCO is from Streeter and McMahan [2012], but the bound was suboptimal. The algorithm was then extended to Hilbert spaces in Orabona [2013], still with a suboptimal bound. The optimal bound in Hilbert space was obtained in McMahan and Orabona [2014]. The idea of using a coin-betting to do parameter-free OCO was introduced in Orabona and Pal [2016]. The Krichevsky-Trofimov algorithm is from Krichevsky and Trofimov [1981] and its extension to the “continuous coin” is from Orabona and Pal [2016]. The regret-reward duality relationship was proved for the first time in McMahan and Orabona [2014]. Lemma A.2 is from Orabona and Pal [2016].

The approach of using a coordinate-wise version of the coin-betting algorithm was proposed in the first paper on parameter-free OLO in Streeter and McMahan [2012]. Recently, the same approach with a special coin-betting algorithm was also used for optimization of deep neural networks [Orabona and Tommasi, 2017]. Theorem 9.9 is from Cutkosky and Orabona [2018]. Note that the original theorem is more general because it works even in Banach spaces. The idea of combining two parameter-free OLO algorithms to obtain the best of the two guarantees is from Cutkosky [2019b].

Orabona and Pal [2016] proposed a different way to transform a coin-betting algorithm into an OCO algorithm that works in \mathbb{R}^d or even in Hilbert spaces. However, that approach seems to work only for the L_2 norm and it is not a black-box reduction. That said, the reduction in Orabona and Pal [2016] seems to have a better empirical performance compared to the one in Theorem 9.9.

There are also reductions that allow to transform an unconstrained OCO learner into a constrained one [Cutkosky and Orabona, 2018]. They work constructing a Lipschitz barrier function on the domain and passing to the algorithm the original subgradients plus the subgradients of the barrier function.

The first parameter-free algorithm for experts is from Chaudhuri et al. [2009], named NormalHedge, where they introduced the concept of ϵ -quantile regret and obtained a bound of $O(\sqrt{(T + \ln^2 d)(1 + \ln \frac{1}{\epsilon})})$, but without a closed formula update. Then, Chernov and Vovk [2010] removed the spurious dependency on d , again with an update without a closed form. Orabona and Pal [2016] showed that this guarantee can be efficiently obtained through the

novel reduction to coin-betting in Theorem 9.17. Later, these kind of regret guarantees were improved to depend on the sum of the squared losses rather than on time, but with an additional $\ln \ln T$ factor, in the Squint algorithm [Koolen and van Erven, 2015]. It is worth noting that the Squint algorithm can be interpreted exactly as a coin-betting algorithm plus the reduction in Theorem 9.17.

The betting strategy in (9.6) and (9.7) are new, and derived from the shifted-KT potentials in Orabona and Pal [2016]. The guarantee is the same obtained by the shifted-KT potentials, but the analysis can be done without knowing the properties of the gamma function.

9.8 Exercises

Problem 9.1. *While the original proof of the KT regret bound is difficult, it is possible to obtain a looser bound using the be-the-leader method in FTRL. In particular, it is easy to show a regret of $O(\ln T)$ for the log wealth.*

Problem 9.2. *Prove that $\ell_t(x) = -\ln(1 + z_t x)$ with $V = \{x \in \mathbb{R} : |x| \leq 1/2\}$ and $|z_t| \leq 1$, $t = 1, \dots, T$ are exp-concave. Then, using the Online Newton Step Algorithm, give an algorithm and a regret bound for a game with these losses. Finally, show a wealth guarantee of the corresponding coin-betting strategy.*

Problem 9.3. *Using the same proof technique in the section, find a betting strategy whose wealth depends on $\sum_{t=1}^T |c_t|$ rather than on T .*

Chapter 10

Multi-Armed Bandit

The Multi-Armed Bandit setting is similar to the Learning with Expert Advice (LEA) setting: In each round, we select one expert A_t and, differently from the full-information setting, we only observe the loss of that expert $g_{t,i}$. The aim is still to compete with the cumulative loss of the best expert in hindsight. The observed losses can be adversarial or stochastic, giving rise to adversarial and stochastic multi-armed bandits.

10.1 Adversarial Multi-Armed Bandit

As in the learning with expert case, we need randomization in order to have a sublinear regret. Indeed, this is just a harder problem than LEA. However, we will assume that the adversary is **oblivious**, that is, he decides the losses of all the rounds before the game starts, but with the knowledge of the online algorithm. This makes the losses deterministic quantities and it avoids the inadequacy in our definition of regret when the adversary is adaptive (see Arora et al. [2012]).

This kind of problems where we do not receive the full-information, i.e., we do not observe the loss vector, are called **bandit problems**. The name comes from the problem of a gambler who plays a pool of slot machines, that can be called “one-armed bandits”. On each round, the gambler places his bet on a slot machine and his goal is to win almost as much money as if he had known in advance which slot machine would return the maximal total reward.

In this problem, we clearly have an *exploration-exploitation trade-off*. In fact, on one hand we would like to play at the slot machine which, based on previous rounds, we believe will give us the biggest win. On the other hand, we have to explore the slot machines to find the best ones. On each round, we have to solve this trade-off.

Given that we do not observe completely observe the loss, we cannot use our two frameworks: Online Mirror Descent (OMD) and Follow-The-Regularized-Leader (FTRL) both needs the loss functions or at least lower bounds to them.

One way to solve this issue is to construct *stochastic estimates* of the unknown losses. This is a natural choice given that we already know that the prediction strategy has to be a randomized one. So, in each round t we construct a probability distribution over the arms \mathbf{x}_t and we sample one action A_t according to this probability distribution. Then, we only observe the coordinate A_t of the loss vector $\mathbf{g}_t \in \mathbb{R}^d$. One possibility to have a stochastic estimate of the losses is to use an *importance-weighted estimator*: Construct the estimator $\tilde{\mathbf{g}}_t$ of the unknown vector \mathbf{g}_t in the following way:

$$\tilde{\mathbf{g}}_t = \begin{cases} \frac{g_{t,i}}{x_{t,i}}, & i = A_t \\ 0, & \text{otherwise} \end{cases}.$$

Note that this estimator has all the coordinates equal to 0, except the coordinate corresponding the arm that was pulled.

This estimator is unbiased, that is $\mathbb{E}_{A_t}[\tilde{\mathbf{g}}_t] = \mathbf{g}_t$. To see why, note that $\tilde{g}_{t,i} = \mathbf{1}[A_t = i] \frac{g_{t,i}}{x_{t,i}}$ and $\mathbb{E}_{A_t}[\mathbf{1}[A_t = i]] = x_{t,i}$. Hence, for $i = 1, \dots, d$, we have

$$\mathbb{E}_{A_t}[\tilde{g}_{t,i}] = \mathbb{E}_{A_t} \left[\mathbf{1}[A_t = i] \frac{g_{t,i}}{x_{t,i}} \right] = \frac{g_{t,i}}{x_{t,i}} \mathbb{E}_{A_t}[\mathbf{1}[A_t = i]] = g_{t,i}.$$

Let's also calculate the (uncentered) variance of the coordinates of this estimator. We have

$$\mathbb{E}_{A_t}[\tilde{g}_{t,i}^2] = \mathbb{E}_{A_t} \left[\mathbf{1}[A_t = i] \frac{g_{t,i}^2}{x_{t,i}^2} \right] = \frac{g_{t,i}^2}{x_{t,i}}.$$

We can now think of using OMD with these estimated losses and an entropic regularizer. Hence, assume $\|g_t\|_\infty \leq L_\infty$ and set $\psi : \mathbb{R}_+^d \rightarrow \mathbb{R}$ defined as $\psi(\mathbf{x}) = \sum_{i=1}^d x_i \ln x_i$, that is the unnormalized negative entropy. Also, set $\mathbf{x}_1 = [1/d, \dots, 1/d]$. Using the OMD analysis, we have

$$\sum_{t=1}^T \langle \tilde{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \tilde{g}_t, \mathbf{u} \rangle \leq \frac{\ln d}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \|\tilde{g}_t\|_\infty^2.$$

We can now take the expectation at both sides and get

$$\begin{aligned} \mathbb{E} \left[\sum_{t=1}^T g_{t,A_t} \right] - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle &= \mathbb{E} \left[\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle \right] - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \\ &= \mathbb{E} \left[\sum_{t=1}^T \langle \tilde{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \tilde{g}_t, \mathbf{u} \rangle \right] \\ &\leq \frac{\ln d}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \mathbb{E}[\|\tilde{g}_t\|_\infty^2] \\ &\leq \frac{\ln d}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \sum_{i=1}^d \frac{g_{t,i}^2}{x_{t,i}}. \end{aligned} \tag{10.1}$$

We are now in troubles, because the terms in the sum scale as $\max_{i=1,\dots,d} \frac{1}{x_{t,i}}$. So, we need a way to control the smallest probability over the arms.

One way to do it, is to take a convex combination of \mathbf{x}_t and a uniform probability. That is, we can predict with $\tilde{\mathbf{x}}_t = (1 - \alpha)\mathbf{x}_t + \alpha[1/d, \dots, 1/d]$, where α will be chosen in the following. So, α can be seen as the minimum amount of exploration we require to the algorithm. Its value will be chosen by the regret analysis to optimally trade-off exploration vs exploitation. The resulting algorithm is in Algorithm 10.1.

Algorithm 10.1 Exponential Weights with Explicit Exploration for Multi-Armed Bandit

Require: $\eta, \alpha > 0$

- 1: Set $\mathbf{x}_1 = [1/d, \dots, 1/d]$
 - 2: **for** $t = 1$ **to** T **do**
 - 3: Set $\tilde{\mathbf{x}}_t = (1 - \alpha)\mathbf{x}_t + \alpha[1/d, \dots, 1/d]$
 - 4: Draw A_t according to $P(A_t = i) = \tilde{x}_{t,i}$
 - 5: Select expert A_t
 - 6: Observe *only* the loss of the selected arm $g_{t,A_t} \in [-L_\infty, L_\infty]$ and pay it
 - 7: Construct the estimate $\tilde{g}_t = \begin{cases} \frac{g_{t,i}}{\tilde{x}_{t,i}}, & i = A_t \\ 0, & \text{otherwise} \end{cases}$
 - 8: $x_{t+1,i} \propto x_{t,i} \exp(-\eta \tilde{g}_{t,i})$, $i = 1, \dots, d$
 - 9: **end for**
-

The same probability distribution is used in the estimator:

$$\tilde{g}_t = \begin{cases} \frac{g_{t,i}}{x_{t,i}}, & i = A_t \\ 0, & \text{otherwise} \end{cases}. \tag{10.2}$$

We can have that $\frac{1}{\tilde{x}_{t,i}} \leq \frac{d}{\alpha}$. However, we pay a price in the bias introduced:

$$\sum_{t=1}^T \langle \tilde{\mathbf{g}}_t, \tilde{\mathbf{x}}_t - \mathbf{u} \rangle = \sum_{t=1}^T \langle \tilde{\mathbf{g}}_t, (1 - \alpha)\mathbf{x}_t - \mathbf{u} \rangle + \frac{\alpha}{d} \sum_{t=1}^T \sum_{i=1}^d \tilde{g}_{t,i}.$$

Observing that $\mathbb{E}[\sum_{i=1}^d \tilde{g}_{t,i}] = \sum_{i=1}^d g_{t,i} \leq dL_\infty$, we have

$$\mathbb{E} \left[\sum_{t=1}^T \langle \tilde{\mathbf{g}}_t, \tilde{\mathbf{x}}_t - \mathbf{u} \rangle \right] \leq \mathbb{E} \left[\sum_{t=1}^T \langle \tilde{\mathbf{g}}_t, (1 - \alpha)\mathbf{x}_t - \mathbf{u} \rangle \right] + \alpha L_\infty T \leq \mathbb{E}[\text{Regret}_T(\mathbf{u})] + \alpha L_\infty T.$$

Putting together the last inequality and the upper bound to the expected regret in (10.1), we have

$$\mathbb{E} \left[\sum_{t=1}^T g_{t,A_t} \right] - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq \frac{\ln d}{\eta} + \frac{\eta d^2 L_\infty^2 T}{2\alpha} + \alpha L_\infty T.$$

Setting $\alpha \propto \sqrt{d^2 L_\infty \eta}$ and $\eta \propto \left(\frac{\ln d}{d L_\infty^{3/2} T} \right)^{2/3}$, we obtain a regret of $O(L_\infty (dT)^{2/3} \ln^{1/3} d)$.

This is way worse than the $O(\sqrt{T \ln d})$ of the full-information case. However, while it is expected that the bandit case must be more difficult than the full information one, it turns out that this is not the optimal strategy.

10.1.1 Exponential-weight algorithm for Exploration and Exploitation: Exp3

It turns out that the algorithm above actually works, even without the mixing with the uniform distribution! We were just too loose in our regret guarantee. So, we will analyse the following algorithm, that is called Exponential-weight algorithm for Exploration and Exploitation (Exp3), that is nothing else than OMD with entropic regularizer and stochastic estimates of the losses. Note that now we will assume that $g_{t,i} \in [0, L_\infty]$.

Algorithm 10.2 Exp3

Require: $\eta > 0$

- 1: $\mathbf{x}_1 = [1/d, \dots, 1/d]$
 - 2: **for** $t = 1$ **to** T **do**
 - 3: Draw A_t according to $P(A_t = i) = x_{t,i}$
 - 4: Select expert A_t
 - 5: Observe *only* the loss of the selected arm $g_{t,A_t} \in [0, L_\infty]$ and pay it
 - 6: Construct the estimate $\tilde{\mathbf{g}}_t = \begin{cases} \frac{g_{t,i}}{x_{t,i}}, & i = A_t \\ 0, & \text{otherwise} \end{cases}$
 - 7: $x_{t+1,i} \propto x_{t,i} \exp(-\eta \tilde{g}_{t,i})$, $i = 1, \dots, d$
 - 8: **end for**
-

Let's take another look to the regret guarantee we have. From the OMD analysis, we have the following one-step inequality that holds for any \mathbf{g}_t

$$\langle \eta \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq B_\psi(\mathbf{u}; \mathbf{x}_t) - B_\psi(\mathbf{u}; \mathbf{x}_{t+1}) - B_\psi(\mathbf{x}_{t+1}; \mathbf{x}_t) + \langle \eta \mathbf{g}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle.$$

Let's now focus on the term $-B_\psi(\mathbf{x}_{t+1}; \mathbf{x}_t) + \langle \eta \mathbf{g}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle$. We said that for a twice differentiable function f , there exists $\alpha \in [0, 1]$ such that $B_f(\mathbf{x}; \mathbf{y}) = \frac{1}{2}(\mathbf{x} - \mathbf{y})^\top \nabla^2 f(\mathbf{z})(\mathbf{x} - \mathbf{y})$, where $\mathbf{z} = \alpha \mathbf{x} + (1 - \alpha)\mathbf{y}$. Hence, there exists $\alpha_t \in [0, 1]$ such that $\mathbf{z}_t = \alpha_t \mathbf{x}_t + (1 - \alpha_t)\mathbf{x}_{t+1}$ and

$$B_\psi(\mathbf{x}_{t+1}; \mathbf{x}_t) = \frac{1}{2}(\mathbf{x}_t - \mathbf{x}_{t+1})^\top \nabla^2 \psi(\mathbf{z}_t)(\mathbf{x}_t - \mathbf{x}_{t+1}) = \frac{1}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_{\nabla^2 \psi(\mathbf{z}_t)}^2.$$

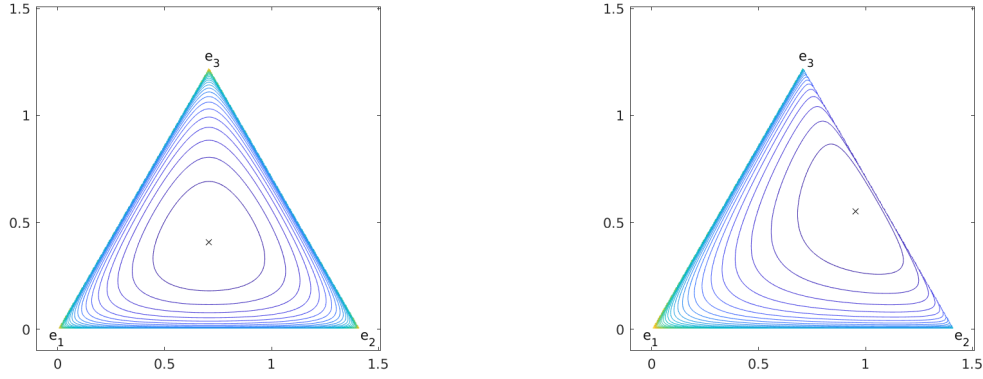


Figure 10.1: Contour plots of the KL divergence in 3-dimensions when $\mathbf{x}_t = [1/3, 1/3, 1/3]$ (left) and when $\mathbf{x}_t = [0.1, 0.45, 0.45]$ (right).

So, assuming the Hessian in \mathbf{z}_t to be PD, we can bound the last two terms in the one-step inequality of OMD as

$$\begin{aligned} -B_\psi(\mathbf{x}_{t+1}; \mathbf{x}_t) + \langle \eta \mathbf{g}_t, \mathbf{x}_t - \mathbf{x}_{t+1} \rangle &\leq -B_\psi(\mathbf{x}_{t+1}; \mathbf{x}_t) + \frac{\eta^2}{2} \mathbf{g}_t^\top (\nabla^2 \psi(\mathbf{z}_t))^{-1} \mathbf{g}_t + \frac{1}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_{\nabla^2 \psi(\mathbf{z}_t)}^2 \\ &= \frac{\eta^2}{2} \mathbf{g}_t^\top (\nabla^2 \psi(\mathbf{z}_t))^{-1} \mathbf{g}_t, \end{aligned} \quad (10.3)$$

where we used Fenchel-Young inequality with the function $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|_{\nabla^2 \psi(\mathbf{x}_t)}^2$ and $\mathbf{z}_t = \alpha_t \mathbf{x}_t + (1 - \alpha_t) \mathbf{x}_{t+1}$ and $\alpha_t \in [0, 1]$.

When we use the strong convexity, we are upper bounding the terms in the sum with the inverse of the smallest eigenvalue of the Hessian of the regularizer. However, we can do better if we consider the local norm. In fact, in the coordinates where \mathbf{x}_t is small, we have a smaller growth of the divergence. This can be seen also graphically in Figure 10.1. Indeed, for the entropic regularizer, we have that the Hessian is a diagonal matrix:

$$(\nabla^2 \psi(\mathbf{z}))_{ii} = \frac{1}{z_i} \quad i = 1, \dots, d.$$

This expression of the Hessian a regret of

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq \frac{B_\psi(\mathbf{u}; \mathbf{x}_1)}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \sum_{i=1}^d z_{t,i} g_{t,i}^2,$$

where $\mathbf{z}_t = \alpha_t \mathbf{x}_t + (1 - \alpha_t) \mathbf{x}_{t+1}$ and $\alpha_t \in [0, 1]$. Note that for any $\alpha_t \in [0, 1]$ \mathbf{z}_t is in the simplex, so this upper bound is always better than

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq \frac{B_\psi(\mathbf{u}; \mathbf{x}_1)}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \|\mathbf{g}_t\|_\infty^2,$$

that we derived just using the strong convexity of the entropic regularizer.

However, we do not know the exact value of \mathbf{z}_t , but only that it is on the line segment between \mathbf{x}_t and \mathbf{x}_{t+1} . Yet, if you could say that $z_{t,i} = O(x_{t,i})$, in the bandit case we would obtain an expected regret guarantee of $O(\sqrt{dT \ln d})$, greatly improving the bound we proved above! In other words, it might be possible to get the regret guarantee

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq \frac{\ln d}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \sum_{i=1}^d x_{t,i} g_{t,i}^2, \quad \forall \mathbf{u} \in V, \quad (10.4)$$

where $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_1 = 1, x_i \geq 0\}$. Let's see how.

Remark 10.1. While it is possible to prove (10.4) from first principles using the specific properties for the entropic regularizer, such proof will not shed any light of what is actually going on. So, in the following we will instead try to prove such regret in a very general way. Indeed, this general proof will allow us to easily prove the optimal bound for multi-armed bandits using OMD with the Tsallis entropy as regularizer.

Now, for a generic ψ , consider the OMD algorithm that produces the predictions in two steps:

- Set $\tilde{\mathbf{x}}_{t+1}$ such that $\nabla\psi(\tilde{\mathbf{x}}_{t+1}) = \nabla\psi(\mathbf{x}_t) - \eta\mathbf{g}_t$.
- Set $\mathbf{x}_{t+1} = \operatorname{argmin}_{\mathbf{x} \in V} B_\psi(\mathbf{x}; \tilde{\mathbf{x}}_{t+1})$.

As we showed, under weak conditions, these two steps are equivalent to the usual OMD single-step update.

Now, the idea is to consider an alternative analysis of OMD that explicitly depends on $\tilde{\mathbf{x}}_{t+1}$, the new prediction before the Bregman projection step.

First, let's state the Generalized Pythagorean Theorem for Bregman divergences.

Lemma 10.2. Let $\tilde{\mathbf{x}} \in \operatorname{int} \operatorname{dom} \psi$ and define $\mathbf{x}' = \operatorname{argmin}_{\mathbf{y} \in V} B_\psi(\mathbf{y}; \tilde{\mathbf{x}})$, then $B_\psi(\mathbf{x}; \tilde{\mathbf{x}}) \geq B_\psi(\mathbf{x}; \mathbf{x}') + B_\psi(\mathbf{x}'; \tilde{\mathbf{x}})$ for all $\mathbf{x} \in V$.

Proof. From the first order optimality condition of \mathbf{x}' we have that $\langle \nabla\psi(\mathbf{x}') - \nabla\psi(\tilde{\mathbf{x}}), \mathbf{x} - \mathbf{x}' \rangle \geq 0, \forall \mathbf{x} \in V$. Hence, we have

$$\begin{aligned} B_\psi(\mathbf{x}; \mathbf{x}') + B_\psi(\mathbf{x}'; \tilde{\mathbf{x}}) &= \psi(\mathbf{x}) - \psi(\mathbf{x}') - \langle \nabla\psi(\mathbf{x}'), \mathbf{x} - \mathbf{x}' \rangle + \psi(\mathbf{x}') - \psi(\tilde{\mathbf{x}}) - \langle \nabla\psi(\tilde{\mathbf{x}}), \mathbf{x}' - \tilde{\mathbf{x}} \rangle \\ &= \psi(\mathbf{x}) - \psi(\tilde{\mathbf{x}}) - \langle \nabla\psi(\mathbf{x}'), \mathbf{x} - \mathbf{x}' \rangle - \langle \nabla\psi(\tilde{\mathbf{x}}), \mathbf{x}' - \tilde{\mathbf{x}} \rangle \\ &\leq \psi(\mathbf{x}) - \psi(\tilde{\mathbf{x}}) - \langle \nabla\psi(\tilde{\mathbf{x}}), \mathbf{x} - \mathbf{x}' \rangle - \langle \nabla\psi(\tilde{\mathbf{x}}), \mathbf{x}' - \tilde{\mathbf{x}} \rangle \\ &= \psi(\mathbf{x}) - \psi(\tilde{\mathbf{x}}) - \langle \nabla\psi(\tilde{\mathbf{x}}), \mathbf{x} - \tilde{\mathbf{x}} \rangle \\ &= B_\psi(\mathbf{x}; \tilde{\mathbf{x}}). \end{aligned} \quad \square$$

The Generalized Pythagorean Theorem is often used to prove that the Bregman divergence between any point \mathbf{x} in V and an arbitrary point $\tilde{\mathbf{x}}$ decreases when we consider the Bregman projection $\tilde{\mathbf{x}}$ in V .

We are now ready to prove our regret guarantee.

Lemma 10.3. For the two-steps OMD update above the following regret bound holds:

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq \frac{B_\psi(\mathbf{u}; \mathbf{x}_1)}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \mathbf{g}_t^\top (\nabla^2 \psi(\mathbf{z}_t))^{-1} \mathbf{g}_t, \quad \forall \mathbf{u} \in V,$$

where $\mathbf{z}_t = \alpha_t \mathbf{x}_t + (1 - \alpha_t) \tilde{\mathbf{x}}_{t+1}$ and $\alpha_t \in [0, 1]$.

Proof. From the update rule, we have that

$$\begin{aligned} \langle \eta \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle &= \langle \nabla\psi(\mathbf{x}_t) - \nabla\psi(\tilde{\mathbf{x}}_{t+1}), \mathbf{x}_t - \mathbf{u} \rangle \\ &= B_\psi(\mathbf{u}; \mathbf{x}_t) - B_\psi(\mathbf{u}; \tilde{\mathbf{x}}_{t+1}) + B_\psi(\mathbf{x}_t; \tilde{\mathbf{x}}_{t+1}) \\ &\leq B_\psi(\mathbf{u}; \mathbf{x}_t) - B_\psi(\mathbf{u}; \mathbf{x}_{t+1}) + B_\psi(\mathbf{x}_t; \tilde{\mathbf{x}}_{t+1}) - B_\psi(\mathbf{x}_{t+1}; \tilde{\mathbf{x}}_{t+1}) \\ &\leq B_\psi(\mathbf{u}; \mathbf{x}_t) - B_\psi(\mathbf{u}; \mathbf{x}_{t+1}) + B_\psi(\mathbf{x}_t; \tilde{\mathbf{x}}_{t+1}), \end{aligned}$$

where in the second equality we used the 3-points equality for the Bregman divergences and the Generalized Pythagorean Theorem in the first inequality. Hence, summing over time we have

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq \frac{B_\psi(\mathbf{u}; \mathbf{x}_1)}{\eta} + \frac{1}{\eta} \sum_{t=1}^T B_\psi(\mathbf{x}_t; \tilde{\mathbf{x}}_{t+1}).$$

So, as we did in (10.3), we have

$$\begin{aligned}
B_\psi(\mathbf{x}_t; \tilde{\mathbf{x}}_{t+1}) &= \psi(\mathbf{x}_t) - \psi(\tilde{\mathbf{x}}_{t+1}) - \langle \nabla \psi(\tilde{\mathbf{x}}_{t+1}), \mathbf{x}_t - \tilde{\mathbf{x}}_{t+1} \rangle \\
&= -B_\psi(\tilde{\mathbf{x}}_{t+1}; \mathbf{x}_t) + \langle \nabla \psi(\mathbf{x}_t) - \nabla \psi(\tilde{\mathbf{x}}_{t+1}), \mathbf{x}_t - \tilde{\mathbf{x}}_{t+1} \rangle \\
&= -B_\psi(\tilde{\mathbf{x}}_{t+1}; \mathbf{x}_t) + \langle \eta \mathbf{g}_t, \mathbf{x}_t - \tilde{\mathbf{x}}_{t+1} \rangle \\
&\leq -B_\psi(\tilde{\mathbf{x}}_{t+1}; \mathbf{x}_t) + \frac{1}{2} \eta^2 \mathbf{g}_t^\top (\nabla^2 \psi(\mathbf{z}_t))^{-1} \mathbf{g}_t + \frac{1}{2} \|\mathbf{x}_t - \tilde{\mathbf{x}}_{t+1}\|_{\nabla^2 \psi(\mathbf{z}_t)}^2 \\
&= \frac{1}{2} \eta^2 \mathbf{g}_t^\top (\nabla^2 \psi(\mathbf{z}_t))^{-1} \mathbf{g}_t,
\end{aligned}$$

where $\mathbf{z}_t = \alpha_t \mathbf{x}_t + (1 - \alpha_t) \tilde{\mathbf{x}}_{t+1}$ and $\alpha_t \in [0, 1]$.

Putting all together, we have the stated bound. \square

This time it might be easier to get a handle over $z_{t,i}$. Given that we only need an upper bound, we can just take a look at $x_{t,i}$ and $\tilde{x}_{t+1,i}$ and see which one is bigger. This is easy to do: using the update rule we have

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

that is

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

Assuming $g_{t,i} \geq 0$, we have $\tilde{x}_{t+1,i} \leq x_{t,i}$ that implies $z_{t,i} \leq x_{t,i}$.

Overall, we have the following improved regret guarantee for the Learning with Experts setting with positive losses.

Theorem 10.4. Assume $g_{t,i} \geq 0$ for $t = 1, \dots, T$ and $i = 1, \dots, d$. Let $V = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_1 = 1, x_i \geq 0\}$ and $\eta > 0$. Using OMD with the entropic regularizer $\psi : \mathbb{R}_+^d \rightarrow \mathbb{R}$ defined as $\psi(\mathbf{x}) = \sum_{i=1}^d x_i \ln x_i$, learning rate η , and $\mathbf{x}_1 = [1/d, \dots, 1/d]$ gives the following regret guarantee

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq \frac{\ln d}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \sum_{i=1}^d x_{t,i} g_{t,i}^2, \quad \forall \mathbf{u} \in V.$$

Armed with this new tool, we can now turn to the multi-armed bandit problem again.

Let's now consider the OMD with entropic regularizer, learning rate η , and set $\tilde{\mathbf{g}}_t$ equal to the stochastic estimate of \mathbf{g}_t , as in Algorithm 10.2. Applying Theorem 10.4 and taking expectation, we have

$$\mathbb{E} \left[\sum_{t=1}^T g_{t,A_t} \right] - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle = \mathbb{E} \left[\sum_{t=1}^T \langle \tilde{\mathbf{g}}_t, \mathbf{x}_t \rangle - \sum_{t=1}^T \langle \tilde{\mathbf{g}}_t, \mathbf{u} \rangle \right] \leq \frac{B_\psi(\mathbf{u}; \mathbf{x}_1)}{\eta} + \mathbb{E} \left[\sum_{t=1}^T \sum_{i=1}^d x_{t,i} \tilde{g}_{t,i}^2 \right].$$

Now, focusing on the terms $\mathbb{E}[x_{t,i} \tilde{g}_{t,i}^2]$, we have

$$\mathbb{E} \left[\sum_{i=1}^d x_{t,i} \tilde{g}_{t,i}^2 \right] = \mathbb{E} \left[\mathbb{E} \left[\sum_{i=1}^d x_{t,i} \tilde{g}_{t,i}^2 \middle| A_1, \dots, A_{t-1} \right] \right] = \mathbb{E} \left[\sum_{i=1}^d x_{t,i} \frac{g_{t,i}^2}{x_{t,i}} \right] \leq d L_\infty^2. \quad (10.5)$$

So, setting $\eta \propto \sqrt{\frac{\ln d}{L_\infty^2 T}}$, we have

$$\mathbb{E} \left[\sum_{t=1}^T g_{t,A_t} \right] - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq O \left(L_\infty \sqrt{dT \ln d} \right).$$

Remark 10.5. The need for a different analysis for OMD is due to the fact that we want an easy way to upper bound the Hessian. Indeed, in this analysis $\tilde{\mathbf{x}}_{t+1}$ comes before the normalization into a probability distribution, that simplifies a lot the analysis. The same idea will be used for the Tsallis entropy in the next section.

So, with a tighter analysis we showed that, even without an explicit exploration term, OMD with entropic regularizer solves the multi-armed bandit problem paying only a factor \sqrt{d} more than the full information case. However, this is still not the optimal regret!

In the next section, we will see that changing the regularizer, *with the same analysis*, will remove the $\sqrt{\ln d}$ term in the regret.

10.1.2 Optimal Regret Using OMD with Tsallis Entropy

In this section, we present the Implicitly Normalized Forecaster (INF) also known as OMD with Tsallis entropy for multi-armed bandit.

Define $\psi_q : \mathbb{R}_+^d \rightarrow \mathbb{R}$ as $\psi_q(\mathbf{x}) = \frac{1}{1-q} \left(1 - \sum_{i=1}^d x_i^q\right)$, where $q \in [0, 1]$ and in $q = 1$ we extend the function by continuity. This is the negative **Tsallis entropy** of the vector \mathbf{x} . This is a strict generalization of the Shannon entropy, because when q goes to 1, $\psi_q(\mathbf{x})$ converges to the negative (Shannon) entropy of \mathbf{x} .

We will instantiate OMD with this regularizer for the multi-armed problem, as in Algorithm 10.3.

Algorithm 10.3 INF Algorithm

Require: $\eta > 0$

- 1: $\mathbf{x}_1 = [1/d, \dots, 1/d]$
 - 2: **for** $t = 1$ **to** T **do**
 - 3: Draw A_t according to $P(A_t = i) = x_{t,i}$
 - 4: Select expert A_t
 - 5: Observe *only* the loss of the selected arm $g_{t,A_t} \in [0, L_\infty]$ and pay it
 - 6: Construct the estimate $\tilde{\mathbf{g}}_t = \begin{cases} \frac{g_{t,i}}{x_{t,i}}, & i = A_t \\ 0, & \text{otherwise} \end{cases}$
 - 7: $\tilde{\mathbf{x}}_{t+1} = \operatorname{argmin}_{\mathbf{x} \in V} \eta \langle \tilde{\mathbf{g}}_t, \mathbf{x} \rangle - \frac{1}{1-q} \sum_{i=1}^d x_i^q + \frac{q}{1-q} \sum_{i=1}^d x_{t,i}^{q-1} x_i$
 - 8: **end for**
-

Note that $\operatorname{argmin}_{\mathbf{x}} \psi_q(\mathbf{x}) = \frac{1}{d}$ and $\min_{\mathbf{x}} \psi_q(\mathbf{x}) = \frac{1-d^{1-q}}{1-q}$.

We will not use any interpretation of this regularizer from the information theory point of view. As we will see in the following, the only reason to choose it is its Hessian. In fact, the Hessian of this regularizer is still diagonal and it is equal to

$$(\nabla^2 \psi_q(\mathbf{z}))_{ii} = q \frac{1}{z_i^{2-q}}.$$

Now, we can use again the modified analysis for OMD in Lemma 10.3. So, for any $\mathbf{u} \in V$, we obtain

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq \frac{B_{\psi_q}(\mathbf{u}; \mathbf{x}_1)}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \mathbf{g}_t^\top (\nabla^2 \psi_q(\mathbf{z}_t))^{-1} \mathbf{g}_t = \frac{d^{1-q} - \sum_{i=1}^d u_i^q}{\eta(1-q)} + \frac{\eta}{2} \sum_{t=1}^T \sum_{i=1}^d g_{t,i}^2 z_{t,i}^{2-q},$$

where $\mathbf{z}_t = \alpha_t \mathbf{x}_t + (1 - \alpha_t) \tilde{\mathbf{x}}_{t+1}$ and $\alpha_t \in [0, 1]$.

As we did for Exp3, now we need an upper bounds to the $z_{t,i}$. From the update rule and the definition of ψ , we have

$$-\frac{q}{1-q} \tilde{\mathbf{x}}_{t+1,i}^{q-1} = -\frac{q}{1-q} \mathbf{x}_{t,i}^{q-1} - \eta g_{t,i},$$

that is

$$\tilde{\mathbf{x}}_{t+1,i} = \frac{\mathbf{x}_{t,i}}{\left(1 + \frac{1-q}{q} \eta g_{t,i} \mathbf{x}_{t,i}^{1-q}\right)^{\frac{1}{1-q}}}.$$

So, if $g_{t,i} \geq 0$, $\tilde{\mathbf{x}}_{t+1,i} \leq \mathbf{x}_{t,i}$, that implies that $z_{t,i} \leq \mathbf{x}_{t,i}$.

Hence, putting all together, we have

$$\sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{u} \rangle \leq \frac{d^{1-q} - \sum_{i=1}^d u_i^q}{\eta(1-q)} + \frac{\eta}{2q} \sum_{t=1}^T \sum_{i=1}^d g_{t,i}^2 \mathbf{x}_{t,i}^{2-q}.$$

We can now specialize the above reasoning, considering $q = 1/2$ in the Tsallis entropy, to obtain the following theorem.

Theorem 10.6. Assume $g_{t,i} \in [0, L_\infty]$. Set $q = 1/2$ and $\mathbf{x}_1 = [1/d, \dots, 1/d]$. Then, Algorithm 10.3

$$\mathbb{E} \left[\sum_{t=1}^T g_{t,A_t} \right] - \sum_{t=1}^T \langle \mathbf{g}_t, \mathbf{u} \rangle \leq \frac{2\sqrt{d}}{\eta} + \eta\sqrt{d}L_\infty^2 T.$$

Proof. We only need to calculate the terms

$$\mathbb{E} \left[\sum_{i=1}^d \tilde{g}_{t,i}^2 x_{t,i}^{3/2} \right].$$

Proceeding as in (10.5), we obtain

$$\begin{aligned} \mathbb{E} \left[\sum_{i=1}^d \tilde{g}_{t,i}^2 x_{t,i}^{3/2} \right] &= \mathbb{E} \left[\mathbb{E} \left[\sum_{i=1}^d x_{t,i}^{3/2} \tilde{g}_{t,i}^2 \middle| A_1, \dots, A_{t-1} \right] \right] = \mathbb{E} \left[\sum_{i=1}^d x_{t,i}^{3/2} \frac{g_{t,i}^2}{x_{t,i}^2} x_{t,i} \right] = \mathbb{E} \left[\sum_{i=1}^d g_{t,i}^2 \sqrt{x_{t,i}} \right] \\ &\leq \mathbb{E} \left[\sqrt{\sum_{i=1}^d g_{t,i}^2} \sqrt{\sum_{i=1}^d x_{t,i}} \right] \leq L_\infty \sqrt{d}. \end{aligned} \quad \square$$

Choosing $\eta \propto \frac{1}{L_\infty \sqrt{T}}$, we finally obtain an expected regret of $O(L_\infty \sqrt{dT})$, that can be proved to be the optimal one.

There is one last thing, is how do we compute the prediction of this algorithm? In each step, we have to solve a constrained optimization problem. So, we can write the corresponding Lagrangian:

$$L(\mathbf{x}, \beta) = \sum_{i=1}^d \left(\eta \tilde{g}_{t,i} + \frac{q}{1-q} x_{t,i}^{q-1} \right) x_i - \frac{1}{1-q} \sum_{i=1}^d x_i^q + \beta \left(\sum_{i=1}^d x_i - 1 \right).$$

From the KKT conditions, we have

$$x_{t+1,i} = \left[\frac{1-q}{q} \left(\beta + \frac{q}{1-q} x_{t,i}^{q-1} + \eta \tilde{g}_{t,i} \right) \right]^{\frac{1}{q-1}}.$$

and we also know that $\sum_{i=1}^d x_{t+1,i} = 1$. So, we have a 1-dimensional problem in β that must be solved in each round.

10.2 Stochastic Bandits

Today, we will consider the *stochastic bandit* setting. Here, each arm is associated with an unknown probability distribution. At each time step, the algorithm selects one arm A_t and it receives a loss (or reward) g_{t,A_t} drawn i.i.d. from the distribution of the arm A_t . We focus on minimizing the *pseudo-regret*, that is the regret with respect to the optimal action in expectation, rather than the optimal action on the sequence of realized losses:

$$\text{Regret}_T := \mathbb{E} \left[\sum_{t=1}^T g_{t,A_t} \right] - \min_{i=1, \dots, d} \mathbb{E} \left[\sum_{t=1}^T g_{t,i} \right] = \mathbb{E} \left[\sum_{t=1}^T g_{t,A_t} \right] - \min_{i=1, \dots, d} \mu_i,$$

where we denoted by μ_i the expectation of the distribution associated with the arm i .

Remark 10.7. The usual notation in the stochastic bandit literature is to consider rewards instead of losses. Instead, to keep our notation coherent with the OCO literature, we will consider losses. The two things are completely equivalent up to a multiplication by -1 .

Before presenting our first algorithm for stochastic bandits, we will introduce some basic notions on concentration inequalities that will be useful in our definitions and proofs.

10.2.1 Concentration Inequalities Bits

Suppose that X_1, X_2, \dots, X_n is a sequence of independent and identically distributed random variables and with mean $\mu = \mathbb{E}[X_1]$ and variance $\sigma^2 = \text{Var}[X_1]$. Having observed X_1, X_2, \dots, X_n we would like to estimate the common mean μ . The most natural estimator is the *empirical mean*

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n X_i .$$

Linearity of expectation shows that $\mathbb{E}[\hat{\mu}] = \mu$, which means that $\hat{\mu}$ is an *unbiased estimator* of μ . Yet, $\hat{\mu}$ is a random variable itself. So, can we quantify how far $\hat{\mu}$ will be from μ ?

We could use Chebyshev's inequality to upper bound the probability that $\hat{\mu}$ is far from μ :

$$\mathbb{P}[|\hat{\mu} - \mu| \geq \epsilon] \leq \frac{\text{Var}[\hat{\mu}]}{\epsilon^2} .$$

Using the fact that $\text{Var}[\hat{\mu}] = \frac{\sigma^2}{n}$, we have that

$$\mathbb{P}[|\hat{\mu} - \mu| \geq \epsilon] \leq \frac{\sigma^2}{n\epsilon^2} .$$

So, we can expect the probability of having a “bad” estimate to go to zero as one over the number of samples in our empirical mean. Is this the best we can get? To understand what we can hope for, let's take a look at the central limit theorem.

We know that, defining $S_n = \sum_{i=1}^n (X_i - \mu)$, $\frac{S_n}{\sqrt{n\sigma^2}} \rightarrow N(0, 1)$, the standard Gaussian distribution, as n goes to infinity. This means that

$$\mathbb{P}[\hat{\mu} - \mu \geq \epsilon] = \mathbb{P}[S_n \geq n\epsilon] = \mathbb{P}\left[\frac{S_n}{\sqrt{2\sigma^2}} \geq \sqrt{\frac{n}{\sigma^2}}\epsilon\right] \approx \int_{\epsilon\sqrt{\frac{n}{\sigma^2}}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx ,$$

where the approximation comes from the central limit theorem. The integral cannot be calculated with a closed form, but we can easily upper bound it. Indeed, for $a > 0$, we have

$$\int_a^{\infty} \exp\left(-\frac{x^2}{2}\right) dx = \int_a^{\infty} \frac{x}{x} \exp\left(-\frac{x^2}{2}\right) dx \leq \frac{1}{a} \int_a^{\infty} x \exp\left(-\frac{x^2}{2}\right) dx = \frac{1}{a} \exp\left(-\frac{a^2}{2}\right) .$$

Hence, we have

$$\mathbb{P}[\hat{\mu} - \mu \geq \epsilon] \lesssim \sqrt{\frac{\sigma^2}{2\pi\epsilon^2 n}} \exp\left(-\frac{n\epsilon^2}{2\sigma^2}\right) . \quad (10.6)$$

This is better than what we got with Chebyshev's inequality and we would like to obtain an exact bound with a similar asymptotic rate. To do that, we will focus our attention on *subgaussian* random variables.

Definition 10.8 (Subgaussian Random Variable). *We say that a random variable is σ -subgaussian if for all $\lambda \in \mathbb{R}$ we have that $\mathbb{E}[\exp(\lambda X)] \leq \exp(\lambda^2 \sigma^2 / 2)$.*

Example 10.9. *The following random variable are subgaussian:*

- If X is Gaussian with mean zero and variance σ^2 , then X is σ -subgaussian.
- If X has mean zero and $X \in [a, b]$ almost surely, then X is $(b - a)/2$ -subgaussian.

We have the following properties for subgaussian random variables.

Lemma 10.10 ([Lattimore and Szepesvári, 2018, Lemma 5.4]). *Assume that X_1 and X_2 are independent and σ_1 -subgaussian and σ_2 -subgaussian respectively. Then,*

- (a) $\mathbb{E}[X_1] = 0$ and $\text{Var}[X_1] \leq \sigma_1^2$.
- (b) cX_1 is $|c|\sigma_1$ -subgaussian.
- (c) $X_1 + X_2$ is $\sqrt{\sigma_1^2 + \sigma_2^2}$ -subgaussian.

Subgaussian random variables behave like Gaussian random variables, in the sense that their tail probabilities are upper bounded by the ones of a Gaussian of variance σ^2 . To prove it, let's first state the Markov's inequality.

Theorem 10.11 (Markov's inequality). *For a non-negative random variable X and $\epsilon > 0$, we have that $\mathbb{P}[X \geq \epsilon] \leq \frac{\mathbb{E}[X]}{\epsilon}$.*

With Markov's inequality, we can now formalize the above statement on subgaussian random variables.

Theorem 10.12. *If a random variable is σ -subgaussian, then $\mathbb{P}[X \geq \epsilon] \leq \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$.*

Proof. For any $\lambda > 0$, we have

$$\mathbb{P}[X \geq \epsilon] = \mathbb{P}[\exp(\lambda X) \geq \exp(\lambda \epsilon)] \leq \frac{\mathbb{E}[\exp(\lambda X)]}{\exp(\lambda \epsilon)} \leq \exp(\lambda^2 \sigma^2 / 2 - \lambda \epsilon).$$

Minimizing the right hand side of the inequality w.r.t. λ , we have the stated result. \square

An easy consequence of the above theorem is that the empirical average of subgaussian random variables concentrates around its expectation, *with the same asymptotic rate in (10.6)*.

Corollary 10.13. *Assume that $X_i - \mu$ are independent, σ -subgaussian random variables. Then, for any $\epsilon \geq 0$, we have*

$$\mathbb{P}[\hat{\mu} \geq \mu + \epsilon] \leq \exp\left(-\frac{n\epsilon^2}{2\sigma^2}\right) \quad \text{and} \quad \mathbb{P}[\hat{\mu} \leq \mu - \epsilon] \leq \exp\left(-\frac{n\epsilon^2}{2\sigma^2}\right),$$

where $\hat{\mu} = \frac{1}{n} \sum_{i=1}^n X_i$.

Equating the upper bounds on the r.h.s. of the inequalities in the Corollary to δ , we have the equivalent statement that, with probability at least $1 - 2\delta$, we have

$$\mu \in \left[\hat{\mu} - \sqrt{\frac{2\sigma^2 \ln \frac{1}{\delta}}{n}}, \hat{\mu} + \sqrt{\frac{2\sigma^2 \ln \frac{1}{\delta}}{n}} \right].$$

10.2.2 Explore-Then-Commit Algorithm

Algorithm 10.4 Explore-Then-Commit Algorithm

Require: $T, m \in \mathbb{N}, 1 \leq m \leq \frac{T}{d}$

1: $S_{0,i} = 0, \hat{\mu}_{0,i} = 0, i = 1, \dots, d$

2: **for** $t = 1$ **to** T **do**

3: Choose $A_t = \begin{cases} (t \bmod d) + 1, & t \leq dm \\ \arg\min_i \hat{\mu}_{dm,i}, & t > dm \end{cases}$

4: Observe g_{t,A_t} and pay it

5: $S_{t,i} = S_{t-1,i} + \mathbf{1}[A_t = i]$

6: $\hat{\mu}_{t,i} = \frac{1}{S_{t,i}} \sum_{j=1}^t g_{j,A_j} \mathbf{1}[A_j = i], i = 1, \dots, d$

7: **end for**

We are now ready to present the most natural algorithm for the stochastic bandit setting, called Explore-Then-Commit (ETC) algorithm. That is, we first identify the best arm over md exploration rounds and then we commit to it. This algorithm is summarized in Algorithm 10.4.

In the following, we will denote by $S_{t,i} = \sum_{j=1}^t \mathbf{1}[A_t = i]$, that is the number of times that the arm i was pulled in the first t rounds.

Define by μ^* the expected loss of the arm with the smallest expectation, that is $\min_{i=1,\dots,d} \mu_i$. Critical quantities in our analysis will be the *gaps*, $\Delta_i := \mu_i - \mu^*$ for $i = 1, \dots, d$, that measure the expected difference in losses between the arms and the optimal one. In particular, we can decompose the regret as a sum over the arms of the expected number of times we pull an arm multiplied by its gap.

Lemma 10.14. *For any policy of selection of the arms, the regret is upper bounded by*

$$\text{Regret}_T = \sum_{i=1}^d \mathbb{E}[S_{T,i}] \Delta_i.$$

Proof. Observe that

$$\sum_{t=1}^T g_{t,A_t} = \sum_{t=1}^T \sum_{i=1}^d g_{t,i} \mathbf{1}[A_t = i].$$

Hence,

$$\begin{aligned} \text{Regret}_T &= \mathbb{E} \left[\sum_{t=1}^T g_{t,A_t} \right] - T\mu^* = \mathbb{E} \left[\sum_{t=1}^T (g_{t,A_t} - \mu^*) \right] = \sum_{i=1}^d \sum_{t=1}^T \mathbb{E}[\mathbf{1}[A_t = i] (g_{t,i} - \mu^*)] \\ &= \sum_{i=1}^d \sum_{t=1}^T \mathbb{E}[\mathbb{E}[\mathbf{1}[A_t = i] (g_{t,i} - \mu^*) | A_t]] = \sum_{i=1}^d \sum_{t=1}^T \mathbb{E}[\mathbf{1}[A_t = i] \mathbb{E}[g_{t,i} - \mu^* | A_t]] \\ &= \sum_{i=1}^d \sum_{t=1}^T \mathbb{E}[\mathbf{1}[A_t = i] (\mu_{A_t} - \mu^*)] = \sum_{i=1}^d \sum_{t=1}^T \mathbb{E}[\mathbf{1}[A_t = i] (\mu_i - \mu^*)]. \quad \square \end{aligned}$$

The above Lemma quantifies the intuition that in order to have a small regret we have to select the suboptimal arms less often than the best one.

We are now ready to prove the regret guarantee of the ETC algorithm.

Theorem 10.15. *Assume that the losses of the arms minus their expectations are 1-subgaussian and $1 \leq m \leq T/d$. Then, ETC guarantees a regret of*

$$\text{Regret}_T \leq m \sum_{i=1}^d \Delta_i + (T - md) \sum_{i=1}^d \Delta_i \exp \left(-\frac{m\Delta_i^2}{4} \right).$$

Proof. Let's assume without loss of generality that the optimal arm is the first one.

So, for $i \neq 1$, we have

$$\begin{aligned} \sum_{t=1}^T \mathbb{E}[\mathbf{1}[A_t = i]] &= m + (T - md) \mathbb{P} \left[\hat{\mu}_{md,i} \leq \min_{j \geq i} \hat{\mu}_{md,j} \right] \\ &\leq m + (T - md) \mathbb{P} [\hat{\mu}_{md,i} \leq \hat{\mu}_{md,1}] \\ &= m + (T - md) \mathbb{P} [\hat{\mu}_{md,1} - \mu_1 - (\hat{\mu}_{md,i} - \mu_i) \geq \Delta_i]. \end{aligned}$$

From Lemma 10.10, we have that $\hat{\mu}_{md,i} - \mu_i - (\hat{\mu}_{md,1} - \mu_1)$ is $\sqrt{2/m}$ -subgaussian. So, from Theorem 10.12, we have

$$\mathbb{P} [\hat{\mu}_{md,1} - \mu_1 - (\hat{\mu}_{md,i} - \mu_i) \geq \Delta_i] \leq \exp \left(-\frac{m\Delta_i^2}{4} \right). \quad \square$$

The bound shows the trade-off between exploration and exploitation: if m is too big, we pay too much during the exploration phase (first term in the bound). On the other hand, if m is small, the probability to select a suboptimal arm increases (second term in the bound). Knowing all the gaps Δ_i , it is possible to choose m that minimizes the bound.

For example, in that case that $d = 2$, the regret is upper bounded by

$$m\Delta + (T - 2m)\Delta \exp\left(-m\frac{\Delta^2}{4}\right) \leq m\Delta + T\Delta \exp\left(-m\frac{\Delta^2}{4}\right),$$

that is minimized by

$$m = \frac{4}{\Delta^2} \ln \frac{T\Delta^2}{4}.$$

Remembering that m must be a natural number we can choose

$$m = \max\left(\left\lceil \frac{4}{\Delta^2} \ln \frac{T\Delta^2}{4} \right\rceil, 1\right).$$

When $\frac{T\Delta^2}{4} \leq 1$, we select $m = 1$. So, we have $\Delta + (T - 2)\Delta \leq T\Delta \leq \frac{4}{\Delta}$. Hence, the regret is upper bounded by

$$\min\left(\Delta T, \frac{4}{\Delta} \left(1 + \max\left(\ln \frac{T\Delta^2}{4}, 0\right)\right) + \Delta\right).$$

The main drawback of this algorithm is that its optimal tuning depends on the gaps δ_i . Assuming the knowledge of the gaps account to make the stochastic bandit problem completely trivial. However, its tuned regret bound gives us a baseline to which compare other bandit algorithms. In particular, in the next section we will present an algorithm that achieves the same asymptotic regret without any knowledge of the gaps.

10.2.3 Upper Confidence Bound Algorithm

Algorithm 10.5 Upper Confidence Bound Algorithm

Require: $\alpha > 2, T \in \mathbb{N}$

- 1: $S_{0,i} = 0, \hat{\mu}_{0,i} = 0, i = 1, \dots, d$
 - 2: **for** $t = 1$ **to** T **do**
 - 3: Choose $A_t = \operatorname{argmin}_{i=1,\dots,d} \begin{cases} \mu_{t-1,i} - \sqrt{\frac{2\alpha \ln t}{S_{t-1,i}}}, & \text{if } S_{t-1,i} \neq 0 \\ -\infty, & \text{otherwise} \end{cases}$
 - 4: Observe g_{t,A_t} and pay it
 - 5: $S_{t,i} = S_{t-1,i} + \mathbf{1}[A_t = i]$
 - 6: $\hat{\mu}_{t,i} = \frac{1}{S_{t,i}} \sum_{j=1}^t g_{j,A_t} \mathbf{1}[A_j = i], i = 1, \dots, d$
 - 7: **end for**
-

The ETC algorithm has the disadvantage of requiring the knowledge of the gaps to tune the exploration phase. Moreover, it solves the exploration vs. exploitation trade-off in a clunky way. It would be better to have an algorithm that smoothly transition from one phase into the other *in a data-dependent way*. So, we now describe an optimal and adaptive strategy called Upper Confidence Bound (UCB) algorithm. It employs the principle of *optimism in the face of uncertainty*, to select in each round the arm that has the *potential to be the best one*.

UCB works keeping an estimate of the expected loss of each arm and also a confidence interval at a certain probability. Roughly speaking, we have that with probability at least $1 - \delta$

$$\mu_i \in \left[\hat{\mu}_i - \sqrt{\frac{2 \ln \frac{1}{\delta}}{S_{t-1,i}}}, \hat{\mu}_i \right],$$

where the “roughly” comes from the fact that $S_{t-1,i}$ is a random variable itself. Then, UCB will query the arm with the smallest lower bound, that is the one that could potentially have the smallest expected loss.

Remark 10.16. The name *Upper Confidence Bound* comes from the fact that traditionally stochastic bandits are defined over rewards, rather than losses. So, in our case we actually use the lower confidence bound in the algorithm. However, to avoid confusion with the literature, we still call it *Upper Confidence Bound algorithm*.

The key points in the proof are on how to choose the right confidence level δ and how to get around the dependency issues.

The algorithm is summarized in Algorithm 10.5 and we can prove the following regret bound.

Theorem 10.17. Assume that the rewards of the arms are 1-subgaussian and $\alpha > 2$ and let $\alpha > 2$. Then, UCB guarantees a regret of

$$\text{Regret}_T \leq \frac{\alpha}{\alpha - 2} \sum_{i=1}^d \Delta_i + \sum_{i: \Delta_i > 0} \frac{8\alpha \ln T}{\Delta_i}.$$

Proof. We analyze one arm at the time. Also, without loss of generality, assume that the optimal arm is the first one. For arm i , we want to prove that $\mathbb{E}[S_{T,i}] \leq \frac{8\alpha \ln T}{\Delta_i^2} + \frac{\alpha}{\alpha - 2}$.

The proof is based on the fact that once I have sampled an arm enough times, the probability to take a suboptimal arm is small.

Let t^* the biggest time index such that $S_{t^*-1,i} \leq \frac{8\alpha \ln T}{\Delta_i^2}$. If $t^* = T$, then the statement above is true. Hence, we can safely assume $t^* < T$. Now, for $t > t^*$, we have

$$S_{t-1,i} > \frac{8\alpha \ln T}{\Delta_i^2}. \quad (10.7)$$

Consider $t > t^*$ and such that $A_t = i$, then we claim that at least one of the two following equations must be true:

$$\hat{\mu}_{t-1,1} - \sqrt{\frac{2\alpha \ln t}{S_{t-1,1}}} \geq \mu_1, \quad (10.8)$$

$$\hat{\mu}_{t-1,i} + \sqrt{\frac{2\alpha \ln t}{S_{t-1,i}}} < \mu_i. \quad (10.9)$$

If the first one is true, the confidence interval around our estimate of the expectation of the optimal arm does not contain μ_1 . On the other hand, if the second one is true the confidence interval around our estimate of the expectation μ_i does not contain μ_i . So, we claim that if $t > t^*$ and we selected a suboptimal arm, then at least one of these two bad events happened.

Let's prove the claim: if both the inequalities above are false, $t > t^*$, and $A_t = i$, we have

$$\begin{aligned} \hat{\mu}_{t-1,1} - \sqrt{\frac{2\alpha \ln t}{S_{t-1,1}}} &< \mu_1 \quad ((10.8) \text{ false}) \\ &= \mu_i - \Delta_i \\ &< \mu_i - 2\sqrt{\frac{2\alpha \ln T}{S_{t-1,i}}} \quad (\text{for (10.7)}) \\ &\leq \mu_i - 2\sqrt{\frac{2\alpha \ln t}{S_{t-1,i}}} \\ &\leq \hat{\mu}_{t-1,i} - \sqrt{\frac{2\alpha \ln t}{S_{t-1,i}}} \quad ((10.9) \text{ false}), \end{aligned}$$

that, by the selection strategy of the algorithm, would imply $A_t \neq i$.

Note that $S_{t^*,i} \leq \frac{8\alpha \ln T}{\Delta_i^2} + 1$. Hence, we have

$$\begin{aligned} \mathbb{E}[S_{T,i}] &= \mathbb{E}[S_{t^*,i}] + \sum_{t=t^*+1}^T \mathbb{E}[\mathbf{1}[A_t = i, (10.8) \text{ or } (10.9) \text{ true}]] \\ &\leq \frac{8\alpha \ln T}{\Delta_i^2} + 1 + \sum_{t=t^*+1}^T \mathbb{E}[\mathbf{1}[(10.8) \text{ or } (10.9) \text{ true}]] \\ &\leq \frac{8\alpha \ln T}{\Delta_i^2} + 1 + \sum_{t=t^*+1}^T (\mathbb{P}[(10.8) \text{ true}] + \mathbb{P}[(10.9) \text{ true}]) . \end{aligned}$$

Now, we upper bound the probabilities in the sum. First, note that, given that the losses on the arms are i.i.d., we have

$$\left\{ \hat{\mu}_{t-1,1} - \sqrt{\frac{2\alpha \ln t}{S_{t-1,1}}} \geq \mu_1 \right\} \subset \left\{ \max_{s=1, \dots, t-1} \frac{1}{s} \sum_{j=1}^s g_{j,1} - \sqrt{\frac{2\alpha \ln t}{s}} \geq \mu_1 \right\} = \bigcup_{s=1}^{t-1} \left\{ \frac{1}{s} \sum_{j=1}^s g_{j,1} - \sqrt{\frac{2\alpha \ln t}{s}} \geq \mu_1 \right\}$$

Hence, we have

$$\begin{aligned} \mathbb{P}[(10.8) \text{ true}] &\leq \sum_{s=1}^{t-1} \mathbb{P} \left[\frac{1}{s} \sum_{j=1}^s g_{j,1} - \sqrt{\frac{2\alpha \ln t}{s}} \geq \mu_1 \right] \quad (\text{union bound}) \\ &\leq \sum_{s=1}^{t-1} t^{-\alpha} = (t-1)t^{-\alpha} . \end{aligned}$$

Given that the same bound holds for $\mathbb{P}[(10.9) \text{ true}]$, we have

$$\begin{aligned} \mathbb{E}[S_{T,i}] &\leq \frac{8\alpha \ln T}{\Delta_i^2} + 1 + \sum_{t=1}^{\infty} 2(t-1)t^{-\alpha} \\ &\leq \frac{8\alpha \ln T}{\Delta_i^2} + 1 + \sum_{t=2}^{\infty} 2t^{1-\alpha} \\ &\leq \frac{8\alpha \ln T}{\Delta_i^2} + 1 + 2 \int_1^{\infty} x^{1-\alpha} \\ &= \frac{8\alpha \ln T}{\Delta_i^2} + \frac{\alpha}{\alpha-2} . \end{aligned}$$

Using the decomposition of the regret we proved last time, $\text{Regret}_T = \sum_{i=1}^d \Delta_i \mathbb{E}[S_{T,i}]$, we have the stated bound. \square

The bound above can become meaningless if the gaps are too small. So, here we prove another bound that does not depend on the inverse of the gaps.

Theorem 10.18. *Assume that the rewards of the arms minus their expectations are 1-subgaussian and let $\alpha > 2$. Then, UCB guarantees a regret of*

$$\text{Regret}_T \leq 4\sqrt{2\alpha d T \ln T} + \frac{\alpha}{\alpha-2} \sum_{i=1}^d \Delta_i .$$

Proof. Let $\Delta > 0$ be some value to be tuned subsequently and recall from the proof of Theorem 10.17 that for each suboptimal arm i we can bound

$$\mathbb{E}[S_{T,i}] \leq \frac{\alpha}{\alpha-2} + \frac{8\alpha \ln T}{\Delta_i^2} .$$

Hence, using the regret decomposition we proved last time, we have

$$\begin{aligned}
\text{Regret}_T &= \sum_{i: \Delta_i < \Delta} \Delta_i \mathbb{E}[S_{T,i}] + \sum_{i: \Delta_i \geq \Delta} \Delta_i \mathbb{E}[S_{T,i}] \\
&\leq T\Delta + \sum_{i: \Delta_i \geq \Delta} \Delta_i \mathbb{E}[S_{T,i}] \\
&\leq T\Delta + \sum_{i: \Delta_i \geq \Delta} \left(\Delta_i \frac{\alpha}{\alpha - 2} + \frac{8\alpha \ln T}{\Delta_i} \right) \\
&\leq T\Delta + \frac{\alpha}{\alpha - 2} \sum_{i=1}^d \Delta_i + \frac{8\alpha d \ln T}{\Delta}.
\end{aligned}$$

Choosing $\Delta = \sqrt{\frac{8\alpha d \ln T}{T}}$, we have the stated bound. \square

Remark 10.19. Note that while the UCB algorithm is considered parameter-free, we still have to know the subgaussianity of the arms. While this can be easily upper bounded for stochastic arms with bounded support, it is unclear how to do it without any prior knowledge on the distribution of the arms.

It is possible to prove that the UCB algorithm is asymptotically optimal, in the sense of the following Theorem.

Theorem 10.20 ([Bubeck and Cesa-Bianchi, 2012, Theorem 2.2]). Consider a strategy that satisfies $\mathbb{E}[S_{T,i}] = o(T^a)$ for any set of Bernoulli rewards distributions, any arm i with $\Delta_i > 0$ and any $a > 0$. Then, for any set of Bernoulli reward distributions, the following holds

$$\liminf_{T \rightarrow +\infty} \frac{\text{Regret}_T}{\ln T} \geq \sum_{i: \Delta_i} \frac{1}{2\Delta_i}.$$

10.3 History Bits

The algorithm in Algorithm 10.1 is from Cesa-Bianchi and Lugosi [2006, Theorem 6.9]. The Exp3 algorithm was proposed in Auer et al. [2002b].

The INF algorithm was proposed by Audibert and Bubeck [2009] and re-casted as an OMD procedure in Audibert et al. [2011]. The connection with the Tsallis entropy was done in Abernethy et al. [2015]. The specific proof presented here is new and it builds on the proof by Abernethy et al. [2015]. Note that Abernethy et al. [2015] proved the same regret bound for a FTRL procedure over the stochastic estimates of the losses (that they call Gradient-Based Prediction Algorithm), while here we proved it using a OMD procedure.

The ETC algorithm goes back to Robbins [1952], even if Robbins proposed what is now called epoch-greedy [Langford and Zhang, 2008]. For more history on ETC, take a look at chapter 6 in Lattimore and Szepesvári [2018]. The proofs presented here are from Lattimore and Szepesvári [2018] as well.

The use of confidence bounds and the idea of optimism first appeared in the work by Lai and Robbins [1985]. The first version of UCB is by Lai [1987]. The version of UCB I presented is by Auer et al. [2002a] under the name UCB1. Note that, rather than considering 1-subgaussian environments, Auer et al. [2002a] considers bandits where the rewards are confined to the $[0, 1]$ interval. The proof of Theorem 10.17 is a minor variation of the one of Theorem 2.1 in Bubeck and Cesa-Bianchi [2012], which also popularized the subgaussian setup. Theorem 10.18 is from Bubeck and Cesa-Bianchi [2012].

10.4 Exercises

Problem 10.1. Prove that in the modified proof of OMD, the terms $B_\psi(\mathbf{x}_t; \tilde{\mathbf{x}}_{t+1})$ can be upper bounded by $\langle \eta \mathbf{g}_t, \mathbf{x}_t - \tilde{\mathbf{x}}_{t+1} \rangle$.

Problem 10.2. *Building on the previous exercise, prove that regret bounds of the same order can be obtained for Exp3 and for the Poly-INF/OMD with Tsallis entropy directly upper bounding the terms $\langle \eta \mathbf{g}_t, \mathbf{x}_t - \tilde{\mathbf{x}}_{t+1} \rangle$, without passing through the Bregman divergences.*

Problem 10.3. *Prove a similar regret bound to the one in Theorem 10.18 for an optimally tuned Explore-Then-Commit algorithm.*

Appendix A

Appendix

A.1 The Lambert Function and Its Applications

The Lambert function $W(x) : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is defined by the equality

$$x = W(x) \exp(W(x)) \quad \text{for } x \geq 0. \quad (\text{A.1})$$

Hence, from the definition we have that $\exp(W(x)/2) = \sqrt{\frac{x}{W(x)}}$.

Theorem A.1 ([Hoorfar and Hassani, 2008, Theorem 2.3]).

$$W(x) \leq \ln \frac{x + C}{1 + \ln(C)}, \quad \forall x > -\frac{1}{e}, \quad C > \frac{1}{e}.$$

The following lemma provides upper and lower bounds on $W(x)$.

Theorem A.2. *The Lambert function $W(x)$ satisfies*

$$0.6321 \ln(x + 1) \leq W(x) \leq \ln(x + 1), \quad \forall x \geq 0.$$

Proof. The inequalities are satisfied for $x = 0$, hence we in the following we assume $x > 0$. We first prove the lower bound. From (A.1) we have

$$W(x) = \ln \left(\frac{x}{W(x)} \right). \quad (\text{A.2})$$

From this equality, using the elementary inequality $\ln(x) \leq \frac{a}{e} x^{\frac{1}{a}}$ for any $a > 0$, we get

$$W(x) \leq \frac{1}{a e} \left(\frac{x}{W(x)} \right)^a \quad \forall a > 0,$$

that is

$$W(x) \leq \left(\frac{1}{a e} \right)^{\frac{1}{1+a}} x^{\frac{a}{1+a}} \quad \forall a > 0. \quad (\text{A.3})$$

Using (A.3) in (A.2), we have

$$W(x) \geq \ln \left(\frac{x}{\left(\frac{1}{a e} \right)^{\frac{1}{1+a}} x^{\frac{a}{1+a}}} \right) = \frac{1}{1+a} \ln(a e x) \quad \forall a > 0.$$

Consider now the function $g(x) = \frac{x}{x+1} - \frac{b}{\ln(1+b)(b+1)} \ln(x+1)$ defined in $[0, b]$ where b is a positive number that will be decided in the following. This function has a maximum in $x^* = (1 + \frac{1}{b}) \ln(1+b) - 1$, the derivative is positive in

$[0, x^*]$ and negative in $[x^*, b]$. Hence the minimum is in $x = 0$ and in $x = b$, where it is equal to 0. Using the property just proved on g , setting $a = \frac{1}{x}$, we have

$$W(x) \geq \frac{x}{x+1} \geq \frac{b}{\ln(1+b)(b+1)} \ln(x+1) \quad \forall x \leq b.$$

For $x > b$, setting $a = \frac{x+1}{ex}$, we have

$$W(x) \geq \frac{ex}{(e+1)x+1} \ln(x+1) \geq \frac{eb}{(e+1)b+1} \ln(x+1) \quad (\text{A.4})$$

Hence, we set b such that

$$\frac{eb}{(e+1)b+1} = \frac{b}{\ln(1+b)(b+1)}$$

Numerically, $b = 1.71825\dots$, so

$$W(x) \geq 0.6321 \ln(x+1).$$

For the upper bound, we use Theorem A.1 and set $C = 1$. □

Theorem A.3. Let $a, b > 0$. Then, the Fenchel conjugate of $f(x) = b \exp(x^2/(2a))$ is

$$f^*(\theta) = \sqrt{a}|\theta| \sqrt{W(a\theta^2/b^2)} - b \exp\left(\frac{W(a\theta^2/b^2)}{2}\right) = \sqrt{a}|\theta| \left(\sqrt{W(a\theta^2/b^2)} - \frac{1}{\sqrt{W(a\theta^2/b^2)}} \right).$$

Moreover,

$$f^*(\theta) \leq \sqrt{a}|\theta| \sqrt{\ln(a\theta^2/b^2 + 1)} - b.$$

Proof. First, observe that

$$\max_x \theta x - b \exp(x^2/(2a)) = \max_y b \left(\frac{\sqrt{a}\theta}{b} y - \exp(y^2/2) \right).$$

Also, by the definition of Lambert function, we have

$$\operatorname{argmax}_y uy - \exp(y^2/2) = \operatorname{sign}(u) \sqrt{W(y^2)},$$

where $\operatorname{sign}(u) = 0$ for $u = 0$. Hence, $\operatorname{argmax}_x \theta x - b \exp(x^2/(2a)) = \operatorname{sign}(\theta) \sqrt{W(a\theta^2/b^2)}$. So,

$$\begin{aligned} \max_x \theta x - b \exp(x^2/(2a)) &= \max_y b \left(\frac{\sqrt{a}\theta}{b} y - \exp(y^2/2) \right) \\ &= \sqrt{a}|\theta| \sqrt{W(a\theta^2/b^2)} - b \exp\left(\frac{W(a\theta^2/b^2)}{2}\right) \\ &= \sqrt{a}|\theta| \sqrt{W(a\theta^2/b^2)} - b \sqrt{\frac{a\theta^2/b^2}{W(a\theta^2/b^2)}} \\ &= \sqrt{a}|\theta| \left(\sqrt{W(a\theta^2/b^2)} - \frac{1}{\sqrt{W(a\theta^2/b^2)}} \right). \end{aligned} \quad \square$$

The inequality is obtained through Theorem A.2.

Bibliography

- J. Abernethy, C. Lee, A. Sinha, and A. Tewari. Online linear optimization via smoothing. In *Conference on Learning Theory (COLT)*, pages 807–823, 2014. URL <http://proceedings.mlr.press/v35/abernethy14>. 72
- J. D. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In Rocco A. Servedio and Tong Zhang, editors, *Proc. of Conference on Learning Theory (COLT)*, pages 263–274. Omnipress, 2008. URL https://repository.upenn.edu/cgi/viewcontent.cgi?article=1492&context=statistics_papers. 51, 72
- J. D. Abernethy, C. Lee, and A. Tewari. Fighting bandits with a new kind of smoothness. In *Advances in Neural Information Processing Systems 28*, pages 2197–2205. Curran Associates, Inc., 2015. URL <http://papers.nips.cc/paper/6030-fighting-bandits-with-a-new-kind-of-smoothness.pdf>. 108
- R. Arora, O. Dekel, and A. Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proc. of the International Conference on Machine Learning (ICML)*, January 2012. URL <https://www.microsoft.com/en-us/research/publication/online-bandit-learning-adaptive-adversary-regret-policy-regret/>. 94
- J.-Y. Audibert and S. Bubeck. Minimax policies for adversarial and stochastic bandits. In *Proc. of the Conference on Learning Theory (COLT)*, 2009. URL <https://www.di.ens.fr/willow/pdfscurrent/COLT09a.pdf>. 108
- J.-Y. Audibert, S. Bubeck, and G. Lugosi. Minimax policies for combinatorial prediction games. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 107–132, 2011. URL <http://proceedings.mlr.press/v19/audibert11a.html>. 108
- P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002a. URL <https://homes.di.unimi.it/cesa-bianchi/Pubblicazioni/ml-02.pdf>. 108
- P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002b. URL <http://rob.schapire.net/papers/AuerCeFrSc01.pdf>. 108
- P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *J. Comput. Syst. Sci.*, 64(1):48–75, 2002c. URL <http://homes.dsi.unimi.it/~cesabian/Pubblicazioni/jcss-02.pdf>. 30
- K. S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001. URL <https://link.springer.com/article/10.1023/A:1010896012157>. 73
- H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011. URL <https://link.springer.com/book/10.1007/978-1-4419-9467-7>. 12, 33, 34, 40, 46

- A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003. URL <https://web.iem.technion.ac.il/images/user-files/becka/papers/3.pdf>. 50, 51
- A. Ben-Tal, T. Margalit, and A. Nemirovski. The ordered subsets mirror descent optimization method with applications to tomography. *SIAM Journal on Optimization*, 12(1):79–108, 2001. URL <https://epubs.siam.org/doi/pdf/10.1137/S1052623499354564>. 51
- A. Beygelzimer, F. Orabona, and C. Zhang. Efficient online bandit multiclass learning with $\tilde{O}(\sqrt{T})$ regret. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 488–497. JMLR. org, 2017. URL <https://arxiv.org/abs/1702.07958>. 78
- A. Blum, A. Kalai, and J. Langford. Beating the hold-out: Bounds for k -fold and progressive cross-validation. In *COLT*, volume 99, pages 203–208, 1999. URL https://www.ri.cmu.edu/pub_files/pub1/blum_a_1999_1/blum_a_1999_1.pdf. 22
- S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787. URL <https://web.stanford.edu/~boyd/cvxbook/>. 7
- S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012. URL <http://sbubeck.com/SurveyBCB12.pdf>. 108
- N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006. URL <https://www.cambridge.org/us/academic/subjects/computer-science/pattern-recognition-and-machine-learning/prediction-learning-and-games?format=HB&isbn=9780521841085>. 35, 50, 51, 66, 80, 108
- N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. Inf. Theory*, 50(9):2050–2057, 2004. URL <https://homes.di.unimi.it/~cesabian/Pubblicazioni/J20.pdf>. 22
- K. Chaudhuri, Y. Freund, and D. J. Hsu. A parameter-free hedging algorithm. In *Advances in neural information processing systems*, pages 297–305, 2009. URL <https://arxiv.org/abs/0903.2851>. 92
- G. Chen and M. Teboulle. Convergence analysis of a proximal-like minimization algorithm using bregman functions. *SIAM Journal on Optimization*, 3(3):538–543, 1993. URL <https://epubs.siam.org/doi/pdf/10.1137/0803026>. 41
- A. Chernov and V. Vovk. Prediction with advice of unknown number of experts. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010. URL <https://arxiv.org/abs/1408.2040>. 92
- C.-K. Chiang, T. Yang, C.-J. Lee, M. Mahdavi, C.-J. Lu, R. Jin, and S. Zhu. Online optimization with gradual variations. In *Proc. of the Conference on Learning Theory (COLT)*, volume 23, pages 6.1–6.20, 2012. URL <http://proceedings.mlr.press/v23/chiang12.html>. 73
- F. Cucker and D. X. Zhou. *Learning Theory: An Approximation Theory Viewpoint*. Cambridge University Press, New York, NY, USA, 2007. 77
- A. Cutkosky. *Algorithms and Lower Bounds for Parameter-free Online Learning*. PhD thesis, Stanford University, 2018. URL <https://www-cs.stanford.edu/people/ashokc/papers/thesis.pdf>. 36
- A. Cutkosky. Anytime online-to-batch, optimism and acceleration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proc. of the 36th International Conference on Machine Learning*, volume 97 of *Proc. of Machine Learning Research*, pages 1446–1454, Long Beach, California, USA, 09–15 Jun 2019a. PMLR. URL <http://proceedings.mlr.press/v97/cutkosky19a/cutkosky19a.pdf>. 22

- A. Cutkosky. Combining online learning guarantees. In *Proc. of the Conference on Learning Theory (COLT)*, 2019b. URL <https://arxiv.org/abs/1902.09003>. 92
- A. Cutkosky and F. Orabona. Black-box reductions for parameter-free online learning in Banach spaces. In *Proc. of the Conference on Learning Theory (COLT)*, 2018. URL <https://arxiv.org/abs/1802.06293>. 92
- S. de Rooij, T. van Erven, P. D. Grünwald, and W. M. Koolen. Follow the leader if you can, hedge if you must. *Journal of Machine Learning Research*, 15(37):1281–1316, 2014. URL <http://jmlr.org/papers/v15/rooij14a.html>. 73
- J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT*, 2010. URL https://stanford.edu/~jduchi/projects/DuchiHaSi10_colt.pdf. 28, 30
- Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. URL <https://www.sciencedirect.com/science/article/pii/S002200009791504X>. 51
- Y. Freund and R. E. Schapire. Large margin classification using the Perceptron algorithm. *Machine Learning*, pages 277–296, 1999. URL <https://cseweb.ucsd.edu/~yfreund/papers/LargeMarginsUsingPerceptron.pdf>. 78
- C. Gentile. The robustness of the p -norm algorithms. *Machine Learning*, 53(3):265–299, 2003. URL <https://link.springer.com/article/10.1023/A:1026319107706>. 78
- C. Gentile and N. Littlestone. The robustness of the p -norm algorithms. In *Proc. of the Twelfth Annual Conference on Computational Learning Theory, COLT '99*, pages 1–11, New York, NY, USA, 1999. ACM. URL <http://doi.acm.org/10.1145/307400.307405>. 51
- A. J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. In *Proc. of the 10th Annual Conference on Computational Learning Theory*, pages 171–183, 1997. 51
- E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *International Conference on Computational Learning Theory*, pages 499–513. Springer, 2006. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.3483&rep=rep1&type=pdf>. 30, 73
- E. Hazan, A. Rakhlin, and P. L. Bartlett. Adaptive online gradient descent. In *Advances in Neural Information Processing Systems*, pages 65–72, 2008. URL <https://papers.nips.cc/paper/3319-adaptive-online-gradient-descent.pdf>. 30
- A. Hoorfar and M. Hassani. Inequalities on the lambert w function and hyperpower function. *J. Inequal. Pure and Appl. Math*, 9(2):5–9, 2008. URL http://emis.ams.org/journals/JIPAM/images/107_07_JIPAM/107_07_www.pdf. 110
- P. Joulani, A. György, and C. Szepesvári. A modular analysis of adaptive (non-)convex optimization: Optimism, composite objectives, and variational bounds. In *Proc. of the International Conference on Algorithmic Learning Theory (ALT)*, volume 76, pages 681–720, 2017. URL <http://proceedings.mlr.press/v76/joulani17a.html>. 73
- J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, January 1997. URL <https://users.soe.ucsc.edu/~manfred/pubs/J36.pdf>. 51
- J. Kivinen and M. K. Warmuth. Averaging expert predictions. In *European Conference on Computational Learning Theory*, pages 153–167. Springer, 1999. URL <https://users.soe.ucsc.edu/~manfred/pubs/C50.pdf>. 73

- W. M. Koolen and T. van Erven. Second-order quantile methods for experts and combinatorial games. In *Proc. of the Conference On Learning Theory (COLT)*, pages 1155–1175, 2015. URL <https://arxiv.org/abs/1502.08009>. 93
- R. Krichevsky and V. Trofimov. The performance of universal encoding. *IEEE Trans. on Information Theory*, 27(2): 199–207, 1981. URL <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1056331>. 92
- S. Lacoste-Julien, M. Schmidt, and F. Bach. A simpler approach to obtaining an $O(1/t)$ convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012. URL <https://arxiv.org/abs/1212.2002>. 22
- T. L. Lai. Adaptive treatment allocation and the multi-armed bandit problem. *The Annals of Statistics*, pages 1091–1114, 1987. URL <https://projecteuclid.org/euclid.aos/1176350495>. 108
- T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1): 4–22, 1985. URL <https://core.ac.uk/download/pdf/82425825.pdf>. 108
- J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824, 2008. URL <https://papers.nips.cc/paper/3178-the-epoch-greedy-algorithm-for-multi-armed-bandits-with-side-information>. 108
- T. Lattimore and C. Szepesvári. Bandit algorithms. *preprint*, 2018. URL <https://tor-lattimore.com/downloads/book/book.pdf>. 102, 108
- N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994. URL <https://www.sciencedirect.com/science/article/pii/S0890540184710091>. 51
- B. McMahan and J. Abernethy. Minimax optimal algorithms for unconstrained linear optimization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2724–2732. Curran Associates, Inc., 2013. URL <http://papers.nips.cc/paper/5148-minimax-optimal-algorithms-for-unconstrained-linear-optimization.pdf>. 36
- H. B. McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and L1 regularization. In *Proc. of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 525–533, 2011. URL <http://proceedings.mlr.press/v15/mcmahan11b/mcmahan11b.pdf>. 73
- H. B. McMahan. A survey of algorithms and analysis for adaptive online learning. *The Journal of Machine Learning Research*, 18(1):3117–3166, 2017. URL <http://www.jmlr.org/papers/volume18/14-428/14-428.pdf>. 72
- H. B. McMahan and F. Orabona. Unconstrained online linear learning in Hilbert spaces: Minimax algorithms and normal approximations. In *Proc of the Annual Conference on Learning Theory, COLT*, 2014. URL <https://arxiv.org/abs/1403.0628>. 92
- H. B. McMahan and M. J. Streeter. Adaptive bound optimization for online convex optimization. In *COLT*, 2010. URL <https://static.googleusercontent.com/media/research.google.com/en/pubs/archive/36483.pdf>. 28, 30
- H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica. Ad click prediction: a view from the trenches. In *Proc. of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013. URL <https://static.googleusercontent.com/media/research.google.com/en/pubs/archive/41159.pdf>. 73

- A.S. Nemirovsky and D. Yudin. *Problem complexity and method efficiency in optimization*. Wiley, New York, NY, USA, 1983. URL https://books.google.com/books/about/Problem_Complexity_and_Method_Efficiency.html?id=6ULvAAAAAMAAJ. 50
- Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009. URL <https://link.springer.com/content/pdf/10.1007/s10107-007-0149-x.pdf>. 72
- F. Orabona. Dimension-free exponentiated gradient. In *Advances in Neural Information Processing Systems 26*, pages 1806–1814. Curran Associates, Inc., 2013. URL <https://papers.nips.cc/paper/4920-dimension-free-exponentiated-gradient.pdf>. 36, 92
- F. Orabona and D. Pál. Scale-free algorithms for online linear optimization. In *International Conference on Algorithmic Learning Theory*, pages 287–301. Springer, 2015. URL <https://arxiv.org/abs/1502.05744>. 72, 73
- F. Orabona and D. Pal. Coin betting and parameter-free online learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 577–585. Curran Associates, Inc., 2016. URL <https://arxiv.org/pdf/1602.04128.pdf>. 36, 81, 92, 93
- F. Orabona and D. Pál. Scale-free online learning. *Theoretical Computer Science*, 716:50–69, 2018. URL <https://arxiv.org/pdf/1601.01974.pdf>. Special Issue on ALT 2015. 30, 36
- F. Orabona and T. Tommasi. Training deep networks without learning rates through coin betting. In *Advances in Neural Information Processing Systems*, pages 2160–2170, 2017. URL <https://arxiv.org/abs/1705.07795>. 92
- F. Orabona, K. Crammer, and N. Cesa-Bianchi. A generalized online mirror descent with applications to classification and regression. *Machine Learning*, 99:411–435, 2015. URL <https://arxiv.org/abs/1304.2994>. 72, 73
- A. Rakhlin and K. Sridharan. Online learning with predictable sequences. In *Proc. of the Conference on Learning Theory (COLT)*, volume 30, pages 993–1019, 2013. URL <http://proceedings.mlr.press/v30/Rakhlin13.html>. 73
- S. J. Reddi, S. Kale, and S. Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/pdf?id=ryQu7f-RZ>. 7
- H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952. URL <https://projecteuclid.org/euclid.bams/1183517370>. 108
- R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970. URL <https://press.princeton.edu/titles/1815.html>. 7, 8, 12, 33
- F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.3398&rep=rep1&type=pdf>. 78
- S. Shalev-Shwartz. Online learning: Theory, algorithms, and applications. Technical report, The Hebrew University, 2007. URL <https://www.cs.huji.ac.il/~shais/papers/ShalevThesis07.pdf>. PhD thesis. 23, 48, 72
- S. Shalev-Shwartz and Y. Singer. Online learning meets optimization in the dual. In *International Conference on Computational Learning Theory*, pages 423–437. Springer, 2006. URL <https://storage.googleapis.com/pub-tools-public-publication-data/pdf/25.pdf>. 72
- S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. Technical report, The Hebrew University, 2007a. URL <https://www.cse.huji.ac.il/~shais/papers/ShalevSi07report.pdf>. 73

- S. Shalev-Shwartz and Y. Singer. Convex repeated games and Fenchel duality. In *Advances in neural information processing systems*, pages 1265–1272, 2007b. URL https://ttic.uchicago.edu/~shai/papers/ShalevSi06_fench.pdf. 72
- S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient Solver for SVM. In *Proc. of ICML*, pages 807–814, 2007. URL <https://ttic.uchicago.edu/~nati/Publications/Pegasos.pdf>. 22
- N. Srebro, K. Sridharan, and A. Tewari. Optimistic rates for learning with a smooth loss. *arXiv preprint arXiv:1009.3896*, 2010. URL <https://arxiv.org/abs/1009.3896>. 28
- J. Steinhardt and P. Liang. Adaptivity and optimism: An improved exponentiated gradient algorithm. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 1593–1601, 2014. URL <https://cs.stanford.edu/~плиang/papers/eg-icml2014.pdf>. 73
- M. Streeter and B. McMahan. No-regret algorithms for unconstrained online convex optimization. In *Advances in Neural Information Processing Systems 25*, pages 2402–2410. Curran Associates, Inc., 2012. URL <https://arxiv.org/pdf/1211.2260.pdf>. 36, 92
- M. Streeter and H. B. McMahan. Less regret via online conditioning, 2010. URL <https://arxiv.org/abs/1002.4862>. arXiv:1002.4862. 30
- T. van Erven, W. M. Koolen, S. D. Rooij, and P. Grünwald. Adaptive Hedge. In *Advances in Neural Information Processing Systems*, pages 1656–1664, 2011. URL <https://papers.nips.cc/paper/4191-adaptive-hedge.pdf>. 73
- V. Vovk. Competitive on-line statistics. *International Statistical Review*, 69(2):213–248, 2001. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-5823.2001.tb00457.x>. 73
- V. G. Vovk. Aggregating strategies. *Proc. of Computational Learning Theory, 1990*, pages 371–386, 1990. 51
- M. K. Warmuth and A. K. Jagota. Continuous and discrete-time nonlinear gradient descent: Relative loss bounds and convergence. In *Electronic proceedings of the 5th International Symposium on Artificial Intelligence and Mathematics*, volume 326, 1997. URL <https://users.soe.ucsc.edu/~manfred/pubs/C45.pdf>. 50
- L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010. URL <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/xiao10JMLR.pdf>. 72, 73
- T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proc. of International Conference on Machine learning*, pages 919–926, New York, NY, USA, 2004. ACM. URL <http://tongzhang-ml.org/papers/icml04-stograd.pdf>. 22
- M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proc. of ICML*, pages 928–936, 2003. URL <https://www.aaai.org/Papers/ICML/2003/ICML03-120.pdf>. 14
- M. Zinkevich. *Theoretical guarantees for algorithms in multi-agent settings*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2004. URL <http://martin.zinkevich.org/publications/thesis.ps>. 72